```
; ****************************************************************************
; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.3
; ----------------------------------------------------------------------------
; Last Update: 15/02/2021
; ----------------------------------------------------------------------------
; Beginning: 04/01/2016
; ----------------------------------------------------------------------------
; Assembler: NASM version 2.15 (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

; 15/11/2020
VBE3INFOSEG equ 97E0h ; 512 bytes before Video_Pg_Backup
; 15/12/2020
```

```
                                VBE3MODEINFOSEG equ 97C0h ; 512 bytes before VBE3INFOBLOCK
                                
                                ; 29/11/2020
                                VBE3INFOBLOCK equ 97E00h ; linear address (512 bytes)
                                VBE3MODEINFOBLOCK equ 97C00h ; linear address (256 bytes)
                                VBE3SAVERESTOREBLOCK equ 97600h ; linear address (2048 bytes)
                                VBE3CRTCINFOBLOCK equ 97D80h ; linear address (64 bytes) ; 17/01/2021
                                VBE3BIOSDATABLOCK equ 97000h ; linear address (1536 bytes)
                                VBE3STACKADDR equ 96000h ; linear address (1024 bytes)
                                ; VBE3 32 bit Protected Mode Interface (16 bit) Selectors (in GDT)
                                VBE3CS equ 30h ; _vbe3_CS:
                                VBE3BDS equ 38h ; _vbe3_BDS:
                                VBE3A000 equ 40h ; _A0000Sel:
                                VBE3B000 equ 48h ; _B0000Sel:
                                VBE3B800 equ 50h ; _B8000Sel:
                                VBE3DS equ 58h ; _vbe3_DS:
                                VBE3SS equ 60h ; _vbe3_SS:
                                VBE3ES equ 68h ; _vbe3_ES:
                                KCODE16 equ 70h ; _16bit_CS:
                                ; 14/01/2021
                                ; 06/12/2020
                                VBE3VIDEOSTATE equ 95800h ; 2048 bytes
                                ; 05/01/2021
                                VGAFONT16USER equ 94000h ; 8x16 pixels user font (256 chars)
                                                  ; (reserved/allocated font space: 4096 bytes)
                                
                                VGAFONT8USER equ 95000h   ; 8x8 pixels user font (256 chars)
                                                 ; (reserved/allocated font space: 2048 bytes)
                                ; 17/01/2021
                                ; temporary (initial) location for EDID information
                                VBE3EDIDINFOBLOCK equ 97D00h ; linear address (128 bytes)
                                
                                ; 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. Always 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)
                                
                                ; 29/11/2020
                                ; Free/Reserved memory blocks (in 1st 1MB): 93200h to 96000h (available)
                                ; 06/12/2020
                                ; Free/Reserved memory blocks (in 1st 1MB): 93200h to 95800h (available)
                                
                                ; 15/12/2020
                                LFB_ADDR     equ LFB_Info+LFBINFO.LFB_addr
                                LFB_SIZE     equ LFB_Info+LFBINFO.LFB_size
                                
                                [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[FA6D]                   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[F66D]                   mov   [mem_1m_1k], cx
00000017 8916[F86D]                   mov   [mem_16m_64k], dx
```

```
                                        ; 05/11/2014
                                        ;and   dx, dx
                                        ;jz    short L2
0000001B 81F90004                         cmp     cx, 1024
                                        ;jnb   short L0
0000001F 7351                           jnb    short V0 ; 14/11/2020
                                               ; 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
                                V0:
                                        ; 15/12/2020
00000072 8B36[F86D]                     mov    si, [mem_16m_64k]
00000076 8936[E10E]                     mov    [real_mem_16m_64k], si
                                        ; 15/11/2020
                                        ; 14/11/2020 (TRDOS 386 v2.0.3)
                                        ; check VESA (VBE) VIDEO BIOS version

0000007A B8034F                         mov    ax, 4F03h  ; Return current VBE mode
0000007D CD10                           int    10h
0000007F 83F84F                         cmp    ax, 004Fh  ; successful (vbe) function call
00000082 7567                           jne    short L0   ; not a VESA VBE compatible bios

                                        ;mov   ah, 3
                                        ;;jmp  short v1

                                        ; 15/11/2020
00000084 BBE097                         mov    bx, VBE3INFOSEG  ; 97E0h for current version
00000087 8EC3                           mov    es, bx
00000089 31FF                           xor    di, di
0000008B 2666C70556424532               mov    dword [es:di], 'VBE2' ; request VESA VBE3 info
                                               ; es:di = buffer address (512 bytes)
                                        ;mov   ax, 4F00h ; Return VBE controller information
00000093 86C4                           xchg   al, ah
00000095 CD10                           int    10h

                                        ; dx = cs
                                        ; es = VBE3INFOSEG (97E0h)
                                        ; di = 0
                                        ; ss = (endofkernelfile/16)+16
                                        ; sp = 0FFFEh

00000097 83F84F                         cmp    ax, 004Fh
0000009A 754D                           jne    short V1 ; old vga bios (not VESA compatible)

                                        ; 15/11/2020
0000009C 2666813D56455341               cmp    dword [es:di], 'VESA'
000000A4 7543                           jne    short V1

                                        ;mov   ax, [es:di+4]
                                        ;      ; ax = vbe version in BCD format (0200h or 0300h)
                                        ;mov   [vbe3], ah ; version number (major)

                                        ; 15/11/2020
000000A6 268A4505                       mov    al, [es:di+5]
                                               ; al = high byte of VBE version number (02h or 03h)

000000AA A2[5409]                       mov    [vbe3], al ; version number (major)
                                                          ; 02h or 03h is expected
                                        ; 17/01/2021
                                        ; Read EDID
000000AD B301                           mov    bl, 01h      ; Read EDID
000000AF 31C9                           xor    cx, cx ; Controller unit number
                                                      ; (00 = primary controller)
000000B1 31D2                           xor    dx, dx ; EDID block number = 0
000000B3 B8C097                         mov    ax, VBE3MODEINFOSEG  ; 97C0h for current version
000000B6 8EC0                           mov    es, ax
000000B8 BF0001                         mov    di, VBE3EDIDINFOBLOCK - VBE3MODEINFOBLOCK
                                        ; es:di = temporary address of 128 bytes EDID information
000000BB B8154F                         mov    ax, 4F15h ; VBE/DDC Services
000000BE CD10                           int    10h
                                        ;cmp   ax, 4Fh
                                        ;jne   short v2
000000C0 A2[2F43]                       mov    [edid], al ; 4Fh > 0
                                ;V2:
                                        ; 17/01/2021
000000C3 31FF                           xor    di, di
```

```
                                        ; 15/12/2020
                                        ; Get linear frame buffer info (for VESA VBE mode 118h)
                                        ;mov   si, VBE3MODEINFOSEG  ; 97C0h for current version
                                        ;mov   es, si
                                        ; di = 0
000000C5 B91841                         mov    cx, 04118h  ; 1024*768, 24 bpp, LFB
000000C8 B8014F                         mov    ax, 4F01h ; Return VBE mode information
000000CB CD10                           int    10h
                                        ;cmp   ax, 4Fh
                                        ;jne   short V1
                                        ; 19/12/2020
                                        ;mov   si, [es:di+MODEINFO.PhysBasePtr+2]
                                                   ; hw of LFB base address
                                        ; MODEINFO structure starts from offset -2
000000CD 268B752A                       mov    si, [es:di+MODEINFO.PhysBasePtr] ; hw of LFB addr
000000D1 8936[E30E]                     mov    [def_LFB_addr], si ; k_LFB_size = 3145728 bytes
000000D5 81EE0001                       sub    si, 256

                                        ; 15/12/2020
                                        ; check memory and decrease it to 3.5 GB if it is 4GB
                                        ; (reserve upper memory for LFB)
000000D9 8B3E[F86D]                     mov    di, [mem_16m_64k]
000000DD 893E[E10E]                     mov    [real_mem_16m_64k], di

000000E1 39F7                           cmp    di, si
000000E3 7604                           jna    short V1

000000E5 8936[F86D]                     mov    [mem_16m_64k], si

                                        ; VESA VBE3 video hardware
                                        ; (example: NVIDIA GEFORCE FX550, 256 MB)
                                        ; uses upper memory from 0D0000000h to 0DFFFFFFFh

                                              ;;cmp  di, 0CF00h ; 3328 MB - 16MB
                                              ;jna   short V1  ; <= 3328 MB memory, it is not required
                                                             ; decrease
                                              ;cmp   al, 3
                                              ;jb    short V2
                                              ; VESA VBE 3
                                              ;mov   word [mem_16m_64k], 0CF00h ; 3328 MB - 16MB
                                              ;jmp   short V1
                                        ;V2:
                                              ; VESA VBE 2
                                              ; Check Bochs/Qemu/VirtualBox Emulator
                                              ; LFB base address: 0E0000000h
                                              ;sub   ax, ax ; 0
                                              ;mov   dx, 1CEh ; VBE_DISPI_IOPORT_INDEX
                                              ;out   dx, ax ; VBE_DISPI_INDEX_ID register
                                              ;inc   dx
                                              ;in    ax, dx
                                              ;and   al, 0F0h
                                              ;cmp   ax, 0B0C0h
                                              ;jne   short V1
                                              ;
                                              ; BOCHS/QEMU/VIRTUALBOX
                                              ;mov   word [mem_16m_64k], 0DF00h ; 3584 MB - 16MB
                                        V1:
000000E9 1E                                   push   ds
000000EA 07                                   pop    es ; restore extra data segment
                                        L0:

                                        %include 'diskinit.s' ; 07/03/2015
                                      <1> ; ****************************************************************************
                                      <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.2 - diskinit.s
                                      <1> ; ----------------------------------------------------------------------------
                                      <1> ; Last Update: 29/08/2020
                                      <1> ; ----------------------------------------------------------------------------
                                      <1> ; Beginning: 24/01/2016
                                      <1> ; ----------------------------------------------------------------------------
                                      <1> ; Assembler: NASM version 2.14 (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>       ; 29/08/2020
                                      <1>       ; 17/07/2020
                                      <1>       ; 14/07/2020 (TRDOS 386 v2.0.2)
                                      <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)
000000EB BA7F00                       <1>       mov    dx, 7Fh
                                      <1> L1:
000000EE FEC2                         <1>       inc    dl
000000F0 B441                         <1>       mov    ah, 41h ; Check extensions present
                                      <1>                      ; Phoenix EDD v1.1 - EDD v3
000000F2 BBAA55                       <1>       mov    bx, 55AAh
000000F5 CD13                         <1>       int    13h
000000F7 721A                         <1>       jc     short L2
                                      <1>
000000F9 81FB55AA                     <1>       cmp    bx, 0AA55h
000000FD 7514                         <1>       jne    short L2
```

```
                                <1>       inc    byte [hdc]   ; count of hard disks (EDD present)
                                <1>         mov     [last_drv], dl  ; last hard disk number
                                <1>       mov    bx, hd0_type - 80h
                                <1>       add    bx, dx
                                <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
                                <1>       cmp    dl, 83h       ; drive number < 83h
                                <1>       jb     short L1
                                <1> L2:
                                <1>       ; 23/11/2014
                                <1>       ; 19/11/2014
                                <1>       xor    dl, dl  ; 0
                                <1>       ; 04/02/2016 (esi -> si)
                                <1>       mov    si, fd0_type
                                <1> L3:
                                <1>       ; 14/01/2015
                                <1>       mov    [drv], dl
                                <1>       ;
                                <1>       mov    ah, 08h ; Return drive parameters
                                <1>       int    13h
                                <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>              ;
                                <1>       mov    [si], bl ;  Drive type
                                <1>                     ; 4 = 1.44 MB, 80 track, 3 1/2"
                                <1>       ; 14/01/2015
                                <1>       call   set_disk_parms
                                <1>       ; 10/12/2014
                                <1>       cmp    si, fd0_type
                                <1>       ja     short L4
                                <1>       inc    si ; fd1_type
                                <1>       mov    dl, 1
                                <1>       jmp    short L3
                                <1> L4:
                                <1>       mov    dl, 7Fh
                                <1>       ; 24/12/2014
                                <1>       cmp    byte [hdc], 0 ; EDD present or not ?
                                <1>         ja      L10       ; yes, all fixed disk operations
                                <1>                      ; will be performed according to
                                <1>                      ; present EDD specification
                                <1>
                                <1>       ; 17/07/2020
                                <1>       ; Note: Virtual CPU will not come here while
                                <1>       ; running in QEMU, Bochs, VirtualBox emulators !!!
                                <1>
                                <1>       ; 17/07/2020
                                <1>       ; Older BIOS (INT 13h, AH = 48h is not available)
                                <1> L6:
                                <1>       inc    dl
                                <1>         mov     [drv], dl
                                <1>         mov     [last_drv], dl ; 14/01/2015
                                <1>       mov    ah, 08h ; Return drive parameters
                                <1>       int    13h    ; (conventional function)
                                <1>         jc      L13       ; fixed disk drive not ready
                                <1>         mov     [hdc], dl ; number of drives
                                <1>       ;; 14/01/2013
                                <1>       ;;push cx
                                <1>       call   set_disk_parms
                                <1>       ;;pop  cx
                                <1>       ;
                                <1>       ;;and  cl, 3Fh       ; sectors per track (bits 0-6)
                                <1>         mov     dl, [drv]
                                <1>       mov    bx, 65*4 ; hd0 parameters table (INT 41h)
                                <1>       cmp    dl, 80h
                                <1>       jna    short L7
                                <1>       add    bx, 5*4       ; hd1 parameters table (INT 46h)
                                <1> L7:
                                <1>       xor    ax, ax
                                <1>       mov    ds, ax
                                <1>         mov     si, [bx]
                                <1>         mov     ax, [bx+2]
                                <1>       mov    ds, ax
                                <1>         cmp     cl, [si+FDPT_SPT] ; sectors per track
                                <1>         jne     L12 ; invalid FDPT
                                <1>       mov    di, HD0_DPT
                                <1>       cmp    dl, 80h
                                <1>       jna    short L8
                                <1>       mov    di, HD1_DPT
                                <1> L8:
                                <1>       ; 30/12/2014
                                <1>       mov    ax, DPT_SEGM
                                <1>       mov    es, ax
                                <1>       ; 24/12/2014
                                <1>       mov    cx, 8
                                <1>       rep    movsw  ; copy 16 bytes to the kernel's DPT location
                                <1>       mov    ax, cs
                                <1>       mov    ds, ax
                                <1>
                                <1>       ; 02/02/2015
                                <1>       ;mov   cl, [drv]
                                <1>       ;mov   bl, cl
                                <1>       ;mov   ax, 1F0h
                                <1>       ;and   bl, 1
                                <1>       ;jz    short L9
                                <1>       ;shl   bl, 4
                                <1>       ;sub   ax, 1F0h-170h
```

```
                                <1>
                                <1>       ; 17/07/2020
                                <1>       ; (Only 1F0h port address must be valid for old ROM BIOSes)
00000190 B8F001                 <1>       mov    ax, 1F0h
00000193 B3A0                   <1>       mov    bl, 0A0h
00000195 80FA80                 <1>       cmp    dl, 80h
00000198 7603                   <1>       jna    short L9
                                <1>       ; dl = 81h
0000019A 80C310                 <1>       add    bl, 10h  ; slave disk
                                <1>       ;sub   ax, 1F0h-170h
                                <1> L9:
0000019D AB                     <1>       stosw  ; I/O PORT Base Address (1F0h, 170h)
0000019E 050602                 <1>       add    ax, 206h
000001A1 AB                     <1>       stosw  ; CONTROL PORT Address (3F6h, 376h)
000001A2 88D8                   <1>       mov    al, bl  ; bit 4, master/slave disk bit
                                <1>       ;add   al, 0A0h ; 17/07/2020
000001A4 AA                     <1>       stosb  ; Device/Head Register upper nibble
                                <1>       ;
000001A5 FE06[FB6D]             <1>       inc    byte [drv]
000001A9 BB[806D]               <1>       mov    bx, hd0_type - 80h
000001AC 01CB                   <1>       add    bx, cx
000001AE 800F80                 <1>         or     byte [bx], 80h  ; present sign (when lower nibble is 0)
000001B1 A0[FD6D]               <1>       mov    al, [hdc]
000001B4 FEC8                   <1>       dec    al
000001B6 0F843D01               <1>         jz     L13
000001BA 80FA80                 <1>       cmp    dl, 80h
000001BD 0F867CFF               <1>         jna  L6 ; Max. 2 hard disks  ; 17/07/2020
000001C1 E93301                 <1>         jmp    L13
                                <1> L10:
000001C4 FEC2                   <1>       inc    dl
                                <1>       ; 25/12/2014
000001C6 8816[FB6D]             <1>       mov    [drv], dl
000001CA B408                   <1>       mov    ah, 08h ; Return drive parameters
000001CC CD13                   <1>       int    13h    ; (conventional function)
000001CE 0F822501               <1>         jc     L13
                                <1>       ; 14/01/2015
000001D2 8A16[FB6D]             <1>       mov    dl, [drv]
000001D6 52                     <1>       push   dx
000001D7 51                     <1>       push   cx
000001D8 E82A01                 <1>       call   set_disk_parms
000001DB 59                     <1>       pop    cx
000001DC 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
000001DD BE[5000]               <1>       mov    si, _int13h_48h_buffer
                                <1>       ; 09/07/2016
000001E0 B81E00                 <1>       mov    ax, 001Eh
000001E3 8824                   <1>       mov    [si], ah ; 0
000001E5 46                     <1>       inc    si
000001E6 8904                   <1>       mov    word [si], ax
                                <1>       ; word [si] = 30
                                <1>       ;
000001E8 B448                   <1>       mov    ah, 48h       ; Get drive parameters (EDD function)
000001EA CD13                   <1>       int    13h
000001EC 0F820701               <1>         jc     L13
                                <1>
                                <1>       ; 29/08/2020
                                <1>       ; 04/02/2016 (ebx -> bx)
                                <1>       ; 14/01/2015
000001F0 28FF                   <1>       sub    bh, bh
000001F2 88D3                   <1>       mov    bl, dl
                                <1>       ;sub   bl, 80h
                                <1>       ; 29/08/2020
000001F4 81C3[806D]             <1>       add    bx, (hd0_type - 80h)
                                <1>       ;mov   al, [bx]
000001F8 8A07                   <1>       mov    al, [bx]
000001FA 0C80                   <1>       or     al, 80h
000001FC 8807                   <1>       mov    [bx], al
000001FE 81EB[FE6D]             <1>       sub    bx, hd0_type - 2 ; 15/01/2015
                                <1>       ;add   bx, drv.status
                                <1>       ;mov   [bx], al
                                <1>       ; 29/08/2020
00000202 8887[4A6E]             <1>       mov    [bx+drv.status], al
                                <1>       ; 04/02/2016 (eax -> ax)
                                <1>       ;mov   ax, [si+16]
                                <1>       ; 14/07/2020
                                <1>       ;mov   di, [si+18]
                                <1>       ;;test ax, [si+18]
                                <1>       ;test  ax, di ; 14/07/2020
                                <1>       ;jz    short L10_A0h ; (!) ; 17/07/2020
                                <1>                     ; 'CHS only' disks on EDD system
                                <1>                     ;  are reported with ZERO disk size
                                <1>                     ; (if so, we must not overwrite
                                <1>                     ; calculated disk size in 'set_disk_parms')
                                <1>       ; 29/08/2020
00000206 8B4410                 <1>       mov    ax, [si+16]
00000209 8B7C12                 <1>       mov    di, [si+18]
0000020C 09C0                   <1>       or     ax, ax
0000020E 7504                   <1>       jnz    short L10_LBA
00000210 09FF                   <1>       or     di, di
00000212 740B                   <1>       jz     short L10_A0h
                                <1> L10_LBA:
                                <1>       ;sub   bx, drv.status
00000214 C1E302                 <1>       shl    bx, 2
                                <1>       ;add   bx, drv.size ; disk size (in sectors)
                                <1>       ;mov   [bx], ax
                                <1>       ; 29/08/2020
00000217 8987[2E6E]             <1>       mov    [bx+drv.size], ax
                                <1>       ;mov   ax, [si+18]
                                <1>       ;;mov  [bx], ax
                                <1>       ;mov   [bx+2], ax ; BugFix ; 15/07/2020
```

```
                                <1>       ; 14/07/2020
                                <1>       ;mov   [bx+2], di ; 15/07/2020
                                <1>       ; 29/08/2020
0000021B 89BF[306E]             <1>       mov    [bx+drv.size+2], di
                                <1> L10_A0h:
                                <1>       ; 17/07/2020
                                <1>       ; Note: Virtual CPU will jump here from above (!) test
                                <1>       ;     while running in QEMU
                                <1>
                                <1>       ; Jump here to fix a ZERO (LBA) disk size problem
                                <1>       ; for CHS disks (28/02/2015)
                                <1>
                                <1>       ; 30/12/2014
0000021F BF0000                 <1>       mov    di, HD0_DPT
00000222 88D0                   <1>       mov    al, dl
00000224 83E003                 <1>       and    ax, 3
00000227 C0E005                 <1>       shl    al, 5 ; * 32
0000022A 01C7                   <1>       add    di, ax
0000022C B80090                 <1>       mov    ax, DPT_SEGM
0000022F 8EC0                   <1>       mov    es, ax
                                <1>       ;
00000231 88E8                   <1>       mov    al, ch ; max. cylinder number (bits 0-7)
00000233 88CC                   <1>       mov    ah, cl
00000235 C0EC06                 <1>       shr    ah, 6  ; max. cylinder number (bits 8-9)
00000238 40                     <1>       inc    ax     ; logical cylinders (limit 1024)
00000239 AB                     <1>       stosw
0000023A 88F0                   <1>       mov    al, dh ; max. head number
                                <1>       ;
0000023C 30F6                   <1>       xor    dh, dh  ; 29/08/2020 (dh = 0 is needed here)
                                <1>       ;
0000023E FEC0                   <1>       inc    al
00000240 AA                     <1>       stosb         ; logical heads (limits 256)
00000241 B0A0                   <1>       mov    al, 0A0h ; Indicates translated table
00000243 AA                     <1>       stosb
00000244 8A440C                 <1>       mov    al, [si+12]
00000247 AA                     <1>       stosb         ; physical sectors per track
00000248 31C0                   <1>       xor    ax, ax
                                <1>       ;dec   ax      ; 02/01/2015
0000024A AB                     <1>       stosw         ; precompensation (obsolete)
                                <1>       ;xor   al, al  ; 02/01/2015
0000024B AA                     <1>       stosb         ; reserved
0000024C B008                   <1>       mov    al, 8   ; drive control byte
                                <1>                      ; (do not disable retries,
                                <1>                      ; more than 8 heads)
0000024E AA                     <1>       stosb
0000024F 8B4404                 <1>       mov    ax, [si+4]
00000252 AB                     <1>       stosw         ; physical number of cylinders
                                <1>       ;push  ax      ; 02/01/2015
00000253 8A4408                 <1>       mov    al, [si+8]
00000256 AA                     <1>       stosb         ; physical num. of heads (limit 16)
00000257 29C0                   <1>       sub    ax, ax
                                <1>       ;pop   ax      ; 02/01/2015
00000259 AB                     <1>       stosw         ; landing zone (obsolete)
0000025A 88C8                   <1>       mov    al, cl  ; logical sectors per track (limit 63)
0000025C 243F                   <1>       and    al, 3Fh
0000025E AA                     <1>       stosb
                                <1>       ;sub   al, al  ; checksum
                                <1>       ;stosb
                                <1>       ;
0000025F 83C61A                 <1>       add    si, 26   ; (BIOS) DPTE address pointer
00000262 AD                     <1>       lodsw
00000263 50                     <1>       push   ax ; *  ; (BIOS) DPTE offset
00000264 AD                     <1>       lodsw
00000265 50                     <1>       push   ax ; **       ; (BIOS) DPTE segment
                                <1>       ;
                                <1>       ; checksum calculation
00000266 89FE                   <1>       mov    si, di
00000268 06                     <1>       push   es
00000269 1F                     <1>       pop    ds
                                <1>       ;mov   cx, 16
0000026A B90F00                 <1>       mov    cx, 15
0000026D 29CE                   <1>       sub    si, cx
0000026F 30E4                   <1>       xor    ah, ah
                                <1>       ;del   cl
                                <1> L11:
00000271 AC                     <1>       lodsb
00000272 00C4                   <1>       add    ah, al
00000274 E2FB                   <1>       loop   L11
                                <1>       ;
00000276 88E0                   <1>       mov    al, ah
00000278 F6D8                   <1>       neg    al     ; -x+x = 0
0000027A AA                     <1>       stosb         ; put checksum in byte 15 of the tbl
                                <1>       ;
0000027B 1F                     <1>       pop    ds ; **       ; (BIOS) DPTE segment
0000027C 5E                     <1>       pop    si ; * ; (BIOS) DPTE offset
                                <1>       ;
                                <1>       ; 14/07/2020
                                <1>       ; 0FFFFh:0FFFFh = invalid DPTE address
0000027D 8B0C                   <1>       mov    cx, [si]
0000027F 8B4402                 <1>       mov    ax, [si+2]
00000282 21C1                   <1>       and    cx, ax
00000284 41                     <1>       inc    cx
00000285 7404                   <1>       jz     short L11c ; 0FFFFh:0FFFFh
00000287 0B04                   <1>       or     ax, [si]
00000289 752A                   <1>       jnz    short L11e ; <> 0
                                <1> L11c:
                                <1>       ; 17/07/2020
                                <1>       ; TRDOS 386 v2 DRVINIT assumptions:
                                <1>       ; (also by regarding QEMU, Bochs and VirtualBox settings)
                                <1>       ;     Hard disk 0 port address: 1F0h
                                <1>       ;     Hard disk 1 port address: 1F0h
                                <1>       ;     Hard disk 2 port address: 170h
                                <1>       ;     Hard disk 3 port address: 170h
                                <1>
                                <1>       ; in QEMU, hda=hd0 (1F0h) and hdb=hd1 (1F0h) -IRQ14-
```

```
                        <1>       ;      and hdc=hd2 (170h) and hdd=hd3 (170h) -IRQ15-
                        <1>
0000028B B8F001         <1>       mov    ax, 1F0h
                        <1>
                        <1>       ; 15/07/2020
                        <1>       ; 14/07/2020
                        <1>       ; Invalid DPTE address...
                        <1>       ; Default DPTE parms must be set for DISK_IO_CONT
                        <1>       ; (diskio.s)
                        <1>       ; 17/07/2020
                        <1>
                        <1>       ;mov   bl, dl
                        <1>       ;and   bl, 1
                        <1>       ;jz    short L11d
                        <1>
0000028E B3A0           <1>       mov    bl, 0A0h
                        <1>
00000290 F6C201         <1>       test   dl, 1
00000293 7403           <1>       jz     short L11g  ; Master (as default, for 80h & 82h))
                        <1>       ;shl   bl, 4 ; bl = 16 (bit 4 = 1 -> slave)
00000295 80C310         <1>       add    bl, 10h  ; Slave (as default, for 81h & 83h)
                        <1> L11g:
                        <1>       ; 17/07/2020
00000298 80FA82         <1>       cmp    dl, 82h      ; Hard disk 3 or 4 ?
0000029B 7203           <1>       jb     short L11d ; Primary ATA channel (hd0, hd1)
                        <1>                         ; (port address = 1F0h)
                        <1>
                        <1>       ; Secondary ATA channel (hd2, hd3)
                        <1>       ; (port address = 170h)
                        <1>
0000029D 2D8000         <1>       sub    ax, 1F0h-170h
                        <1> L11d:
                        <1>       ; 14/07/2020
000002A0 AB             <1>       stosw  ; I/O PORT Base Address (1F0h, 170h)
000002A1 050602         <1>       add    ax, 206h
000002A4 AB             <1>       stosw  ; CONTROL PORT Address (3F6h, 376h)
000002A5 88D8           <1>       mov    al, bl  ; Master/Slave bit (0 = Master)
                        <1>       ; 17/07/2020
                        <1>       ;or    al, 0A0h ; CHS (LBA enable bit = 0)
                        <1>                       ; (Bits 5&7, reserved bits  =  1)
000002A7 30E4           <1>       xor    ah, ah
                        <1>       ;stosb ; Device/Head Register upper nibble
000002A9 AB             <1>       stosw
000002AA 30C0           <1>       xor    al, al
000002AC B90500         <1>       mov    cx, 5
000002AF F3AB           <1>       rep    stosw ; clear remain part of the (fake) DPTE
000002B1 0E             <1>       push   cs
000002B2 1F             <1>       pop    ds
000002B3 EB2E           <1>       jmp    short L11f
                        <1> L11e:
                        <1>       ; 23/02/2015
000002B5 57             <1>       push   di
                        <1>       ; ES:DI points to DPTE (FDPTE) location
                        <1>       ;;mov  cx, 8
                        <1>       ;mov   cl, 8
000002B6 B90800         <1>       mov    cx, 8 ; 14/07/2020
000002B9 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)
000002BB 5B             <1>       pop    bx
000002BC 8CC8           <1>       mov    ax, cs
000002BE 8ED8           <1>       mov    ds, ax
000002C0 268B07         <1>       mov    ax, [es:bx]
000002C3 3DF001         <1>       cmp    ax, 1F0h
000002C6 7413           <1>       je     short L11a
000002C8 3D7001         <1>       cmp    ax, 170h
000002CB 740E           <1>       je     short L11a
                        <1>       ; invalidation
                        <1>       ; (because base port address is not 1F0h or 170h)
                        <1>       ;xor   bh, bh
                        <1>       ;mov   bl, dl
                        <1>       ; 29/08/2020
                        <1>       ;xor   dh, dh ; 0
000002CD 89D3           <1>       mov    bx, dx
                        <1>       ;sub   bl, 80h
                        <1>       ;mov   byte [bx+hd0_type], 0 ; not a valid disk drive !
                        <1>         ;or  byte [bx+drv.status+2], 0F0h ; (failure sign)
                        <1>       ; 29/08/2020
000002CF C687[806D]00   <1>       mov    byte [bx+hd0_type-80h], 0
000002D4 808F[CC6D]F0   <1>       or     byte [bx+drv.status-7Eh], 0F0h
000002D9 EB0F           <1>       jmp    short L11b
                        <1> L11a:
                        <1>       ; LBA validation
000002DB 268A4704       <1>       mov    al, [es:bx+4] ; Head register upper nibble
000002DF A840           <1>       test   al, 40h ; LBA bit (bit 6)
000002E1 7507           <1>       jnz    short L11b ; LBA type I/O is OK! (E0h or F0h)
                        <1> L11f:
                        <1>       ; force CHS type I/O for this drive (A0h or B0h)
                        <1>       ;sub   bh, bh
                        <1>       ;mov   bl, dl
                        <1>       ; 29/08/2020
                        <1>       ;xor   dh, dh ; 0
000002E3 89D3           <1>       mov    bx, dx
                        <1>       ;sub   bl, 80h ; 26/02/2015
                        <1>         ;and byte [bx+drv.status+2], 0FEh ; clear bit 0
                        <1>                           ; bit 0 = LBA ready bit
                        <1>       ; 29/08/2020
000002E5 80A7[CC6D]FE   <1>       and    byte [bx+drv.status-7Eh], 0FEh
                        <1>       ; 'diskio' procedure will check this bit !
                        <1> L11b:
000002EA 3A16[FC6D]     <1>       cmp    dl, [last_drv] ; 25/12/2014
000002EE 7307           <1>         jnb     short L13
```

```
464 000002F0 E9D1FE              <1>         jmp     L10
465                              <1>
466                              <1> L12:
467                              <1>         ; Restore data registers
468 000002F3 8CC8                <1>         mov     ax, cs
469 000002F5 8ED8                <1>         mov     ds, ax
470                              <1> L13:
471                              <1>         ; 13/12/2014
472 000002F7 0E                  <1>         push    cs
473 000002F8 07                  <1>         pop     es
474                              <1> L14:
475                              <1>         ; clear keyboard buffer
476 000002F9 B411                <1>         mov     ah, 11h
477 000002FB CD16                <1>         int     16h
478 000002FD 744D                <1>         jz      short L16 ; no keys in keyboard buffer
479 000002FF B010                <1>         mov     al, 10h
480 00000301 CD16                <1>         int     16h
481 00000303 EBF4                <1>         jmp     short L14
482                              <1>
483                              <1> set_disk_parms:
484                              <1>         ; 29/08/2020
485                              <1>         ; 04/02/2016 (ebx -> bx)
486                              <1>         ; 10/07/2015
487                              <1>         ; 14/01/2015
488                              <1>         ;push  bx
489 00000305 28FF                <1>         sub     bh, bh
490 00000307 8A1E[FB6D]          <1>         mov     bl, [drv]
491 0000030B 80FB80              <1>         cmp     bl, 80h
492 0000030E 7203                <1>         jb      short sdp0
493 00000310 80EB7E              <1>         sub     bl, 7Eh
494                              <1> sdp0:
495                              <1>         ;add   bx, drv.status
496                              <1>         ;mov   byte [bx], 80h ; 'Present' flag
497                              <1>         ; 29/08/2020
498 00000313 C687[4A6E]80        <1>         mov     byte [bx+drv.status], 80h
499                              <1>         ;
500 00000318 88E8                <1>         mov     al, ch ; last cylinder (bits 0-7)
501 0000031A 88CC                <1>         mov     ah, cl ;
502 0000031C C0EC06              <1>         shr     ah, 6  ; last cylinder (bits 8-9)
503                              <1>         ;sub   bx, drv.status
504 0000031F D0E3                <1>         shl     bl, 1
505                              <1>         ;add   bx, drv.cylinders
506 00000321 40                  <1>         inc     ax  ; convert max. cyl number to cyl count
507                              <1>         ;mov   [bx], ax
508                              <1>         ; 29/08/2020
509 00000322 8987[046E]          <1>         mov     [bx+drv.cylinders], ax
510 00000326 50                  <1>         push    ax ; ** cylinders
511                              <1>         ;sub   bx, drv.cylinders
512                              <1>         ;add   bx, drv.heads
513 00000327 30E4                <1>         xor     ah, ah
514 00000329 88F0                <1>         mov     al, dh ; heads
515 0000032B 40                  <1>         inc     ax
516                              <1>         ;mov   [bx], ax
517                              <1>         ; 29/08/2020
518 0000032C 8987[126E]          <1>           mov  [bx+drv.heads], ax
519                              <1>         ;sub     bx, drv.heads
520                              <1>           ;add     bx, drv.spt
521 00000330 30ED                <1>         xor     ch, ch
522 00000332 80E13F              <1>         and     cl, 3Fh      ; sectors (bits 0-6)
523                              <1>         ;mov   [bx], cx
524                              <1>           ; 29/08/2020
525 00000335 898F[206E]          <1>         mov     [bx+drv.spt], cx
526                              <1>         ;sub   bx, drv.spt
527 00000339 D1E3                <1>         shl     bx, 1
528                              <1>         ;add   bx, drv.size ; disk size (in sectors)
529                              <1>         ; LBA size = cylinders * heads * secpertrack
530 0000033B F7E1                <1>         mul     cx
531 0000033D 89C2                <1>         mov     dx, ax ; heads*spt
532 0000033F 58                  <1>         pop     ax ; ** cylinders
533 00000340 48                  <1>         dec     ax ; 1 cylinder reserved (!?)
534 00000341 F7E2                <1>         mul     dx ; cylinders * (heads*spt)
535                              <1>         ;mov   [bx], ax
536                              <1>         ;mov   [bx+2], dx
537                              <1>         ; 29/08/2020
538 00000343 8987[2E6E]          <1>         mov     [bx+drv.size], ax
539 00000347 8997[306E]          <1>         mov     [bx+drv.size+2], dx
540                              <1>         ;
541                              <1>         ;pop   bx
542 0000034B C3                  <1>         retn
543                              <1>
544                              <1> L16:  ; 28/05/2016
373
374                                        ; 10/11/2014
375 0000034C FA                            cli     ; Disable interrupts (clear interrupt flag)
376                                                ; Reset Interrupt MASK Registers (Master&Slave)
377                                        ;mov   al, 0FFh     ; mask off all interrupts
378                                        ;out   21h, al             ; on master PIC (8259)
379                                        ;jmp   $+2  ; (delay)
380                                        ;out   0A1h, al     ; on slave PIC (8259)
381                                        ;
382                                        ; Disable NMI
383 0000034D B080                          mov     al, 80h
384 0000034F E670                          out     70h, al             ; set bit 7 to 1 for disabling NMI
385                                        ;23/02/2015
386                                        ;nop                ;
387                                        ;in     al, 71h             ; read in 71h just after writing out to 70h
388                                                            ; for preventing unknown state (!?)
389                                        ;
390                                        ; 20/08/2014
391                                        ; Moving the kernel 64 KB back (to physical address 0)
392                                        ; DS = CS = 1000h
393                                        ; 05/11/2014
394 00000351 31C0                          xor     ax, ax
395 00000353 8EC0                          mov     es, ax ; ES = 0
396                                        ;
```

```
                                        ; 04/07/2016 - TRDOS 386 (64K - 128K kernel)
00000355 31F6                                 xor    si, si
00000357 31FF                           xor    di, di
00000359 B90040                         mov    cx, 16384
0000035C F366A5                         rep    movsd
                                        ;
0000035F 06                             push   es ; 0
00000360 68[6403]                       push   L17
00000363 CB                             retf
                                    L17:
00000364 B90010                         mov    cx, 1000h
00000367 8EC1                           mov    es, cx  ; 1000h
00000369 01C9                           add    cx, cx
0000036B 8ED9                           mov    ds, cx  ; 2000h
0000036D 29F6                           sub    si, si
0000036F 29FF                           sub    di, di
00000371 B90040                         mov    cx, 16384
00000374 F366A5                         rep    movsd

                                        ; Turn off the floppy drive motor
00000377 BAF203                           mov     dx, 3F2h
0000037A EE                               out     dx, al ; 0 ; 31/12/2013

                                        ; Enable access to memory above one megabyte
                                    L18:
0000037B E464                           in     al, 64h
0000037D A802                           test   al, 2
0000037F 75FA                             jnz     short L18
00000381 B0D1                           mov    al, 0D1h     ; Write output port
00000383 E664                           out    64h, al
                                    L19:
00000385 E464                           in     al, 64h
00000387 A802                           test   al, 2
00000389 75FA                             jnz     short L19
0000038B B0DF                           mov    al, 0DFh     ; Enable A20 line
0000038D E660                           out    60h, al
                                   ;L20:
                                        ;
                                        ; Load global descriptor table register

                                          ;mov     ax, cs
                                          ;mov     ds, ax

0000038F 2E0F0116[686D]                   lgdt    [cs:gdtd]

00000395 0F20C0                           mov     eax, cr0
                                        ; or   eax, 1
00000398 40                             inc     ax
00000399 0F22C0                         mov     cr0, eax

                                        ; Jump to 32 bit code

0000039C 66                             db 66h                  ; Prefix for 32-bit
0000039D EA                             db 0EAh             ; Opcode for far jump
0000039E [A4030000]                     dd StartPM          ; Offset to start, 32-bit
                                                            ; (1000h:StartPM = StartPM + 10000h)
000003A2 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
000003A4 66B81000                       mov ax, KDATA           ; Save data segment identifier
000003A8 8ED8                             mov ds, ax              ; Move a valid data segment into DS register
000003AA 8EC0                                mov es, ax              ; Move data segment into ES register
000003AC 8EE0                                mov fs, ax              ; Move data segment into FS register
000003AE 8EE8                                mov gs, ax              ; Move data segment into GS register
000003B0 8ED0                             mov ss, ax              ; Move data segment into SS register
000003B2 BC00000900                         mov esp, 90000h         ; Move the stack pointer to 090000h

                                   clear_bss: ; Clear uninitialized data area
                                        ; 11/03/2015
000003B7 31C0                           xor  eax, eax ; 0
000003B9 B9F4620000                     mov  ecx, (bss_end - bss_start)/4
                                        ;shr  ecx, 2 ; bss section is already aligned for double words
000003BE BF[D6860100]                   mov  edi, bss_start
000003C3 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
000003C5 B108                           mov    cl, 8
000003C7 BF00001000                     mov    edi, MEM_ALLOC_TBL
000003CC F3AB                           rep    stosd           ; clear Memory Allocation Table
                                                               ; for the first 1 MB memory
                                        ;
000003CE 668B0D[F66D0000]               mov    cx, [mem_1m_1k]        ; Number of contiguous KB between
                                                               ; 1 and 16 MB, max. 3C00h = 15 MB.
000003D5 66C1E902                       shr    cx, 2           ; convert 1 KB count to 4 KB count
000003D9 890D[C8890100]                 mov    [free_pages], ecx
000003DF 668B15[F86D0000]               mov    dx, [mem_16m_64k]  ; Number of contiguous 64 KB blocks
                                                               ; between 16 MB and 4 GB.
```

```
000003E6 6609D2                 or      dx, dx
000003E9 7413                   jz      short mi_0
                                ;
000003EB 6689D0                 mov     ax, dx
000003EE C1E004                 shl     eax, 4          ; 64 KB -> 4 KB (page count)
000003F1 0105[C8890100]         add     [free_pages], eax
000003F7 0500100000             add     eax, 4096       ; 16 MB = 4096 pages
000003FC EB07                   jmp     short mi_1
                        mi_0:
000003FE 6689C8                 mov     ax, cx
00000401 66050001               add     ax, 256                 ; add 256 pages for the first 1 MB
                        mi_1:
00000405 A3[C4890100]           mov     [memory_size], eax ; Total available memory in pages
                                                           ; 1 alloc. tbl. bit = 1 memory page
                                                           ; 32 allocation bits = 32 mem. pages
                                ;
0000040A 05FF7F0000             add     eax, 32767      ; 32768 memory pages per 1 M.A.T. page
0000040F 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
00000412 66A3[D8890100]         mov     [mat_size], ax          ; Memory Alloc. Table Size in pages
00000418 C1E00C                 shl     eax, 12                 ; 1 M.A.T. page = 4096 bytes
                                ;                       ; Max. 32 M.A.T. pages (4 GB memory)
0000041B 89C3                   mov     ebx, eax        ; M.A.T. size in bytes
                                ; Set/Calculate Kernel's Page Directory Address
0000041D 81C300001000           add     ebx, MEM_ALLOC_TBL
00000423 891D[C0890100]         mov     [k_page_dir], ebx  ; Kernel's Page Directory address
                                                           ; just after the last M.A.T. page
                                ;
00000429 83E804                 sub     eax, 4          ; convert M.A.T. size to offset value
0000042C A3[D0890100]           mov     [last_page], eax   ; last page ofset in the M.A.T.
                                ;                       ; (allocation status search must be
                                                           ; stopped after here)
00000431 31C0                   xor     eax, eax
00000433 48                     dec     eax             ; FFFFFFFFh (set all bits to 1)
00000434 6651                   push    cx
00000436 C1E905                 shr     ecx, 5          ; convert 1 - 16 MB page count to
                                                           ; count of 32 allocation bits
00000439 F3AB                   rep     stosd
0000043B 6659                   pop     cx
0000043D 40                     inc     eax             ; 0
0000043E 80E11F                 and     cl, 31          ; remain bits
00000441 7412                   jz      short mi_4
00000443 8907                   mov     [edi], eax      ; reset
                        mi_2:
00000445 0FAB07                 bts     [edi], eax      ; 06/11/2014
00000448 FEC9                   dec     cl
0000044A 7404                   jz      short mi_3
0000044C FEC0                   inc     al
0000044E EBF5                   jmp     short mi_2
                        mi_3:
00000450 28C0                   sub     al, al          ; 0
00000452 83C704                 add     edi, 4          ; 15/11/2014
                        mi_4:
00000455 6609D2                 or      dx, dx         ; check 16M to 4G memory space
00000458 7421                   jz      short mi_6     ; max. 16 MB memory, no more...
                                ;
0000045A B900021000             mov     ecx, MEM_ALLOC_TBL + 512 ; End of first 16 MB memory
                                ;
0000045F 29F9                   sub     ecx, edi       ; displacement (to end of 16 MB)
00000461 7406                   jz      short mi_5     ; jump if EDI points to
                                                       ;         end of first 16 MB
00000463 D1E9                   shr     ecx, 1         ; convert to dword count
00000465 D1E9                   shr     ecx, 1         ; (shift 2 bits right)
00000467 F3AB                   rep     stosd          ; reset all bits for reserved pages
                                                       ; (memory hole under 16 MB)
                        mi_5:
00000469 6689D1                 mov     cx, dx         ; count of 64 KB memory blocks
0000046C D1E9                   shr     ecx, 1         ; 1 alloc. dword per 128 KB memory
0000046E 9C                     pushf                  ; 16/11/2014
0000046F 48                     dec     eax            ; FFFFFFFFh (set all bits to 1)
00000470 F3AB                   rep     stosd
00000472 40                     inc     eax            ; 0
00000473 9D                     popf                   ; 16/11/2014
00000474 7305                   jnc     short mi_6
00000476 6648                   dec     ax             ; eax = 0000FFFFh
00000478 AB                     stosd
00000479 6640                   inc     ax             ; 0
                        mi_6:
0000047B 39DF                   cmp     edi, ebx       ; check if EDI points to
0000047D 730A                   jnb     short mi_7     ; end of memory allocation table
                                ;                      ; (>= MEM_ALLOC_TBL + 4906)
0000047F 89D9                   mov     ecx, ebx       ; end of memory allocation table
00000481 29F9                   sub     ecx, edi       ; convert displacement/offset
00000483 D1E9                   shr     ecx, 1         ; to dword count
00000485 D1E9                   shr     ecx, 1         ; (shift 2 bits right)
00000487 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)
00000489 BA00001000             mov     edx, MEM_ALLOC_TBL
                                ;sub    ebx, edx       ; Mem. Alloc. Tbl. size in bytes
                                ;shr    ebx, 12                ; Mem. Alloc. Tbl. size in pages
0000048E 668B0D[D8890100]       mov     cx, [mat_size]         ; Mem. Alloc. Tbl. size in pages
00000495 89D7                   mov     edi, edx
00000497 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!
0000049A 01D7                   add     edi, edx       ; points to M.A.T.'s itself
                                ; eax = 0
0000049C 290D[C8890100]         sub     [free_pages], ecx ; 07/11/2014
                        mi_8:
```

```
607 000004A2 0FB307                     btr    [edi], eax     ; clear bit 0 to bit x (1 to 31)
608                                     ;dec   bl
609 000004A5 FEC9                       dec    cl
610 000004A7 7404                       jz     short mi_9
611 000004A9 FEC0                       inc    al
612 000004AB EBF5                       jmp    short mi_8
613                                 mi_9:
614                                     ;
615                                     ; Reset Kernel's Page Dir. and Page Table bits in M.A.T.
616                                     ;            (allocate pages for system page tables)
617
618                                     ; edx = MEM_ALLOC_TBL
619 000004AD 8B0D[C4890100]             mov    ecx, [memory_size] ; memory size in pages (PTEs)
620 000004B3 81C1FF030000               add    ecx, 1023      ; round up (1024 PTEs per table)
621 000004B9 C1E90A                     shr    ecx, 10              ; convert memory page count to
622                                                            ; page table count (PDE count)
623                                     ;
624 000004BC 51                         push   ecx            ; (**) PDE count (<= 1024)
625                                     ;
626 000004BD 41                         inc    ecx            ; +1 for kernel page directory
627                                     ;
628 000004BE 290D[C8890100]             sub    [free_pages], ecx ; 07/11/2014
629                                     ;
630 000004C4 8B35[C0890100]             mov    esi, [k_page_dir] ; Kernel's Page Directory address
631 000004CA C1EE0C                     shr    esi, 12              ; convert to page number
632                                 mi_10:
633 000004CD 89F0                       mov    eax, esi       ; allocation bit offset
634 000004CF 89C3                       mov    ebx, eax
635 000004D1 C1EB03                     shr    ebx, 3         ; convert to alloc. byte offset
636 000004D4 80E3FC                     and    bl,  0FCh      ; clear bit 0 and bit 1
637                                                           ;   to align on dword boundary
638 000004D7 83E01F                     and    eax, 31              ; set allocation bit position
639                                                           ;  (bit 0 to bit 31)
640                                     ;
641 000004DA 01D3                       add    ebx, edx       ; offset in M.A.T. + M.A.T. address
642                                     ;
643 000004DC 0FB303                     btr    [ebx], eax     ; reset relevant bit (0 to 31)
644                                     ;
645 000004DF 46                         inc    esi            ; next page table
646 000004E0 E2EB                       loop   mi_10          ; allocate next kernel page table
647                                                           ; (ecx = page table count + 1)
648                                     ;
649 000004E2 59                         pop    ecx            ; (**) PDE count (= pg. tbl. count)
650                                     ;
651                                     ; Initialize Kernel Page Directory and Kernel Page Tables
652                                     ;
653                                     ; Initialize Kernel's Page Directory
654 000004E3 8B3D[C0890100]             mov    edi, [k_page_dir]
655 000004E9 89F8                       mov    eax, edi
656 000004EB 0C03                       or     al, PDE_A_PRESENT + PDE_A_WRITE
657                                                           ; supervisor + read&write + present
658 000004ED 89CA                       mov    edx, ecx       ; (**) PDE count (= pg. tbl. count)
659                                 mi_11:
660 000004EF 0500100000                 add    eax, 4096      ; Add page size (PGSZ)
661                                                            ; EAX points to next page table
662 000004F4 AB                         stosd
663 000004F5 E2F8                       loop   mi_11
664 000004F7 29C0                       sub    eax, eax       ; Empty PDE
665 000004F9 66B90004                   mov    cx, 1024       ; Entry count (PGSZ/4)
666 000004FD 29D1                       sub    ecx, edx
667 000004FF 7402                       jz     short mi_12
668 00000501 F3AB                       rep    stosd          ; clear remain (empty) PDEs
669                                     ;
670                                     ; Initialization of Kernel's Page Directory is OK, here.
671                                 mi_12:
672                                     ; Initialize Kernel's Page Tables
673                                     ;
674                                     ; (EDI points to address of page table 0)
675                                     ; eax = 0
676 00000503 8B0D[C4890100]             mov    ecx, [memory_size] ; memory size in pages
677 00000509 89CA                       mov    edx, ecx       ; (***)
678 0000050B B003                       mov    al, PTE_A_PRESENT + PTE_A_WRITE
679                                                           ; supervisor + read&write + present
680                                 mi_13:
681 0000050D AB                         stosd
682 0000050E 0500100000                 add    eax, 4096
683 00000513 E2F8                       loop   mi_13
684 00000515 6681E2FF03                 and    dx, 1023       ; (***)
685 0000051A 740B                       jz     short mi_14
686 0000051C 66B90004                   mov    cx, 1024
687 00000520 6629D1                     sub    cx, dx         ; from dx (<= 1023) to 1024
688 00000523 31C0                       xor    eax, eax
689 00000525 F3AB                       rep    stosd          ; clear remain (empty) PTEs
690                                                           ; of the last page table
691                                 mi_14:
692                                     ;  Initialization of Kernel's Page Tables is OK, here.
693                                     ;
694 00000527 89F8                       mov    eax, edi       ; end of the last page table page
695                                                            ; (beginging of user space pages)
696 00000529 C1E80F                     shr    eax, 15              ; convert to M.A.T. byte offset
697 0000052C 24FC                       and    al, 0FCh       ; clear bit 0 and bit 1 for
698                                                           ; aligning on dword boundary
699
700 0000052E A3[D4890100]               mov    [first_page], eax
701 00000533 A3[CC890100]               mov    [next_page], eax ; The first free page pointer
702                                                            ; for user programs
703                                                            ; (Offset in Mem. Alloc. Tbl.)
704                                     ;
705                                     ; Linear/FLAT (1 to 1) memory paging for the kernel is OK, here.
706                                     ;
707
708                                     ; Enable paging
709                                     ;
710 00000538 A1[C0890100]                 mov     eax, [k_page_dir]
711 0000053D 0F22D8                     mov    cr3, eax
```

```
00000540 0F20C0                     mov    eax, cr0
00000543 0D00000080                 or     eax, 80000000h      ; set paging bit (bit 31)
00000548 0F22C0                     mov    cr0, eax
                                      ;jmp    KCODE:StartPMP

0000054B EA                         db 0EAh             ; Opcode for far jump
0000054C [52050000]                   dd StartPMP             ; 32 bit offset
00000550 0800                       dw KCODE            ; kernel code segment descriptor

                                StartPMP:
                                      ; 06/11//2014
                                      ; Clear video page 0
                                      ;
                                      ; Temporary Code
                                      ;
00000552 B9E8030000                   mov    ecx, 80*25/2
00000557 BF00800B00                   mov    edi, 0B8000h
                                      ; 30/01/2016
                                      ;xor   eax, eax     ; black background, black fore color
0000055C B800070007                   mov    eax, 07000700h  ; black background, light gray fore color
00000561 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)
00000563 BE[324D0100]                 mov    esi, starting_msg
                                      ;; 14/08/2015 (kernel version message will appear
                                      ;;            when protected mode and paging is enabled)
00000568 BF00800B00                   mov    edi, 0B8000h ; 27/08/2014

                                      ; 30/11/2020
                                      ; 14/11/2020 (TRDOS 386 v2.0.3)
                                      ;cmp  byte [vbe3], 3 ; 03h
                                      ;jne  short pkv_1
                                      ;;mov  ah, 0Bh ; Black background, light cyan forecolor
                                      ;; Light red TRDOS 386 version text shows VBE3 is ready !
                                      ;mov  ah, 0Ch ; Black background, light red forecolor
                                      ;jmp  short pkv_2
                                ;pkv_1:
0000056D B40A                         mov    ah, 0Ah ; Black background, light green forecolor
                                ;pkv_2:
                                      ; 20/08/2014
0000056F E8D6030000                   call   printk

                                      ; 'UNIX v7/x86' source code by Robert Nordier (1999)
                                      ; // Set IRQ offsets
                                      ;
                                      ;  Linux (v0.12) source code by Linus Torvalds (1991)
                                      ;
                                                                    ;; ICW1
00000574 B011                         mov    al, 11h                     ; Initialization sequence
00000576 E620                         out    20h, al                     ;      8259A-1
                                      ; jmp  $+2
00000578 E6A0                         out    0A0h, al            ;      8259A-2
                                                                    ;; ICW2
0000057A B020                         mov    al, 20h                     ; Start of hardware ints (20h)
0000057C E621                         out    21h, al                     ;      for 8259A-1
                                      ; jmp  $+2
0000057E B028                         mov    al, 28h                     ; Start of hardware ints (28h)
00000580 E6A1                         out    0A1h, al            ;      for 8259A-2
                                                                    ;
00000582 B004                         mov    al, 04h                     ;; ICW3
00000584 E621                         out    21h, al                     ;      IRQ2 of 8259A-1 (master)
                                      ; jmp  $+2
00000586 B002                         mov    al, 02h                     ;      is 8259A-2 (slave)
00000588 E6A1                         out    0A1h, al            ;
                                                                    ;; ICW4
0000058A B001                         mov    al, 01h                     ;
0000058C E621                         out    21h, al                     ;      8086 mode, normal EOI
                                      ; jmp  $+2
0000058E 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)
00000590 BE[D0490100]                 mov    esi, ilist
00000595 8D3D[D8860100]               lea    edi, [idt]
                                      ; 26/03/2015
0000059B B930000000                   mov    ecx, 48             ; 48 hardware interrupts (INT 0 to INT 2Fh)
                                      ; 02/04/2015
000005A0 BB00000800                   mov    ebx,  80000h
                                rp_sidt1:
000005A5 AD                           lodsd
000005A6 89C2                         mov    edx, eax
000005A8 66BA008E                     mov    dx, 8E00h
000005AC 6689C3                       mov    bx, ax
000005AF 89D8                         mov    eax, ebx     ; /* selector = 0x0008 = cs */
                                                                    ; /* interrupt gate - dpl=0, present */
000005B1 AB                           stosd  ; selector & offset bits 0-15
000005B2 89D0                         mov    eax, edx
```

```
000005B4 AB                             stosd  ; attributes & offset bits 16-23
000005B5 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)
000005B7 B120                           mov    cl, 32       ; 32 software interrupts (INT 30h to INT 4Fh)
                                rp_sidt2:
000005B9 AD                             lodsd
000005BA 21C0                           and    eax, eax
000005BC 7413                           jz     short rp_sidt3
000005BE 89C2                           mov    edx, eax
000005C0 66BA00EE                       mov    dx, 0EE00h    ; P=1b/DPL=11b/01110b
000005C4 6689C3                         mov    bx, ax
000005C7 89D8                           mov    eax, ebx      ; selector & offset bits 0-15
000005C9 AB                             stosd
000005CA 89D0                           mov    eax, edx
000005CC AB                             stosd
000005CD E2EA                           loop   rp_sidt2
000005CF EB16                           jmp    short sidt_OK
                                rp_sidt3:
000005D1 B8[750D0000]                   mov    eax, ignore_int
000005D6 89C2                           mov    edx, eax
000005D8 66BA00EE                       mov    dx, 0EE00h    ; P=1b/DPL=11b/01110b
000005DC 6689C3                         mov    bx, ax
000005DF 89D8                           mov    eax, ebx      ; selector & offset bits 0-15
                                rp_sidt4:
000005E1 AB                             stosd
000005E2 92                             xchg   eax, edx
000005E3 AB                             stosd
000005E4 92                             xchg   edx, eax
000005E5 E2FA                           loop   rp_sidt4
                                sidt_OK:
000005E7 0F011D[6E6D0000]               lidt   [idtd]
                                        ;
                                        ; TSS descriptor setup ; 24/03/2015
000005EE B8[58890100]                   mov    eax, task_state_segment
000005F3 66A3[1A6D0000]                 mov    [gdt_tss0], ax
000005F9 C1C010                         rol    eax, 16
000005FC A2[1C6D0000]                   mov    [gdt_tss1], al
00000601 8825[1F6D0000]                 mov    [gdt_tss2], ah
00000607 66C705[BE890100]68-            mov    word [tss.IOPB], tss_end - task_state_segment
0000060F 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)
00000610 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.)
00000614 0F00D8                         ltr    ax  ; Load task register
                                        ;
                                esp0_set0:
                                        ; 30/07/2015
00000617 8B0D[C4890100]                 mov    ecx, [memory_size] ; memory size in pages
0000061D C1E10C                         shl    ecx, 12 ; convert page count to byte count
00000620 81F900004000                   cmp    ecx, CORE ; beginning of user's memory space (400000h)
                                                         ; (kernel mode virtual address)
00000626 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)
00000628 B900004000                     mov    ecx, CORE
                                esp0_set1:
0000062D 89CC                           mov    esp, ecx ; top of kernel stack (**tss.esp0**)
                                esp0_set_ok:
                                        ; 30/07/2015 (**tss.esp0**)
0000062F 8925[5C890100]                 mov    [tss.esp0], esp
00000635 66C705[60890100]10-              mov     word [tss.ss0], KDATA
0000063D 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!)
                                        ;
0000063E 30C0                           xor    al, al       ; Enable all hardware interrupts!
00000640 E621                           out    21h, al             ; (IBM PC-AT compatibility)
00000642 EB00                           jmp    $+2          ; (All conventional PC-AT hardware
00000644 E6A1                           out    0A1h, al     ;  interrupts will be in use.)
                                                            ; (Even if related hardware component
                                                            ;  does not exist!)
                                        ; Enable NMI
00000646 B07F                           mov    al, 7Fh             ; Clear bit 7 to enable NMI (again)
00000648 E670                           out    70h, al
                                        ; 23/02/2015
0000064A 90                             nop
0000064B 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
0000064D 6631DB                         xor    bx, bx
00000650 66BA0002                       mov    dx, 0200h    ; Row 2, column 0  ; 07/03/2015
00000654 E83E1D0000                     call   _set_cpos    ; 24/01/2016

                                        ; 14/11/2020 (TRDOS 386 v2.0.3)
```

```
                                        ; Check VBE3 protected mode interface/feature(s)
                                        ;cmp   byte [vbe3], 3 ; 03h
                                        ;jne   short display_mem_info
                                        ; 20/11/2020
00000659 803D[54090000]02               cmp    byte [vbe3], 2 ; 02h
00000660 770B                           ja     short vbe3_pmid_chk
                                        ;jb    short display_mem_info
00000662 0F82C2010000                   jb     display_mem_info ; 02/12/2020
00000668 E9F6010000                     jmp    check_boch_plex86_vbe
                                    vbe3_pmid_chk:
0000066D B9EA7F0000                     mov    ecx, 32768 - (20+2) ; 32766 - PMInfoBlockSize
00000672 BE02000C00                     mov    esi, 0C0002h ; 1st word of the video bios rom is 0AA55h

                                    chk_pmi_sign:
                                        ;mov   eax, [esi]
                                        ;cmp   eax, 'PMID'
                                        ; 30/11/2020
                                        ;cmp   al, 'P'
                                        ;jne   short chk_pmi_sign_next
00000677 813E504D4944                   cmp    dword [esi], 'PMID'
                                        ;je    short display_vbios_product_name
0000067D 740E                           je     short verify_pmib_chksum ; 15/11/2020
                                    ;chk_pmi_sign_next:
0000067F 46                             inc    esi  ; inc si
00000680 E2F5                           loop   chk_pmi_sign

                                    not_valid_pmib:
00000682 FE0D[54090000]                 dec    byte [vbe3] ; 2 = VBE2 compatible
                                                           ; (vbe3 feature is defective in this vbios)
                                        ;jmp   short display_mem_info
                                        ; 02/12/2020
00000688 E99D010000                     jmp    display_mem_info

                                    verify_pmib_chksum:
                                        ; 15/11/2020
0000068D 31C0                           xor    eax, eax
                                        ;mov   ecx, eax
                                        ;mov   cl, 20
0000068F 66B91400                       mov    cx, 20 ; 30/11/2020
00000693 56                             push   esi
                                    pmib_sum_bytes:
00000694 AC                             lodsb
00000695 00C4                           add    ah, al
00000697 E2FB                           loop   pmib_sum_bytes
00000699 5E                             pop    esi
0000069A 08E4                           or     ah, ah
0000069C 75E4                           jnz    short not_valid_pmib ; AH must be 0

                                    display_vbios_product_name: ; 14/11/2020

                                        ; ESI points to 'PMID' (0C0000h + 'PMID' offset)

                                        ; 15/11/2020
                                        ;mov   [pmid_addr], si     ; PMInfoBlock offset
                                        ;            ; (in VGA bios, 0C0000h + offset)
                                        ; 02/12/2020
                                        ;push  esi ; * pmid_addr
0000069E 89F7                           mov    edi, esi

                                        ;mov   esi, [VBE3INFOBLOCK+22] ; 097E00h + 16h
                                        ;            ; OemVendorNamePtr (seg16:off16)
000006A0 8B35067E0900                   mov    esi, [VBE3INFOBLOCK+6] ; 097E00h + 06h
                                                    ; OemStringPtr (seg16:off16)
000006A6 30C0                           xor    al, al  ; eax = 0
000006A8 6696                           xchg   ax, si ; ax = offset, si = 0
000006AA C1EE0C                         shr    esi, 12 ; (to convert segment to base addr)
000006AD 6601C6                         add    si, ax  ; esi has an address < 1 MB limit
                                                    ; (OemVendorName is in VBE3INFOBLOCK)
                                                    ; Example:
                                                    ; TRDOS 386 v2.0.3 VESA VBE3 protected mode
                                                    ; interface development reference is ...
                                                    ; NVIDIA GeForce FX5500 VGA BIOS -C000h:029Ch-
                                                    ; Version 4.34.20.54.00 -C000h:02EDh-
                                                    ; ((OemString is 'NVIDIA'))
                                                    ; ((OemVendorName is 'NVIDIA Corporation'))
                                                    ; ((OemProductName is 'NV34 Board - p162-1nz))

                                        ;mov   ah, 0Eh ; Black background, yellow forecolor
                                        ; 30/11/2020
000006B0 B40C                           mov    ah, 0Ch ; Black background, light red forecolor

000006B2 E8E13B0000                     call   print_kmsg

                                        ;mov   ah, 07h

000006B7 BE[69860100]                   mov    esi, vesa_vbe3_bios_msg
                                        ;call  print_kmsg
000006BC E8DD3B0000                     call   pkmsg_loop ; 30/11/2020

                                        ; 02/12/2020
                                        ;pop   edi ; * pmid_addr

                                        ; 30/11/2020
                                        ; 29/11/2020 - TRDOS 386 v2.0.3

                                    struc PMInfo  ;  VESA VBE3 PMInfoBlock ('PMID' block)

00000000 <res 00000004>              .Signature: resb 4  ; db 'PMID' ; PM Info Block Signature
00000004 <res 00000002>              .EntryPoint:      resw 1 ; Offset of PM entry point within BIOS
00000006 <res 00000002>              .PMInitialize: resw 1    ; Offset of PM initialization entry point
00000008 <res 00000002>              .BIOSDataSel:     resw 1       ; Selector to BIOS data area emulation block
0000000A <res 00000002>              .A0000Sel:  resw 1 ; Selector to access A0000h physical mem
0000000C <res 00000002>              .B0000Sel:  resw 1  ; Selector to access B0000h physical mem
```

```
1025 0000000E <res 00000002>          .B8000Sel:  resw 1 ; Selector to access B8000h physical mem
1026 00000010 <res 00000002>          .CodeSegSel:      resw 1 ; Selector to access code segment as data
1027 00000012 <res 00000001>          .InProtectMode: resb 1 ; Set to 1 when in protected mode
1028 00000013 <res 00000001>          .Checksum:  resb 1 ; Checksum byte for structure
1029                                  .size:
1030
1031                                  endstruc
1032
1033                                        ; 29/11/2020
1034                                  vbe3pminit:
1035                                        ; 30/11/2020
1036                                        ;cmp   byte [vbe3], 3 ; is VESA VBE3 PMI ready ?
1037                                        ;jne   short di4
1038
1039                                        ; Allocate 64KB contiguous (kernel) memory block
1040 000006C1 31C0                        xor    eax, eax
1041 000006C3 B900000100                  mov    ecx, 65536
1042 000006C8 E8B05D0000                  call   allocate_memory_block
1043                                       ;jc    short di4
1044 000006CD 0F8250010000                jc     di0 ; 30/11/2020
1045
1046                                       ; of course this block must be in the 1st 16MB
1047                                       ; because vbe3 pmi segments will be 16 bit segments
1048                                       ; (80286 type segment descriptors in GDT)
1049
1050 000006D3 A3[20120300]                 mov    [vbe3bios_addr], eax
1051
1052                                       ; set [pmid_addr] to the new location
1053 000006D8 BE00000C00                  mov    esi, 0C0000h
1054
1055                                       ; 30/11/2020
1056 000006DD 29F7                        sub    edi, esi ; izolate offset
1057 000006DF 01C7                        add    edi, eax ; new address
1058 000006E1 893D[24120300]              mov    [pmid_addr], edi ; new 'PMID' location
1059
1060                                       ; Move VIDEO BIOS from 0C0000h to EAX
1061 000006E7 B900400000                  mov    ecx, 65536/4
1062 000006EC 89C7                        mov    edi, eax ; 30/11/2020
1063 000006EE F3A5                        rep    movsd
1064
1065                                       ; 02/12/2020
1066                                       ; 30/11/2020
1067                                       ; set vbe3 segment selectors
1068
1069                                       ; VBE3CS (VESA VBE3 video bios code segment)
1070 000006F0 BF[226D0000]                mov    edi, _vbe3_CS+2 ; base address bits 0..15
1071 000006F5 66AB                        stosw  ; edi = _vbe3_CS+4
1072 000006F7 C1C810                      ror    eax, 16
1073 000006FA 8807                        mov    [edi], al ; base address, bits 16..23
1074
1075                                       ; VBE3DS ('CodeSegSel' in PMInfoBlock)
1076 000006FC BF[4C6D0000]                mov    edi, _vbe3_DS+4 ; base addr bits 16..23
1077 00000701 8807                        mov    [edi], al
1078 00000703 C1C010                      rol    eax, 16
1079 00000706 668947FE                    mov    [edi-2], ax ; base address, bits 0..15
1080
1081                                       ; VBE3BDS (BIOSDataSel in PMInfoBlock)
1082 0000070A BF[2A6D0000]                mov    edi, _vbe3_BDS+2 ; base addr bits 0..15
1083 0000070F B800700900                  mov    eax, VBE3BIOSDATABLOCK ; 1536 bytes
1084 00000714 66AB                        stosw  ; edi = _vbe3_BDS+4
1085 00000716 C1E810                      shr    eax, 16
1086 00000719 8807                        mov    [edi], al ; base address, bits 16..23
1087
1088                                       ; VBE3SS (1024 bytes)
1089 0000071B BF[526D0000]                mov    edi, _vbe3_SS+2 ; base addr bits 0..15
1090 00000720 B800600900                  mov    eax, VBE3STACKADDR ; size = 1024 bytes
1091 00000725 66AB                        stosw  ; edi = _vbe3_SS+4
1092 00000727 C1E810                      shr    eax, 16
1093 0000072A 8807                        mov    [edi], al ; base address, bits 16..23
1094
1095                                       ; stack pointer (esp) will be set to 1020
1096                                       ; (before VBE3 PMI call)
1097
1098                                       ; VBE3ES (max: 2048 bytes)
1099 0000072C BF[5A6D0000]                mov    edi, _vbe3_ES+2 ; base addr bits 0..15
1100 00000731 B800760900                  mov    eax, VBE3SAVERESTOREBLOCK
1101 00000736 66AB                        stosw  ; edi = _vbe3_ES+4
1102 00000738 C1E810                      shr    eax, 16
1103 0000073B 8807                        mov    [edi], al ; base address, bits 16..23
1104
1105                                       ;Note: low word of _VBE3_ES base address will be
1106                                       ;      set -again- by VBE3 PMI caller routine
1107
1108                                       ; 09/12/2020
1109                                       ;; set pmi32 (as VBE3 PMI is ready)
1110                                       ;inc   byte [pmi32] ; = 1
1111
1112                                       ; KCODE16 (set PMI far return segment)
1113 0000073D BF[626D0000]                mov    edi, _16bit_CS+2 ; base addr bits 0..15
1114 00000742 B8[CA070000]                mov    eax, pminit_return_addr16
1115 00000747 66AB                        stosw  ; edi = _16bit_CS+4
1116 00000749 C1E810                      shr    eax, 16
1117 0000074C 8807                        mov    [edi], al ; base address, bits 16..23
1118
1119                                       ; 30/11/2020
1120                                       ; clear mem from VBE3 BIOS data area emu block
1121                                       ; to end of vbe3 buffers
1122
1123                                       ; 01/12/2020
1124 0000074E BF00700900                  mov    edi, VBE3BIOSDATABLOCK ; 97000h
1125                                       ;mov   cx, (VBE3INFOBLOCK-VBE3BIOSDATABLOCK)/4
1126                                       ;      ; ecx = 3584/4 double words
1127                                       ; 21/12/2020
1128 00000753 66B90003                    mov    cx, (VBE3MODEINFOBLOCK-VBE3BIOSDATABLOCK)/4
1129                                             ; ecx = 3072/4 double words
```

```
                                        ;xor   eax, eax
00000757 30C0                           xor    al, al
00000759 F3AB                           rep    stosd

                                        ; Filling PMInfoBlock selector fields
0000075B 8B3D[24120300]                 mov    edi, [pmid_addr]
00000761 66C747083800                   mov    word [edi+PMInfo.BIOSDataSel], VBE3BDS
00000767 66C7470A4000                   mov    word [edi+PMInfo.A0000Sel], VBE3A000
0000076D 66C7470C4800                   mov    word [edi+PMInfo.B0000Sel], VBE3B000
00000773 66C7470E5000                   mov    word [edi+PMInfo.B8000Sel], VBE3B800
00000779 66C747105800                   mov    word [edi+PMInfo.CodeSegSel], VBE3DS
0000077F C6471201                       mov    byte [edi+PMInfo.InProtectMode], 1

                                        ; Calculate and write checksum byte
00000783 89FE                           mov    esi, edi
00000785 B113                           mov    cl, PMInfo.size - 1
                                        ;xor   ah, ah
                                pmid_chksum:
00000787 AC                             lodsb
00000788 00C4                           add    ah, al
0000078A E2FB                           loop   pmid_chksum
0000078C F6DC                           neg    ah ; 1 -> 255, 255 -> 1
0000078E 8826                           mov    byte [esi], ah  ; checksum

                                        ; far call PM initialization
                                        ; (VBE3 video bios will return via 'retf')

00000790 668B4706                       mov    ax, [edi+PMInfo.PMInitialize]
                                        ; 30/11/2020
00000794 C1E010                         shl    eax, 16 ; save entry address in hw
                                        ; ax = 0

                                        ; 02/12/2020
00000797 68[EE070000]                   push   pminit_ok ; normal, near return address

                                        ; 30/11/2020
                                _VBE3PMI_fcall:
                                        ; ax = function, hw of eax = entry address
0000079C 9C                             pushf  ; save 32 bit flags
0000079D 56                             push   esi ; *
0000079E 55                             push   ebp ; **

0000079F 89E5                           mov    ebp, esp ; save 32 bit stack pointer

000007A1 89C6                           mov    esi, eax

000007A3 FA                             cli

                                        ; Disable interrupts (clear interrupt flag)
                                        ; Reset Interrupt MASK Registers (Master&Slave)
000007A4 B0FF                           mov    al, 0FFh     ; mask off all interrupts
000007A6 E621                           out    21h, al            ; on master PIC (8259)
000007A8 EB00                           jmp    $+2  ; (delay)
000007AA E6A1                           out    0A1h, al     ; on slave PIC (8259)

                                        ; 02/12/2020
000007AC 66B86000                       mov    ax, VBE3SS
000007B0 8ED0                           mov    ss, ax

000007B2 BCFC030000                     mov    esp, 1020 ; 30/11/2020

                                        ; 01/12/2020
                                        ;lss   esp, [stack16]

000007B7 C1E810                         shr    eax, 16      ; now, entry address is in lw

                                        ; 30/11/2020 - 16 bit pm selector test (OK)
                                        ; (32 bit stack push/pop & retf with 32 bit code segment)
                                        ; (16 bit stack push/pop with 16 bit code segment)

                                        ; return
                                        ;push  KCODE16
                                        ;push  0 ; 30/11/2020 (pminit_return_addr16)

                                        ; 30/11/2020 (16 bit stack during retf from video bios)
000007BA C7042400007000                 mov    dword [esp], KCODE16 << 16
                                                             ; ip = 0, cs = KCODE16
                                        ; 01/12/2020
                                        ;mov   dword [VBE3STACKADDR+1020], KCODE16*65536

                                        ;mov   [jumpfar16], eax

                                        ; 02/12/2020
                                        ; 30/11/2020 (32 bit stack during retf from kernel)
                                        ; far jump/call via retf
000007C1 6A30                           push   VBE3CS ; VBE3 video bios's code segment
000007C3 50                             push   eax ; PMInitialize or EntryPoint

000007C4 6689F0                         mov    ax, si ; restore function

                                        ; 02/12/2020
000007C7 31F6                           xor    esi, esi ; (not necessary, it is not used)

000007C9 CB                             retf   ; far return (to 16 bit code segment)

                                        ; 01/12/2020
                                        ;db    0EAh  ; far jump to 16 bit code segment
                                ;jumpfar16:
                                        ;dd    0
                                        ;dw    VBE3CS

                                ;stack16:
                                        ;dd    1020
                                        ;dw    VBE3SS

```

```
                                        align 2
                                
                                pminit_return_addr16:
                                        ; 02/12/2020
                                        ; 30/11/2020
                                        ;;db   66h              ; Prefix for 32-bit
                                        ;db    0EAh             ; Opcode for far jump
                                        ;dd    pminit_return_addr32 ; 32 bit Offset
                                        ;dw    KCODE            ; 32 bit code segment
                                        ; 01/12/2020
000007CA EA[D1070000]0800               jmp    KCODE:pminit_return_addr32
                                
                                pminit_return_addr32:
                                        ; restore 32 bit kernel selectors and 32 bit stack addr
000007D1 BE10000000                     mov    esi, KDATA
000007D6 8EDE                           mov    ds, si
000007D8 8EC6                           mov    es, si
000007DA 8ED6                           mov    ss, si
000007DC 89EC                           mov    esp, ebp  ; top of stack = iretd return addr
                                
000007DE 5D                             pop    ebp ; **
000007DF 5E                             pop    esi ; *
000007E0 9D                             popf   ; restore 32 bit flags
                                
                                        ; enable interrupts
                                
000007E1 FA                             cli
                                
                                        ; 21/12/2020
000007E2 50                             push   eax
                                
000007E3 30C0                           xor    al, al    ; Enable all hardware interrupts!
000007E5 E621                           out    21h, al         ; (IBM PC-AT compatibility)
000007E7 EB00                           jmp    $+2       ; (All conventional PC-AT hardware
000007E9 E6A1                           out    0A1h, al   ;  interrupts will be in use.)
                                                          ; (Even if related hardware component
                                                          ;  does not exist!)
000007EB 58                             pop    eax
                                
000007EC FB                             sti
                                
                                        ; top of stack = return address
                                        ; ('pminit_ok' for PMinit)
                                
000007ED C3                             retn
                                
                                pminit_ok:
                                        ; 03/12/2020
                                        ; (set [pmid_addr] to PMI entry point for next calls)
000007EE 8305[24120300]04               add    dword [pmid_addr], PMInfo.EntryPoint ; + 4
                                
                                        ; 17/01/2021
                                        ; copy EDID data from temporary location to final address
000007F5 803D[2F430000]4F               cmp    byte [edid], 4Fh
000007FC 7511                           jne    short vbe3h_chcl
000007FE B920000000                     mov    ecx, 32 ; 128 bytes, 32 dwords
00000803 BE007D0900                     mov    esi, VBE3EDIDINFOBLOCK ; 97D00h
00000808 BF[9C110300]                   mov    edi, edid_info
0000080D F3A5                           rep    movsd
                                        ; 17/01/2021
                                vbe3h_chcl:
                                        ; 16/01/2021
                                        ;; 06/12/2020
                                        ;; Save video mode 03h regs/dac/bios state
                                        ;
                                        ;mov   ax, 4F04h ; VESA VBE Function 04h
                                        ;              ; Save/Restore State
                                        ;sub   dl, dl ; 0 = return buffer size
                                        ;mov   cx, 0Fh  ; ctrl/bios/dac/regs
                                        ;
                                        ;call  int10h_32bit_pmi
                                        ;; bx = number of 64-byte blocks to hold the state buff
                                        ;
                                        ;;mov  [vbe3stbufsize], bx
                                        ;; 16/01/2021
                                        ;or    word [vbe3stbsflags], 32768  ; set bit 15
                                        ;mov   ax, bx
                                        ;shl   ax, 6  ; * 64
                                        ;mov   [vbestatebufsize+30], ax
                                        ;
                                        ;; 06/12/2020
                                        ;; check 'vbe3stbufsize' (it must be <= 32)
                                        ;
                                        ;cmp   bx, 32
                                        ;ja    short display_mem_info ; light red forecolor
                                        ;
                                        ;; 16/01/2021
                                        ;or    byte [vbe3stbsflags], 1 ; set bit 0
                                
                                        ; 30/11/2020
                                        ; Change VESA VBE3 BIOS text color in order to give
                                        ; "VBE3 PMI initialization has been successed" meaning
                                
0000080F BE40810B00                     mov    esi, 0B8000h + 160*2 ; row 2
00000814 89F7                           mov    edi, esi
                                vbe3h_chcl_next:
00000816 66AD                           lodsw
00000818 80FC0C                         cmp    ah, 0Ch      ; light red forecolor
0000081B 750D                           jne    short display_mem_info
0000081D B40E                           mov    ah, 0Eh ; yellow forecolor
0000081F 66AB                           stosw
00000821 EBF3                           jmp    short vbe3h_chcl_next
                                
                                di0:
                                        ; 30/11/2020
```

```
                                        ; Memory allocation error !
00000823 C605[54090000]00               mov   byte [vbe3], 0 ; disable VBE3

                                    display_mem_info:
                                        ; 19/12/2020
                                        ; temporary
0000082A E8853A0000                     call  default_lfb_info
                                        ;
                                        ; 06/11/2014
0000082F E80B3A0000                     call  memory_info
                                        ; 14/08/2015
                                        ;call getch ; 28/02/2015
                                    drv_init:
00000834 FB                             sti   ; Enable Interrupts
                                        ; 06/02/2015
00000835 8B15[006E0000]                 mov   edx, [hd0_type] ; hd0, hd1, hd2, hd3
0000083B 668B1D[FE6D0000]               mov   bx, [fd0_type] ; fd0, fd1
                                        ; 22/02/2015
00000842 6621DB                         and   bx, bx
00000845 756C                           jnz   short di1
                                        ;
00000847 09D2                           or    edx, edx
00000849 757A                           jnz   short di2
                                        ;
                                    setup_error:
0000084B BE[D64C0100]                   mov   esi, setup_error_msg
                                    psem:
00000850 AC                             lodsb
00000851 08C0                           or    al, al
                                        ;jz   short haltx ; 22/02/2015
00000853 747B                           jz    short di3
00000855 56                             push  esi
                                        ; 13/05/2016
00000856 BB07000000                     mov   ebx, 7 ; Black background,
                                              ; light gray forecolor
                                              ; Video page 0 (BH=0)
0000085B E89D1A0000                     call  _write_tty
00000860 5E                             pop   esi
00000861 EBED                           jmp   short psem

                                    check_boch_plex86_vbe:
                                        ; 20/11/2020
                                        ; check Bochs/Plex86 VGABios VBE extension
                                        ; (check if TRDOS 386 v2 is running on emulators)
                                        ; BOCHS/QEMU/VIRTUALBOX
                                        ;
                                        ; ref: vbe_display_api.txt

                                        ; bochs/plex86 VGAbios VBE source code
                                        ;  by Jeroen Janssen (2002)
                                        ;  and Volker Ruppert (2003-2020)

00000863 29C0                           sub   eax, eax ; 0
00000865 66BACE01                       mov   dx, 1CEh ; VBE_DISPI_IOPORT_INDEX
00000869 66EF                           out   dx, ax ; VBE_DISPI_INDEX_ID register
                                        ;mov  ax, 0B0C0h ; VBE_DISPI_ID0
                                        ;mov  dx, 1CFh ; VBE_DISPI_IOPORT_DATA
0000086B 6642                           inc   dx
                                        ;out  dx, ax
                                        ;nop
0000086D 66ED                           in    ax, dx
0000086F 80FCB0                         cmp   ah, 0B0h
                                        ;jne  short not_boch_qemu_vbe
00000872 75B6                           jne   short display_mem_info
00000874 3CC5                           cmp   al, 0C5h ; it must be 0B0C4h or 0B0C5h ..
                                        ;ja   short not_boch_qemu_vbe
00000876 77B2                           ja    short display_mem_info
00000878 3CC0                           cmp   al, 0C0h ; 0B0C0h to 0B0C5h .. ; Qemu
                                        ;cmp  al, 0C4h ; 0BC04h or 0B0C5h is OK ; Bochs
                                        ;jb   short not_boch_qemu_vbe
0000087A 72AE                           jb    short display_mem_info

                                        ; save VESA VBE2 bios (bochs/qemu) signature
                                        ; for enabling VBE2 functions in TRDOS 386 v2 kernel
0000087C A2[55090000]                   mov   [vbe2bios], al ; 0C4h or 0C5h (for BOCHS)
                                                             ; (0C0h-0C5h for QEMU)
00000881 C605[72860100]32               mov   byte [vbe_vnumber], "2"
                                        ; 26/11/2020
                                        ; "BOCHS/QEMU/VIRTUALBOX VBE2 Video BIOS ..".
00000888 BE[54860100]                   mov   esi, vbe2_bochs_vbios ; BOCH/QEMU vbios msg
0000088D B40E                           mov   ah, 0Eh  ; Yellow font
0000088F E8043A0000                     call  print_kmsg

                                        ; this is not necessary ! (20/11/2020)
00000894 803D[55090000]C4               cmp   byte [vbe2bios], 0C4h
0000089B 728D                           jb    display_mem_info   ; (QEMU)

                                        ; Display kernel version message if 0E9h hack port
                                        ; is enabled (bochs emulator feature)
0000089D 66BAE900                       mov   dx, 0E9h ; hack port for BOCHS
000008A1 BE[AD860100]                   mov   esi, kernel_version_msg
                                    kvmsg_next_char:
000008A6 AC                             lodsb
000008A7 08C0                           or    al, al
000008A9 7505                           jnz   short put_kvmsg_in_hack_port
                                    not_boch_qemu_vbe:
000008AB E97AFFFFFF                     jmp   display_mem_info
                                    put_kvmsg_in_hack_port:
000008B0 EE                             out   dx, al
000008B1 EBF3                           jmp   short kvmsg_next_char

                                    di1:
                                        ; supress 'jmp short T6'
                                        ;  (activate fdc motor control code)
000008B3 66C705[B6090000]90-            mov   word [T5], 9090h ; nop
```

```
1444 000008BB 90
1445                                  ;
1446                                  ;mov   ax, int_0Eh  ; IRQ 6 handler
1447                                  ;mov   di, 0Eh*4    ; IRQ 6 vector
1448                                  ;stosw
1449                                  ;mov   ax, cs
1450                                  ;stosw
1451                                  ;; 16/02/2015
1452                                    ;;mov     dword [DISKETTE_INT], fdc_int ; IRQ 6 handler
1453                                  ;
1454 000008BC E82A490000               CALL   DSKETTE_SETUP; Initialize Floppy Disks
1455                                  ;
1456 000008C1 09D2                    or     edx, edx
1457 000008C3 740B                      jz      short di3
1458                                 di2:
1459 000008C5 E868490000              call         DISK_SETUP   ; Initialize Fixed Disks
1460 000008CA 0F827BFFFFFF              jc      setup_error
1461                                 di3:
1462 000008D0 E83E390000              call   setup_rtc_int; 22/05/2015 (dsectrpm.s)
1463                                  ;
1464 000008D5 E892420100              call   display_disks ; 07/03/2015  (Temporary)
1465                                 ;haltx:
1466                                  ; 14/08/2015
1467                                  ;call  getch ; 22/02/2015
1468                                  ;sti   ; Enable interrupts (for CPU)
1469                                 ;     ; 29/01/2016
1470                                 ;     sub    ah, ah ;  read time count
1471                                 ;     call   int1Ah
1472                                 ;     mov    edx, ecx ; 18.2 * seconds
1473                                 ;md_info_msg_wait1:
1474                                 ;     ; 29/01/2016
1475                                 ;     mov    ah, 1
1476                                 ;     call   int16h
1477                                 ;     jz     short md_info_msg_wait2
1478                                 ;     xor    ah, ah ; 0
1479                                 ;       call    int16h
1480                                 ;     jmp    short md_info_msg_ok
1481                                 ;md_info_msg_wait2:
1482                                 ;     sub    ah, ah  ; read time count
1483                                 ;     call   int1Ah
1484                                 ;     cmp    edx, ecx ; ; 18.2 * seconds
1485                                 ;     jna    short md_info_msg_wait3
1486                                 ;     xchg   edx, ecx
1487                                 ;md_info_msg_wait3:
1488                                 ;     sub    ecx, edx
1489                                 ;     cmp    ecx, 127 ; 7 seconds (18.2 * 7)
1490                                 ;     jb     short md_info_msg_wait1
1491                                 ;md_info_msg_ok:
1492
1493                                  ; 15/12/2020
1494                                  ; set initial values of LFB parameters
1495
1496 000008DA 803D[54090000]02            cmp    byte [vbe3], 2
1497 000008E1 7214                    jb     short di4
1498
1499                                  ;mov   ax, [def_LFB_addr]
1500                                  ;shl   eax, 16
1501                                  ;mov   [LFB_ADDR], eax
1502                                  ;mov   eax, 1024*768*3
1503                                  ;mov   [LFB_SIZE], eax
1504
1505 000008E3 BEFE7B0900                mov    esi, VBE3MODEINFOBLOCK - 2
1506 000008E8 66C7061801              mov    word [esi], 0118h ; default vbe mode
1507                                                            ; 1024*768, 24bpp
1508 000008ED E8AD310000              call   set_lfbinfo_table
1509
1510 000008F2 E8DC600000              call   allocate_lfb_pages_for_kernel
1511                                 di4:
1512                                  ; 08/09/2016
1513 000008F7 0F20C0                  mov    eax, cr0
1514 000008FA A810                    test   al, 10h  ; Bit 4, ET (Extension Type)
1515 000008FC 7408                    jz     short sysinit
1516                                  ; 27/02/2017
1517 000008FE FE05[7C960100]          inc    byte [fpready]
1518                                  ; 80387 (FPU) is ready
1519 00000904 DBE3                    fninit ; Initialize Floating-Point Unit
1520                                 sysinit:
1521                                  ; 30/06/2015
1522 00000906 E82B6B0000              call   sys_init
1523                                  ;
1524                                  ;jmp   cpu_reset ; 22/02/2015
1525                                 hang:
1526                                  ; 23/02/2015
1527                                  ;sti                ; Enable interrupts
1528 0000090B F4                      hlt
1529                                  ;
1530                                  ;nop
1531                                  ;; 03/12/2014
1532                                  ;; 28/08/2014
1533                                  ;mov   ah, 11h
1534                                  ;call  getc
1535                                  ;jz      _c8
1536                                  ;
1537                                  ; 23/02/2015
1538                                  ; 06/02/2015
1539                                  ; 07/09/2014
1540 0000090C 31DB                    xor    ebx, ebx
1541 0000090E 8A1D[EE890100]          mov    bl, [ptty]   ; active_page
1542 00000914 89DE                    mov    esi, ebx
1543 00000916 66D1E6                  shl    si, 1
1544 00000919 81C6[F0890100]          add    esi, ttychr
1545 0000091F 668B06                  mov    ax, [esi]
1546 00000922 6621C0                  and    ax, ax
1547                                  ;jz    short _c8
1548 00000925 74E4                    jz     short hang
```

```
1549 00000927 66C7060000                mov    word [esi], 0
1550 0000092C 80FB03                    cmp    bl, 3        ; Video page 3
1551                                    ;jb    short _c8
1552 0000092F 72DA                      jb     short hang
1553                                    ;
1554                                    ; 13/05/2016
1555                                    ; 07/09/2014
1556                            nxtl:
1557 00000931 6653                      push   bx
1558 00000933 66BB0E00                  mov    bx, 0Eh      ; Yellow character
1559                                                        ; on black background
1560                                                        ; bh = 0 (video page 0)
1561                                                        ; Retro UNIX 386 v1 - Video Mode 0
1562                                                        ; (PC/AT Video Mode 3 - 80x25 Alpha.)
1563 00000937 6650                      push   ax
1564 00000939 E8BF190000                call   _write_tty
1565 0000093E 6658                      pop    ax
1566 00000940 665B                      pop    bx
1567 00000942 3C0D                      cmp    al, 0Dh             ; carriage return (enter)
1568                                    ;jne   short _c8
1569 00000944 75C5                      jne    short hang
1570 00000946 B00A                      mov    al, 0Ah             ; next line
1571 00000948 EBE7                      jmp    short nxtl
1572
1573                            ;_c8:
1574                            ;       ; 25/08/2014
1575                            ;       cli                        ; Disable interrupts
1576                            ;       mov    al, [scounter + 1]
1577                            ;       and    al, al
1578                            ;       jnz    hang
1579                            ;       call   rtc_p
1580                            ;       jmp     hang
1581
1582                                    ; 27/08/2014
1583                                    ; 20/08/2014
1584                            printk:
1585                                      ;mov    edi, [scr_row]
1586                            pkl:
1587 0000094A AC                        lodsb
1588 0000094B 08C0                      or     al, al
1589 0000094D 7404                      jz     short pkr
1590 0000094F 66AB                      stosw
1591 00000951 EBF7                      jmp    short pkl
1592                            pkr:
1593 00000953 C3                        retn
1594
1595                            ; 14/11/2020 (TRDOS 386 v2.0.3)
1596 00000954 00                vbe3: db 0  ; VESA VBE version (must be 03h)
1597                                        ; for using video bios calls in protected mode
1598                            vbe2bios:
1599 00000955 B0                        db 0B0h ;
1600                            ;pmid_addr:
1601                                    ;dw 0  ; > 0 if 'PMID' sign is found
1602                                    ;      ;       ('pmid' offset addr in VGA bios seg, 0C000h)
1603                                    ;; 02/12/2020
1604                                    ;dw 0  ; 32 bit address in pmid_addr
1605
1606                            ; 28/02/2017
1607                            ; 22/01/2017
1608                            ; 15/01/2017
1609                            ; 14/01/2017
1610                            ; 02/01/2017
1611                            ; 25/12/2016
1612                            ; 19/12/2016
1613                            ; 10/12/2016 (callback)
1614                            ; 06/06/2016
1615                            ; 23/05/2016
1616                            ; 22/05/2016 - TRDOS 386 (TRDOS v2.0) Timer Event Modifications
1617                            ; 25/07/2015
1618                            ; 14/05/2015 (multi tasking -time sharing- 'clock', x_timer)
1619                            ; 17/02/2015
1620                            ; 06/02/2015 (unix386.s)
1621                            ; 11/12/2014 - 22/12/2014 (dsectrm2.s)
1622                            ;
1623                            ; IBM PC-XT Model 286 Source Code - BIOS2.ASM (06/10/85)
1624                            ;
1625                            ;-- HARDWARE INT  08 H - ( IRQ LEVEL 0 ) ---------------------------------------
1626                            ;     THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM FROM CHANNEL 0 OF        :
1627                            ;     THE 8254 TIMER.  INPUT FREQUENCY IS 1.19318 MHZ AND THE DIVISOR        :
1628                            ;     IS 65536, RESULTING IN APPROXIMATELY 18.2 INTERRUPTS EVERY SECOND.     :
1629                            ;                                                                            :
1630                            ;     THE INTERRUPT HANDLER MAINTAINS A COUNT (40:6C) OF INTERRUPTS SINCE    :
1631                            ;     POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.        :
1632                            ;     THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT (40:40)  :
1633                            ;     OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE           :
1634                            ;     DISKETTE MOTOR(s), AND RESET THE MOTOR RUNNING FLAGS.                   :
1635                            ;     THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH           :
1636                            ;     INTERRUPT 1CH AT EVERY TIME TICK.  THE USER MUST CODE A           :
1637                            ;     ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE.        :
1638                            ;-------------------------------------------------------------------------------
1639                            ;
1640
1641                            timer_int:  ; IRQ 0
1642                            ;int_08h:   ; Timer
1643                                    ; 14/10/2015
1644                                    ; Here, we are simulating system call entry (for task switch)
1645                                    ; (If multitasking is enabled,
1646                                    ; 'clock' procedure may jump to 'sysrelease')
1647
1648 00000956 1E                      push   ds
1649 00000957 06                      push   es
1650 00000958 0FA0                    push   fs
1651 0000095A 0FA8                    push   gs
1652
1653 0000095C 60                      pushad ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi
```

```
1654 0000095D 66B91000                  mov     cx, KDATA
1655 00000961 8ED9                        mov     ds, cx
1656 00000963 8EC1                        mov     es, cx
1657 00000965 8EE1                        mov     fs, cx
1658 00000967 8EE9                        mov     gs, cx
1659
1660 00000969 0F20D9                    mov    ecx, cr3
1661 0000096C 890D[5C040300]            mov    [cr3reg], ecx ; save current cr3 register value/content
1662
1663                                    ; 14/01/2017
1664 00000972 3B0D[C0890100]            cmp    ecx, [k_page_dir]
1665 00000978 7409                      je     short T3
1666
1667 0000097A 8B0D[C0890100]            mov    ecx, [k_page_dir]
1668 00000980 0F22D9                    mov    cr3, ecx
1669                                T3:
1670                                    ;sti                        ; INTERRUPTS BACK ON
1671 00000983 66FF05[408A0100]          INC    word [TIMER_LOW]     ; INCREMENT TIME
1672 0000098A 7507                      JNZ    short T4             ; GO TO TEST_DAY
1673 0000098C 66FF05[428A0100]          INC    word [TIMER_HIGH]    ; INCREMENT HIGH WORD OF TIME
1674                                T4:                             ; TEST_DAY
1675 00000993 66833D[428A0100]18        CMP    word [TIMER_HIGH],018H    ; TEST FOR COUNT EQUALING 24 HOURS
1676 0000099B 7519                      JNZ    short T5             ; GO TO DISKETTE_CTL
1677 0000099D 66813D[408A0100]B0-       CMP    word [TIMER_LOW],0B0H
1677 000009A5 00
1678 000009A6 750E                      JNZ    short T5             ; GO TO DISKETTE_CTL
1679
1680                                ;-----      TIMER HAS GONE 24 HOURS
1681                                    ;;SUB  AX,AX
1682                                    ;MOV   [TIMER_HIGH],AX
1683                                    ;MOV   [TIMER_LOW],AX
1684 000009A8 29C0                      sub    eax, eax
1685 000009AA A3[408A0100]              mov    [TIMER_LH], eax
1686                                    ;
1687 000009AF C605[448A0100]01          MOV    byte [TIMER_OFL],1
1688
1689                                ;-----      TEST FOR DISKETTE TIME OUT
1690
1691                                T5:
1692                                    ; 23/12/2014
1693 000009B6 EB1D                      jmp    short T6             ; will be replaced with nop, nop
1694                                                                ; (9090h) if a floppy disk
1695                                                                ; is detected.
1696                                    ;mov   al,[CS:MOTOR_COUNT]
1697 000009B8 A0[478A0100]              mov    al, [MOTOR_COUNT]
1698 000009BD FEC8                      dec    al
1699                                    ;mov   [CS:MOTOR_COUNT], al      ; DECREMENT DISKETTE MOTOR CONTROL
1700 000009BF A2[478A0100]              mov    [MOTOR_COUNT], al
1701                                    ;mov   [ORG_MOTOR_COUNT], al
1702 000009C4 750F                      JNZ    short T6             ; RETURN IF COUNT NOT OUT
1703 000009C6 B0F0                      mov    al,0F0h
1704                                    ;AND   [CS:MOTOR_STATUS],al      ; TURN OFF MOTOR RUNNING BITS
1705 000009C8 2005[468A0100]            and    [MOTOR_STATUS], al
1706                                    ;and   [ORG_MOTOR_STATUS], al
1707 000009CE B00C                      MOV    AL,0CH               ; bit 3 = enable IRQ & DMA,
1708                                                                ; bit 2 = enable controller
1709                                                                ;      1 = normal operation
1710                                                                ;      0 = reset
1711                                                                ; bit 0, 1 = drive select
1712                                                                ; bit 4-7 = motor running bits
1713 000009D0 66BAF203                  MOV    DX,03F2H             ; FDC CTL PORT
1714 000009D4 EE                        OUT    DX,AL                ; TURN OFF THE MOTOR
1715                                T6:
1716                                    ;inc   word [CS:wait_count]      ; 22/12/2014 (byte -> word)
1717                                                                ; TIMER TICK INTERRUPT
1718                                    ;;inc  word [wait_count] ;;27/02/2015
1719                                    ;INT   1CH                  ; TRANSFER CONTROL TO A USER ROUTINE
1720                                    ;cli
1721 000009D5 E857040000                call   u_timer                  ; TRANSFER CONTROL TO A USER ROUTINE
1722                                    ; 23/05/2016
1723 000009DA E8CB220100                call   clock                ; Multi Tasking control procedure
1724                                T7:
1725                                    ; 14/10/2015
1726 000009DF B020                      MOV    AL,EOI               ; GET END OF INTERRUPT MASK
1727 000009E1 FA                        CLI                         ; DISABLE INTERRUPTS TILL STACK CLEARED
1728 000009E2 E620                      OUT    INTA00,AL            ; END OF INTERRUPT TO 8259 - 1
1729                                    ;
1730                              rtc_int_2:
1731                                    ; 26/12/2016
1732                                    ;mov   ecx, [cr3reg]
1733                                    ; 13/01/2017
1734 000009E4 803D[D4030300]00          cmp    byte [u.t_lock], 0        ; T_LOCK
1735 000009EB 7730                      ja     short timer_int_return  ; Timer Lock : 'sysrele' is needed !
1736                                    ; 28/02/2017
1737                                    ; We need to exit if the user's IRQ callback service is in progress!
1738                                    ; (To prevent a conflict!)
1739 000009ED 803D[D8030300]00          cmp    byte [u.r_lock], 0  ; R_LOCK, IRQ callback service lock !
1740 000009F4 7727                      ja     short timer_int_return  ; Timer Lock : 'sysrele' is needed !
1741                                    ; 15/01/2017
1742 000009F6 803D[50960100]02          cmp    byte [priority], 2
1743 000009FD 733A                      jnb    short T8  ; current process has a timer event (15/01/2017)
1744                                    ; 22/05/2016
1745 000009FF 803D[51960100]00          cmp    byte [p_change], 0 ; in 'set_run_sequence', in 'rtc_p'
1746 00000A06 7615                      jna    short timer_int_return ; 23/05/2016
1747
1748                                    ; 15/01/2017
1749
1750                                    ; present process must be changed with high priority process
1751                                    ;xor   al, al
1752 00000A08 31C0                      xor    eax, eax ; 26/12/2016
1753 00000A0A A2[51960100]              mov    [p_change], al ; 0
1754                                    ;mov   byte [priority], 2 ; 15/01/2017 (there is a timer event)
1755
1756 00000A0F 803D[5B030300]FF          cmp     byte [sysflg], 0FFh ; user or system space ?
1757 00000A16 7416                      je     short rtc_int_3     ; user space ([sysflg]= 0FFh)
```

```
1758
1759                                    ; system space, wait for 'sysret'
1760                                    ; to change running process
1761                                    ; with high priority (event) process
1762
1763 00000A18 A2[A8030300]                  mov    [u.quant], al ; 0
1764
1765                                  timer_int_return: ; 23/05/2016 - jump from 'rtc_int' ('rtc_int_2')
1766 00000A1D 8B0D[5C040300]                mov    ecx, [cr3reg]       ; previous value/content of cr3 register
1767 00000A23 0F22D9                        mov    cr3, ecx     ; restore cr3 register content
1768                                        ;
1769 00000A26 61                            popad ; edi, esi, ebp, temp (icrement esp by 4), ebx, edx, ecx, eax
1770                                        ;
1771 00000A27 0FA9                          pop    gs
1772 00000A29 0FA1                          pop    fs
1773 00000A2B 07                            pop    es
1774 00000A2C 1F                            pop    ds
1775                                        ;
1776 00000A2D CF                            iretd  ; return from interrupt
1777
1778                                  rtc_int_3:
1779 00000A2E FE05[5B030300]                inc    byte [sysflg]       ; now, we are in system space
1780                                        ;
1781 00000A34 E9E3CE0000                      jmp     sysrelease ; change running process immediatelly
1782
1783                                  T8:
1784                                        ; 13/01/2017 (eax -> ebx)
1785                                        ; callback checking... (19/12/2016)
1786 00000A39 31DB                          xor    ebx, ebx
1787 00000A3B 871D[D0030300]                xchg   ebx, [u.tcb] ; callback address (0 = normal return)
1788 00000A41 09DB                          or     ebx, ebx
1789 00000A43 74D8                          jz     short timer_int_return
1790
1791                                        ; Set user's callback routine as return address from this interrupt
1792                                        ; and set normal return address as return address from callback
1793                                        ; routine!!! (19/12/2016)
1794
1795                                        ; 14/01/2017
1796                                        ; 13/01/2017 - Timer Lock (T_LOCK)
1797 00000A45 FE05[D4030300]                inc    byte [u.t_lock]
1798 00000A4B 8A0D[5B030300]                mov    cl, [sysflg]
1799 00000A51 880D[D5030300]                mov    [u.t_mode], cl
1800
1801 00000A57 8B2D[5C890100]                mov    ebp, [tss.esp0] ; kernel stack address (for ring 0)
1802 00000A5D 83ED14                        sub    ebp, 20             ; eip, cs, eflags, esp, ss
1803 00000A60 892D[5C030300]                mov    [u.sp], ebp
1804 00000A66 8925[60030300]                mov    [u.usp], esp
1805
1806                                        ;or    word [ebp+8], 200h ; 22/01/2017, force enabling interrupts
1807
1808 00000A6C 8B44241C                      mov    eax, [esp+28] ; pushed eax
1809 00000A70 A3[64030300]                  mov    [u.r0], eax
1810
1811 00000A75 E82C0F0100                    call   wswap ; save user's registers & status
1812
1813                                        ; software int is in ring 0 but timer int must return to ring 3
1814                                        ; so, ring 3 return address and stack registers
1815                                        ; (eip, cs, eflags, esp, ss)
1816                                        ; must be copied to timer int return
1817                                        ; eip will be replaced by callback service routine address
1818
1819 00000A7A C605[5B030300]FF              mov    byte [sysflg], 0FFh ; user mode
1820
1821                                        ; system mode (system call)
1822                                        ;mov   ebp, [u.sp] ; EIP (u), CS (UCODE), EFLAGS (u),
1823                                                        ; ESP (u), SS (UDATA)
1824
1825 00000A81 8B4510                      mov    eax, [ebp+16]; SS (UDATA
1826 00000A84 89E6                          mov    esi, esp
1827 00000A86 50                            push   eax
1828 00000A87 50                            push   eax
1829 00000A88 89E7                          mov    edi, esp
1830 00000A8A 893D[60030300]                mov    [u.usp], edi
1831 00000A90 B908000000                    mov    ecx, ((ESPACE/4) - 4) ; except DS, ES, FS, GS
1832 00000A95 F3A5                          rep    movsd
1833 00000A97 B104                          mov    cl, 4
1834 00000A99 F3AB                          rep    stosd
1835 00000A9B 893D[5C030300]                mov    [u.sp], edi
1836 00000AA1 89EE                          mov    esi, ebp
1837 00000AA3 B105                          mov    cl, 5 ; EIP (u), CS (UCODE), EFLAGS (u), ESP (u), SS (UDATA)
1838 00000AA5 F3A5                          rep    movsd
1839
1840 00000AA7 8B0D[B8030300]                mov    ecx, [u.pgdir]
1841 00000AAD 890D[5C040300]                mov    [cr3reg], ecx
1842
1843                                        ; 13/01/207 (eax -> ebx)
1844                                        ; EBX = callback routine address (virtual, not physical address!)
1845
1846                                        ; 09/01/2017
1847                                        ; !!! CALLBACK ROUTINE MUST BE ENDED/RETURNED WITH 'sysrele'
1848                                        ;     system call !!!
1849                                        ; 25/12/2016
1850                                        ; Callback Note: (19/12/2016)
1851                                        ; !!! CALLBACK ROUTINE MUST BE ENDED/RETURNED WITH 'RETN' !!!
1852                                        ;     pushf ; save flags
1853                                        ;     <callback service code>
1854                                        ;     popf  ; restore flags
1855                                        ;     retn ; return to normal running address
1856                                        ;
1857
1858                                        ; 15/01/2017
1859                                        ; 14/01/2017
1860                                        ; 13/01/2017 (eax -> ebx)
1861                                        ; 10/01/2017
1862                                  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
00000AB3 8B2D[5C030300]                 mov   ebp, [u.sp]; kernel's stack, points to EIP (user)
00000AB9 895D00                         mov   [ebp], ebx
00000ABC 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
00000AC1 1E                             push  ds ; ** ; 23/05/2016
00000AC2 50                             push  eax ; *
00000AC3 66B81000                       mov   ax, KDATA
00000AC7 8ED8                           mov   ds, ax
                                        ;
00000AC9 8A25[3E8A0100]                 mov   ah, [RTC_2Hz] ;  2 Hz interrupt to 1 Hz function
00000ACF 80F401                         xor   ah, 1
00000AD2 8825[3E8A0100]                 mov   [RTC_2Hz], ah ; 1 = 0.5 second, 0 = 1 second
00000AD8 753B                           jnz   short rtc_int_return ; half second
                                        ; 1 second
                                    rtc_int_0:
                                        ; 22/05/2016
00000ADA 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
00000ADB 06                             push  es
00000ADC 0FA0                           push  fs
00000ADE 0FA8                           push  gs
00000AE0 60                             pushad  ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi
00000AE1 66B91000                       mov     cx, KDATA
                                          ;mov    ds, cx ; 06/06/2016
```

```
1968 00000AE5 8EC1                      mov     es, cx
1969 00000AE7 8EE1                      mov     fs, cx
1970 00000AE9 8EE9                      mov     gs, cx
1971                                    ;
1972 00000AEB 0F20D9                    mov    ecx, cr3
1973 00000AEE 890D[5C040300]            mov    [cr3reg], ecx ; save current cr3 register value/content
1974                                    ;
1975 00000AF4 803D[D4030300]00          cmp    byte [u.t_lock], 0 ; timer lock (callback) status ?
1976 00000AFB 7711                      ja     short rtc_int_1        ; yes
1977
1978                                    ; 15/01/2017
1979 00000AFD 3B0D[C0890100]            cmp    ecx, [k_page_dir]
1980 00000B03 7409                      je     short rtc_int_1
1981
1982 00000B05 8B0D[C0890100]            mov    ecx, [k_page_dir]
1983 00000B0B 0F22D9                    mov    cr3, ecx
1984                                rtc_int_1:
1985                                    ; Timer event (kernel) functions must be performed with
1986                                    ; 1 second intervals - TRDOS 386 (TRDOS v2.0) feature ! -
1987                                    ;
1988                                    ; 25/08/2014
1989 00000B0E E81A030000                call   rtc_p  ; 19/05/2016 - major modification
1990
1991                                    ; 23/05/2016
1992 00000B13 28E4                      sub    ah, ah ; 0
1993                                    ; 22/05/2016 - TRDOS 386 timer event modifications
1994                                rtc_int_return: ; 19/05/2016
1995                                    ; 22/02/2015 - dsectpm.s
1996                                    ; [ source: http://wiki.osdev.org/RTC ]
1997                                    ; read status register C to complete procedure
1998                                    ;(it is needed to get a next IRQ 8)
1999 00000B15 B00C                      mov    al, 0Ch ;
2000 00000B17 E670                      out    70h, al ; select register C
2001 00000B19 90                        nop
2002 00000B1A E471                      in     al, 71h ; just throw away contents
2003                                    ; 22/02/2015
2004 00000B1C B020                      MOV    AL,EOI       ; END OF INTERRUPT
2005                                    ;CLI                ; DISABLE INTERRUPTS TILL STACK CLEARED
2006 00000B1E E6A0                      OUT    INTB00,AL    ; FOR CONTROLLER #2
2007
2008                                    ; 23/05/2016
2009 00000B20 B020                      MOV    AL,EOI       ; GET END OF INTERRUPT MASK
2010 00000B22 FA                        CLI                 ; DISABLE INTERRUPTS TILL STACK CLEARED
2011 00000B23 E620                      OUT    INTA00,AL    ; END OF INTERRUPT TO 8259 - 1
2012                                    ;
2013                                    ; 23/05/2016
2014 00000B25 20E4                      and    ah, ah
2015 00000B27 0F84B7FEFFFF                jz      rtc_int_2
2016
2017                                    ; ah = 1 (half second)
2018 00000B2D 58                        pop    eax ; *
2019 00000B2E 1F                        pop    ds  ; **
2020 00000B2F CF                        iretd
2021
2022                                ; ////////////////
2023
2024                                    ; 28/08/2014
2025                                irq0:
2026 00000B30 6A00                        push       dword 0
2027 00000B32 EB48                      jmp    short which_irq
2028                                irq1:
2029 00000B34 6A01                        push       dword 1
2030 00000B36 EB44                      jmp    short which_irq
2031                                irq2:
2032 00000B38 6A02                        push       dword 2
2033 00000B3A EB40                      jmp    short which_irq
2034                                irq3:
2035                                    ; 20/11/2015
2036                                    ; 24/10/2015
2037 00000B3C 2EFF15[682F0100]          call   dword [cs:com2_irq3]
2038 00000B43 6A03                      push   dword 3
2039 00000B45 EB35                      jmp    short which_irq
2040                                irq4:
2041                                    ; 20/11/2015
2042                                    ; 24/10/2015
2043 00000B47 2EFF15[642F0100]          call   dword [cs:com1_irq4]
2044 00000B4E 6A04                        push       dword 4
2045 00000B50 EB2A                      jmp    short which_irq
2046                                irq5:
2047 00000B52 6A05                        push       dword 5
2048 00000B54 EB26                      jmp    short which_irq
2049                                irq6:
2050 00000B56 6A06                        push       dword 6
2051 00000B58 EB22                      jmp    short which_irq
2052                                irq7:
2053 00000B5A 6A07                        push       dword 7
2054 00000B5C EB1E                      jmp    short which_irq
2055                                irq8:
2056 00000B5E 6A08                        push       dword 8
2057 00000B60 EB1A                      jmp    short which_irq
2058                                irq9:
2059 00000B62 6A09                        push       dword 9
2060 00000B64 EB16                      jmp    short which_irq
2061                                irq10:
2062 00000B66 6A0A                        push       dword 10
2063 00000B68 EB12                      jmp    short which_irq
2064                                irq11:
2065 00000B6A 6A0B                        push       dword 11
2066 00000B6C EB0E                      jmp    short which_irq
2067                                irq12:
2068 00000B6E 6A0C                        push       dword 12
2069 00000B70 EB0A                      jmp    short which_irq
2070                                irq13:
2071 00000B72 6A0D                        push       dword 13
2072 00000B74 EB06                      jmp    short which_irq
```

```
                                irq14:
00000B76 6A0E                           push    dword 14
00000B78 EB02                         jmp   short which_irq
                                irq15:
00000B7A 6A0F                           push    dword 15
                                      ;jmp   short which_irq

                                      ; 22/01/2017
                                      ; 19/10/2015
                                      ; 29/08/2014
                                      ; 21/08/2014
                                which_irq:
00000B7C 870424                       xchg   eax, [esp]  ; 28/08/2014
00000B7F 53                           push   ebx
00000B80 56                           push   esi
00000B81 57                           push   edi
00000B82 1E                           push   ds
00000B83 06                           push   es
                                      ;
00000B84 88C3                         mov    bl, al
                                      ;
00000B86 B810000000                   mov    eax, KDATA
00000B8B 8ED8                         mov    ds, ax
00000B8D 8EC0                         mov    es, ax
                                      ; 19/10/2015
00000B8F FC                           cld
                                        ; 27/08/2014
00000B90 8105[C6490100]A000-            add     dword [scr_row], 0A0h
00000B98 0000
                                      ;
00000B9A B417                         mov    ah, 17h      ; blue (1) background,
                                                  ; light gray (7) forecolor
00000B9C 8B3D[C6490100]                 mov     edi, [scr_row]
00000BA2 B049                         mov    al, 'I'
00000BA4 66AB                         stosw
00000BA6 B052                         mov    al, 'R'
00000BA8 66AB                         stosw
00000BAA B051                         mov    al, 'Q'
00000BAC 66AB                         stosw
00000BAE B020                         mov    al, ' '
00000BB0 66AB                         stosw
00000BB2 88D8                         mov    al, bl
00000BB4 3C0A                         cmp    al, 10
00000BB6 7208                         jb     short ii1
00000BB8 B031                         mov    al, '1'
00000BBA 66AB                         stosw
00000BBC 88D8                         mov    al, bl
00000BBE 2C0A                         sub    al, 10
                                ii1:
00000BC0 0430                         add    al, '0'
00000BC2 66AB                         stosw
00000BC4 B020                         mov    al, ' '
00000BC6 66AB                         stosw
00000BC8 B021                         mov    al, '!'
00000BCA 66AB                         stosw
00000BCC B020                         mov    al, ' '
00000BCE 66AB                         stosw
                                      ; 23/02/2015
00000BD0 80FB07                       cmp    bl, 7 ; check for IRQ 8 to IRQ 15
00000BD3 7604                         jna    ii2
                                      ; 22/01/2017
00000BD5 B020                         mov    al, 20h  ; END OF INTERRUPT COMMAND TO
00000BD7 E6A0                         out    0A0h, al ; the 2nd 8259
                                ii2:
00000BD9 B020                         mov    al, 20h  ; END OF INTERRUPT COMMAND TO
00000BDB E620                         out    20h, al ; the 2nd 8259
00000BDD 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:
00000BE2 6A00                           push       dword 0
00000BE4 E990000000                     jmp     cpu_except
                                exc1:
00000BE9 6A01                           push       dword 1
00000BEB E989000000                     jmp     cpu_except
                                exc2:
00000BF0 6A02                           push       dword 2
00000BF2 E982000000                     jmp     cpu_except
                                exc3:
00000BF7 6A03                           push       dword 3
00000BF9 EB7E                           jmp     cpu_except
                                exc4:
00000BFB 6A04                           push       dword 4
00000BFD EB7A                           jmp     cpu_except
                                exc5:
00000BFF 6A05                           push       dword 5
00000C01 EB76                           jmp     cpu_except
                                exc6:
00000C03 6A06                           push       dword 6
00000C05 EB72                           jmp     cpu_except
                                exc7:
00000C07 6A07                           push       dword 7
```

```
2177 00000C09 EB6E                      jmp     cpu_except
2178                                exc8:
2179                                    ; [esp] = Error code
2180 00000C0B 6A08                      push    dword 8
2181 00000C0D EB5C                      jmp     cpu_except_en
2182                                exc9:
2183 00000C0F 6A09                      push    dword 9
2184 00000C11 EB66                      jmp     cpu_except
2185                                exc10:
2186                                    ; [esp] = Error code
2187 00000C13 6A0A                      push    dword 10
2188 00000C15 EB54                      jmp     cpu_except_en
2189                                exc11:
2190                                    ; [esp] = Error code
2191 00000C17 6A0B                      push    dword 11
2192 00000C19 EB50                      jmp     cpu_except_en
2193                                exc12:
2194                                    ; [esp] = Error code
2195 00000C1B 6A0C                      push    dword 12
2196 00000C1D EB4C                      jmp     cpu_except_en
2197                                exc13:
2198                                    ; [esp] = Error code
2199 00000C1F 6A0D                      push    dword 13
2200 00000C21 EB48                      jmp     cpu_except_en
2201                                exc14:
2202                                    ; [esp] = Error code
2203 00000C23 6A0E                      push    dword 14
2204 00000C25 EB44                    jmp   short cpu_except_en
2205                                exc15:
2206 00000C27 6A0F                      push    dword 15
2207 00000C29 EB4E                      jmp     cpu_except
2208                                exc16:
2209 00000C2B 6A10                      push    dword 16
2210 00000C2D EB4A                      jmp     cpu_except
2211                                exc17:
2212                                    ; [esp] = Error code
2213 00000C2F 6A11                      push    dword 17
2214 00000C31 EB38                    jmp   short cpu_except_en
2215                                exc18:
2216 00000C33 6A12                      push    dword 18
2217 00000C35 EB42                    jmp   short cpu_except
2218                                exc19:
2219 00000C37 6A13                      push    dword 19
2220 00000C39 EB3E                    jmp   short cpu_except
2221                                exc20:
2222 00000C3B 6A14                      push    dword 20
2223 00000C3D EB3A                    jmp   short cpu_except
2224                                exc21:
2225 00000C3F 6A15                      push    dword 21
2226 00000C41 EB36                    jmp   short cpu_except
2227                                exc22:
2228 00000C43 6A16                      push    dword 22
2229 00000C45 EB32                    jmp   short cpu_except
2230                                exc23:
2231 00000C47 6A17                      push    dword 23
2232 00000C49 EB2E                    jmp   short cpu_except
2233                                exc24:
2234 00000C4B 6A18                      push    dword 24
2235 00000C4D EB2A                    jmp   short cpu_except
2236                                exc25:
2237 00000C4F 6A19                      push    dword 25
2238 00000C51 EB26                    jmp   short cpu_except
2239                                exc26:
2240 00000C53 6A1A                      push    dword 26
2241 00000C55 EB22                    jmp   short cpu_except
2242                                exc27:
2243 00000C57 6A1B                      push    dword 27
2244 00000C59 EB1E                    jmp   short cpu_except
2245                                exc28:
2246 00000C5B 6A1C                      push    dword 28
2247 00000C5D EB1A                    jmp   short cpu_except
2248                                exc29:
2249 00000C5F 6A1D                      push    dword 29
2250 00000C61 EB16                    jmp   short cpu_except
2251                                exc30:
2252 00000C63 6A1E                      push    dword 30
2253 00000C65 EB04                    jmp   short cpu_except_en
2254                                exc31:
2255 00000C67 6A1F                      push    dword 31
2256 00000C69 EB0E                      jmp   short cpu_except
2257
2258                                    ; 19/10/2015
2259                                    ; 19/09/2015
2260                                    ; 01/09/2015
2261                                    ; 28/08/2015
2262                                    ; 28/08/2014
2263                                cpu_except_en:
2264 00000C6B 87442404                  xchg  eax, [esp+4] ; Error code
2265 00000C6F 36A3[78050300]            mov   [ss:error_code], eax
2266 00000C75 58                        pop   eax  ; Exception number
2267 00000C76 870424                    xchg  eax, [esp]
2268                                          ; eax = eax before exception
2269                                          ; [esp] -> exception number
2270                                          ; [esp+4] -> EIP to return
2271                                    ; 22/01/2017
2272                                    ; 19/10/2015
2273                                    ; 19/09/2015
2274                                    ; 01/09/2015
2275                                    ; 28/08/2015
2276                                    ; 29/08/2014
2277                                    ; 28/08/2014
2278                                    ; 25/08/2014
2279                                    ; 21/08/2014
2280                                cpu_except: ; CPU Exceptions
2281 00000C79 FC                        cld
```

```
2282 00000C7A 870424                    xchg   eax, [esp]
2283                                    ; eax = Exception number
2284                                    ; [esp] = eax (before exception)
2285 00000C7D 53                        push   ebx
2286 00000C7E 56                        push   esi
2287 00000C7F 57                        push   edi
2288 00000C80 1E                        push   ds
2289 00000C81 06                        push   es
2290                                    ; 28/08/2015
2291 00000C82 66BB1000                  mov    bx, KDATA
2292 00000C86 8EDB                      mov    ds, bx
2293 00000C88 8EC3                      mov    es, bx
2294 00000C8A 0F20DB                    mov    ebx, cr3
2295 00000C8D 53                        push   ebx ; (*) page directory
2296                                    ; 19/10/2015
2297 00000C8E FC                        cld
2298                                    ; 25/03/2015
2299 00000C8F 8B1D[C0890100]            mov    ebx, [k_page_dir]
2300 00000C95 0F22DB                    mov    cr3, ebx
2301                                    ; 28/08/2015
2302 00000C98 83F80E                    cmp    eax, 0Eh ; 14, PAGE FAULT
2303 00000C9B 750F                      jne    short cpu_except_nfp
2304 00000C9D E809520000                call   page_fault_handler
2305 00000CA2 21C0                      and    eax, eax
2306 00000CA4 0F8401010000                jz   iiretp ; 01/09/2015
2307 00000CAA B00E                      mov    al, 0Eh ; 14
2308                                cpu_except_nfp:
2309                                    ; 23/08/2016
2310 00000CAC 803D[CA6F0000]03          cmp    byte [CRT_MODE], 3
2311 00000CB3 7409                      je     short cpu_except_mode_3
2312 00000CB5 50                        push   eax
2313 00000CB6 B003                      mov    al, 3
2314 00000CB8 E8AB0E0000                call    set_mode
2315 00000CBD 58                        pop    eax
2316                                cpu_except_mode_3:
2317                                    ; 02/04/2015
2318 00000CBE BB[0B090000]              mov    ebx, hang
2319 00000CC3 875C241C                  xchg   ebx, [esp+28]
2320                                    ; EIP (points to instruction which faults)
2321                                    ; New EIP (hang)
2322 00000CC7 891D[7C050300]            mov    [FaultOffset], ebx
2323 00000CCD C744242008000000          mov    dword [esp+32], KCODE ; kernel's code segment
2324 00000CD5 814C242400020000          or     dword [esp+36], 200h ; enable interrupts (set IF)
2325                                    ;
2326 00000CDD 88C4                      mov    ah, al
2327 00000CDF 240F                      and    al, 0Fh
2328 00000CE1 3C09                      cmp    al, 9
2329 00000CE3 7602                      jna    short h1ok
2330 00000CE5 0407                      add    al, 'A'-':'
2331                                h1ok:
2332 00000CE7 C0EC04                    shr    ah, 4
2333 00000CEA 80FC09                    cmp    ah, 9
2334 00000CED 7603                      jna    short h2ok
2335 00000CEF 80C407                    add    ah, 'A'-':'
2336                                h2ok:
2337 00000CF2 86E0                      xchg   ah, al
2338 00000CF4 66053030                  add    ax, '00'
2339 00000CF8 66A3[204C0100]            mov    [excnstr], ax
2340                                    ;
2341                                    ; 29/08/2014
2342 00000CFE A1[7C050300]              mov    eax, [FaultOffset]
2343 00000D03 51                        push   ecx
2344 00000D04 52                        push   edx
2345 00000D05 89E3                      mov    ebx, esp
2346                                    ; 28/08/2015
2347 00000D07 B910000000                mov    ecx, 16        ; divisor value to convert binary number
2348                                                       ; to hexadecimal string
2349                                    ;mov   ecx, 10          ; divisor to convert
2350                                                         ; binary number to decimal string
2351                                b2d1:
2352 00000D0C 31D2                      xor    edx, edx
2353 00000D0E F7F1                      div    ecx
2354 00000D10 6652                      push   dx
2355 00000D12 39C8                      cmp    eax, ecx
2356 00000D14 73F6                      jnb    short b2d1
2357 00000D16 BF[2B4C0100]              mov    edi, EIPstr ; EIP value
2358                                                       ; points to instruction which faults
2359                                    ; 28/08/2015
2360 00000D1B 89C2                      mov    edx, eax
2361                                b2d2:
2362                                    ;add   al, '0'
2363 00000D1D 8A82[1F430000]            mov    al, [edx+hexchrs]
2364 00000D23 AA                        stosb             ; write hexadecimal digit to its place
2365 00000D24 39E3                      cmp    ebx, esp
2366 00000D26 7606                      jna    short b2d3
2367 00000D28 6658                      pop    ax
2368 00000D2A 88C2                      mov    dl, al
2369 00000D2C EBEF                      jmp    short b2d2
2370                                b2d3:
2371 00000D2E B068                      mov    al, 'h' ; 28/08/2015
2372 00000D30 AA                        stosb
2373 00000D31 B020                      mov    al, 20h          ; space
2374 00000D33 AA                        stosb
2375 00000D34 30C0                      xor    al, al     ; to do it an ASCIIZ string
2376 00000D36 AA                        stosb
2377                                    ;
2378 00000D37 5A                        pop    edx
2379 00000D38 59                        pop    ecx
2380                                    ;
2381 00000D39 B44F                      mov    ah, 4Fh      ; red (4) background,
2382                                                  ; white (F) forecolor
2383 00000D3B BE[104C0100]              mov    esi, exc_msg ; message offset
2384                                    ;
2385                                    ; 20/01/2017 (!cpu exception!)
2386                                    ;
```

```
2387 00000D40 8105[C6490100]A000-             add    dword [scr_row], 0A0h
2387 00000D48 0000
2388 00000D4A 8B3D[C6490100]                  mov    edi, [scr_row]
2389                                         ;
2390 00000D50 C605[5B030300]00               mov    byte [sysflg], 0  ; system mode
2391 00000D57 FB                               sti
2392                                         ;
2393 00000D58 E8EDFBFFFF                     call   printk
2394                                         ;
2395 00000D5D B410                           mov    ah, 10h
2396 00000D5F E881010000                     call   int16h ; getc
2397                                         ;
2398 00000D64 B003                           mov    al, 3
2399 00000D66 E8FD0D0000                     call   _set_mode
2400                                         ;
2401 00000D6B B801000000                     mov    eax, 1
2402 00000D70 E90ECD0000                     jmp    sysexit ; terminate process !!!
2403
2404                                         ; 22/01/2017
2405                                         ; 18/04/2016
2406                                         ; 28/08/2015
2407                                         ; 23/02/2015
2408                                         ; 20/08/2014
2409                                  ignore_int:
2410 00000D75 50                             push   eax
2411 00000D76 53                             push   ebx ; 23/02/2015
2412 00000D77 56                             push   esi
2413 00000D78 57                             push   edi
2414 00000D79 1E                             push   ds
2415 00000D7A 06                             push   es
2416                                         ; 18/04/2016
2417 00000D7B 66B81000                       mov    ax, KDATA
2418 00000D7F 8ED8                           mov    ds, ax
2419 00000D81 8EC0                           mov    es, ax
2420                                         ; 28/08/2015
2421 00000D83 0F20D8                         mov    eax, cr3
2422 00000D86 50                             push   eax ; (*) page directory
2423                                         ;
2424 00000D87 B467                           mov    ah, 67h      ; brown (6) background,
2425                                                     ; light gray (7) forecolor
2426 00000D89 BE[D84A0100]                   mov    esi, int_msg ; message offset
2427                                  piemsg:
2428                                           ; 27/08/2014
2429 00000D8E 8105[C6490100]A000-              add     dword [scr_row], 0A0h
2429 00000D96 0000
2430 00000D98 8B3D[C6490100]                    mov     edi, [scr_row]
2431                                           ;
2432 00000D9E E8A7FBFFFF                     call   printk
2433                                         ;
2434                                         ; 23/02/2015
2435 00000DA3 B020                           mov    al, 20h  ; END OF INTERRUPT COMMAND TO
2436 00000DA5 E6A0                           out    0A0h, al ; the 2nd 8259
2437                                         ; 22/08/2014
2438 00000DA7 B020                           mov    al, 20h ; END OF INTERRUPT COMMAND TO 8259
2439 00000DA9 E620                           out    20h, al      ; 8259 PORT
2440                                  iiretp:
2441                                         ; 22/01/2017
2442                                         ; 01/09/2015
2443                                         ; 28/08/2015
2444 00000DAB 58                             pop    eax ; (*) page directory
2445 00000DAC 0F22D8                         mov    cr3, eax
2446                                  iiret:
2447 00000DAF 07                             pop    es
2448 00000DB0 1F                             pop    ds
2449 00000DB1 5F                             pop    edi
2450 00000DB2 5E                             pop    esi
2451 00000DB3 5B                             pop    ebx ; 29/08/2014
2452 00000DB4 58                             pop    eax
2453 00000DB5 CF                             iretd
2454
2455                                         ; 23/05/2016
2456                                         ; 22/08/2014
2457                                         ; IBM PC/AT BIOS source code ----- 10/06/85 (bios.asm)
2458                                         ; (INT 1Ah)
2459                                         ;; Linux (v0.12) source code (main.c) by Linus Torvalds (1991)
2460                                  time_of_day:
2461 00000DB6 E8DF5E0000                     call   UPD_IPR                    ; WAIT TILL UPDATE NOT IN PROGRESS
2462 00000DBB 726F                             jc      short time_of_day_retn ; 23/05/2016
2463 00000DBD B000                           mov    al, CMOS_SECONDS
2464 00000DBF E8F15E0000                     call   CMOS_READ
2465 00000DC4 A2[308A0100]                   mov    [time_seconds], al
2466 00000DC9 B002                           mov    al, CMOS_MINUTES
2467 00000DCB E8E55E0000                     call   CMOS_READ
2468 00000DD0 A2[318A0100]                   mov    [time_minutes], al
2469 00000DD5 B004                           mov    al, CMOS_HOURS
2470 00000DD7 E8D95E0000                     call   CMOS_READ
2471 00000DDC A2[328A0100]                     mov     [time_hours], al
2472 00000DE1 B006                           mov    al, CMOS_DAY_WEEK
2473 00000DE3 E8CD5E0000                     call   CMOS_READ
2474 00000DE8 A2[338A0100]                   mov    [date_wday], al
2475 00000DED B007                           mov    al, CMOS_DAY_MONTH
2476 00000DEF E8C15E0000                     call   CMOS_READ
2477 00000DF4 A2[348A0100]                   mov    [date_day], al
2478 00000DF9 B008                           mov    al, CMOS_MONTH
2479 00000DFB E8B55E0000                     call   CMOS_READ
2480 00000E00 A2[358A0100]                   mov    [date_month], al
2481 00000E05 B009                           mov    al, CMOS_YEAR
2482 00000E07 E8A95E0000                     call   CMOS_READ
2483 00000E0C A2[368A0100]                   mov    [date_year], al
2484 00000E11 B032                           mov    al, CMOS_CENTURY
2485 00000E13 E89D5E0000                     call   CMOS_READ
2486 00000E18 A2[378A0100]                   mov    [date_century], al
2487                                         ;
2488 00000E1D B000                           mov    al, CMOS_SECONDS
2489 00000E1F E8915E0000                     call   CMOS_READ
```

```
2490 00000E24 3A05[308A0100]              cmp   al, [time_seconds]
2491 00000E2A 758A                        jne   short time_of_day
2492
2493                                  time_of_day_retn:
2494 00000E2C C3                          retn
2495
2496                                      ; 15/01/2017
2497                                      ; 10/06/2016
2498                                      ; 07/06/2016
2499                                      ; 06/06/2016
2500                                      ; 23/05/2016
2501                                  rtc_p:
2502 00000E2D B101                        mov   cl, 1 ; 15/01/2017
2503 00000E2F EB02                        jmp   short rtc_p0
2504                                  u_timer:
2505                                      ; Timer Events with 18.2 Hz Timer Ticks
2506                                      ; (and also timer events with RTC seconds)
2507 00000E31 28C9                        sub   cl, cl ; mov cl, 0 ; 15/01/2017
2508                                  rtc_p0:
2509                                      ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
2510                                      ; Major Modification:
2511                                      ; Check and Perform Timer Events (for RTC)
2512                                      ; 25/08/2014 - 07/09/2014
2513                                      ; Retro UNIX 386 v1:
2514                                      ; Print Real Time Clock content
2515
2516                                      ; 15/01/2017
2517 00000E33 880D[50960100]              mov   byte [priority], cl ; 0 or 1 (not 2)
2518 00000E39 8A2D[53960100]              mov   ch, [timer_events]
2519 00000E3F 20ED                        and   ch, ch
2520 00000E41 7420                        jz    short rtc_p3
2521
2522 00000E43 BE[60040300]                mov   esi, timer_set  ; beginning address of
2523                                                             ; timer events space
2524                                  rtc_p1:
2525 00000E48 8B06                        mov   eax, [esi]
2526 00000E4A 20C0                        and   al, al ; 0 = free, >0 = process no.
2527 00000E4C 7416                        jz    short rtc_p4
2528                                      ;
2529 00000E4E C1C810                      ror   eax, 16
2530                                      ; ah = response value, al = interrupt type
2531                                      ; 15/01/2017
2532                                      ; cl = interrupt source
2533                                      ;       1 = RTC, 0 = PIT
2534 00000E51 38C8                        cmp   al, cl
2535 00000E53 750A                        jne   short rtc_p2 ; not as requested or undefined !
2536 00000E55 3C01                        cmp   al, 1 ; 1 ; RTC interrupt ?
2537 00000E57 7410                        je    short rtc_p5 ; yes, check for response
2538                                      ; 06/06/2016 - 18.2 Hz Timer Ticks
2539 00000E59 836E080A                    sub   dword [esi+8], 10 ; 1 tick = 10
2540 00000E5D 7613                        jna   short rtc_p6  ; continue for responding
2541                                  rtc_p2:
2542                                      ; 15/01/2017 (cl -> ch)
2543                                      ; 07/06/2016
2544 00000E5F FECD                        dec   ch    ; remain count of timer events
2545 00000E61 7501                        jnz   short rtc_p4
2546                                  rtc_p3:
2547 00000E63 C3                          retn
2548                                  rtc_p4:
2549                                      ;cmp   esi, timer_set + 240 ; 15*16 (last event)
2550                                      ;jnb   short rtc_p3 ; end of timer event space
2551 00000E64 83C610                      add   esi, 16 ; next timer event
2552 00000E67 EBDF                        jmp   short rtc_p1
2553                                  rtc_p5:
2554                                      ; current timer count ; 06/06/2016 (182)
2555 00000E69 816E08B6000000              sub   dword [esi+8], 182 ; 1 second (10*18.2)
2556 00000E70 77ED                        ja    short rtc_p2  ; check for the next
2557                                  rtc_p6:
2558                                      ; it is the time of response!
2559 00000E72 8B5E04                      mov   ebx, [esi+4] ; set (count limit) value
2560 00000E75 895E08                      mov   [esi+8], ebx ; reset count down value
2561                                                         ; to count limit
2562                                      ; 19/12/2016
2563                                      ; 10/12/2016 - timer callback modification
2564 00000E78 8B7E0C                      mov   edi, [esi+12] ; response (or callback) address
2565 00000E7B 807E0100                    cmp   byte [esi+1], 0 ; >0 = callback
2566 00000E7F 762A                        jna   short rtc_p8
2567
2568                                      ; timer callback !
2569 00000E81 0FB61E                      movzx  ebx, byte [esi] ; process number (>0)
2570 00000E84 89D8                        mov   eax, ebx
2571 00000E86 C0E302                      shl   bl, 2 ; *4
2572 00000E89 89BB[0C010300]              mov   [ebx+p.tcb-4], edi ; user's callback service addr
2573 00000E8F 3A05[B3030300]              cmp   al, [u.uno]
2574 00000E95 7521                        jne   short rtc_p9
2575 00000E97 893D[D0030300]              mov   [u.tcb], edi
2576                                  rtc_p7:
2577                                      ; 15/01/2017
2578 00000E9D B002                        mov   al, 2
2579 00000E9F A2[50960100]                mov   [priority], al ; 2
2580                                      ; 10/01/2017
2581                                      ;mov  byte [u.pri], 2
2582 00000EA4 A2[A9030300]                mov   [u.pri], al ; 2
2583 00000EA9 EBB4                        jmp   short rtc_p2
2584                                  rtc_p8:
2585                                      ; response address is physical address of
2586                                      ; the program's response (signal return) byte
2587                                      ; 06/06/2016
2588                                      ;mov  edi, [esi+12] ; response address
2589 00000EAB 8827                        mov   [edi], ah     ; response value
2590                                      ;
2591 00000EAD C1C010                      rol   eax, 16
2592                                      ; 15/01/2017
2593 00000EB0 3A05[B3030300]              cmp   al, [u.uno] ; running process ?
2594 00000EB6 74E5                        je    short rtc_p7
```

```
                                rtc_p9:
                                      ; al = process number  ; 10/06/2016
00000EB8 B202                         mov   dl, 2 ; priority, 2 = event (high)
00000EBA E89F1D0100                   call  set_run_sequence ; 19/05/2016
00000EBF 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:
00000EC1 6650                         push  ax
00000EC3 B00B                         mov   al, 0Bh  ; In-Service register
00000EC5 E620                         out   20h, al
00000EC7 EB00                           jmp short $+2
00000EC9 EB00                         jmp short $+2
00000ECB E420                         in    al, 20h
00000ECD 2480                         and   al, 80h ; bit 7 (is it real IRQ 7 or fake?)
00000ECF 7404                           jz      short irq7_iret ; Fake (spurious) IRQ, do not send EOI
00000ED1 B020                           mov     al, 20h ; EOI
00000ED3 E620                         out   20h, al
                                irq7_iret:
00000ED5 6658                         pop   ax
00000ED7 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)
                                      ;
00000ED8 D410                         db 0D4h,10h                   ; Undocumented inst. AAM
                                                                    ; AH = AL / 10h
                                                                    ; AL = AL MOD 10h
00000EDA 660D3030                   or ax,'00'                      ; Make it ASCII based

00000EDE 86E0                         xchg ah, al

00000EE0 C3                           retn

                                ; 15/12/2020
                                real_mem_16m_64k:
00000EE1 0000                         dw 0   ; Real size of system memory (if > 16MB)
                                             ; as number of 64K blocks - 256
                                             ; (This is for saving real system memory
                                             ; because if system memory is larger than
                                             ; 3 GB and if a VESA VBE video bios
                                             ; is detected, 'mem_16m_64K' may be
                                             ; decreased to reserve LFB space
                                             ; at the end of system memory.)
                                             ; Upper memory space from LFB base address
                                             ; to 4GB will not be included by M.A.T.
                                def_LFB_addr:
00000EE3 0000                         dw 0   ; HW of default LFB addr (for mode 118h)


                                %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:
00000EE5 9C                   <1>       pushfd ; 28/08/2014
00000EE6 0E                   <1>       push   cs
00000EE7 E801000000           <1>       call   KEYBOARD_IO_1 ; getc_int
00000EEC 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
00000EED 80642408BE     <1>         and     byte [esp+8], 10111110b ; clear zero flag and cary flag
                        <1>       ;
00000EF2 1E             <1>       push   ds               ; SAVE CURRENT DS
00000EF3 53             <1>       push   ebx              ; SAVE BX TEMPORARILY
                        <1>       ;push  ecx              ; SAVE CX TEMPORARILY
00000EF4 66BB1000       <1>         mov     bx, KDATA
00000EF8 8EDB           <1>       mov    ds, bx           ; PUT SEGMENT VALUE OF DATA AREA INTO DS
                        <1>
```

```
                        <1>       ; 14/01/2017
00000EFA 8B1C24         <1>       mov   ebx, [esp]
                        <1>       ;; 15/01/2017
                        <1>       ; 02/01/2017
                        <1>       ;;mov  byte [intflg], 32h  ; keyboard interrupt
00000EFD FB             <1>       sti
                        <1>       ;
                        <1>
00000EFE 08E4           <1>       or     ah, ah             ; CHECK FOR (AH)= 00H
00000F00 743A           <1>       jz     short _K1          ; ASCII_READ
00000F02 FECC           <1>       dec    ah                    ; CHECK FOR (AH)= 01H
00000F04 7453           <1>         jz      short _K2             ; ASCII_STATUS
00000F06 FECC           <1>       dec    ah                 ; CHECK FOR (AH)= 02H
00000F08 0F8494000000   <1>         jz      _K3                   ; SHIFT STATUS
00000F0E FECC           <1>       dec    ah                 ; CHECK FOR (AH)= 03H
00000F10 0F8493000000   <1>         jz      _K300                 ; SET TYPAMATIC RATE/DELAY
00000F16 80EC02         <1>       sub    ah, 2              ; CHECK FOR (AH)= 05H
00000F19 0F84BC000000   <1>         jz      _K500                 ; KEYBOARD WRITE
                        <1> _KIO1:
00000F1F 80EC0B         <1>       sub    ah, 11             ; AH =  10H
00000F22 740C           <1>       jz     short _K1E         ; EXTENDED ASCII READ
00000F24 FECC           <1>       dec    ah                 ; CHECK FOR (AH)= 11H
00000F26 7422           <1>       jz     short _K2E         ; EXTENDED_ASCII_STATUS
00000F28 FECC           <1>       dec    ah                 ; CHECK FOR (AH)= 12H
00000F2A 7458           <1>       jz     short _K3E         ; EXTENDED_SHIFT_STATUS
                        <1> _KIO_EXIT:
                        <1>       ; 02/01/2017
00000F2C FA             <1>       cli
                        <1>       ;;mov  byte [intflg], 0 ;; 15/01/2017
                        <1>       ;
                        <1>       ;pop   ecx                ; RECOVER REGISTER
00000F2D 5B             <1>       pop    ebx                ; RECOVER REGISTER
00000F2E 1F             <1>       pop    ds                 ; RECOVER SEGMENT
00000F2F CF             <1>       iretd                     ; INVALID COMMAND, EXIT
                        <1>
                        <1>       ;----- ASCII CHARACTER
                        <1> _K1E:
00000F30 E8D3000000     <1>       call   _K1S               ; GET A CHARACTER FROM THE BUFFER (EXTENDED)
00000F35 E848010000     <1>       call   _KIO_E_XLAT        ; ROUTINE TO XLATE FOR EXTENDED CALLS
00000F3A EBF0           <1>       jmp    short _KIO_EXIT        ; GIVE IT TO THE CALLER
                        <1> _K1:
00000F3C E8C7000000     <1>       call   _K1S               ; GET A CHARACTER FROM THE BUFFER
00000F41 E847010000     <1>       call   _KIO_S_XLAT        ; ROUTINE TO XLATE FOR STANDARD CALLS
00000F46 72F4           <1>       jc     short _K1          ; CARRY SET MEANS TROW CODE AWAY
                        <1> _K1A:
00000F48 EBE2           <1>       jmp    short _KIO_EXIT         ; RETURN TO CALLER
                        <1>
                        <1>       ;----- ASCII STATUS
                        <1> _K2E:
00000F4A E804010000     <1>       call   _K2S               ; TEST FOR CHARACTER IN BUFFER (EXTENDED)
00000F4F 7420           <1>       jz     short _K2B         ; RETURN IF BUFFER EMPTY
00000F51 9C             <1>       pushf                     ; SAVE ZF FROM TEST
00000F52 E82B010000     <1>       call   _KIO_E_XLAT        ; ROUTINE TO XLATE FOR EXTENDED CALLS
00000F57 EB17           <1>       jmp    short _K2A          ; GIVE IT TO THE CALLER
                        <1> _K2:
00000F59 E8F5000000     <1>       call   _K2S               ; TEST FOR CHARACTER IN BUFFER
00000F5E 7411           <1>       jz     short _K2B         ; RETURN IF BUFFER EMPTY
00000F60 9C             <1>       pushf                     ; SAVE ZF FROM TEST
00000F61 E827010000     <1>       call   _KIO_S_XLAT        ; ROUTINE TO XLATE FOR STANDARD CALLS
00000F66 7308           <1>       jnc    short _K2A          ; CARRY CLEAR MEANS PASS VALID CODE
00000F68 9D             <1>       popf                      ; INVALID CODE FOR THIS TYPE OF CALL
00000F69 E89A000000     <1>       call   _K1S               ; THROW THE CHARACTER AWAY
00000F6E EBE9           <1>       jmp    short _K2          ; GO LOOK FOR NEXT CHAR, IF ANY
                        <1> _K2A:
00000F70 9D             <1>       popf                      ; RESTORE ZF FROM TEST
                        <1> _K2B:
                        <1>       ; 02/01/2017
00000F71 FA             <1>       cli
                        <1>       ;; mov byte [intflg], 0 ;; 15/01/2017
                        <1>       ;
                        <1>       ;pop   ecx                ; RECOVER REGISTER
00000F72 5B             <1>       pop    ebx                ; RECOVER REGISTER
00000F73 1F             <1>       pop    ds                 ; RECOVER SEGMENT
                        <1>       ; (*) 29/05/2016
                        <1>       ; (*) retf 4              ; THROW AWAY (e)FLAGS
00000F74 7208           <1>       jc     short _k2d
00000F76 7505           <1>       jnz    short _k2c
00000F78 804C240840     <1>       or     byte [esp+8], 01000000b   ; set zero flag bit of eflags register
                        <1> _k2c:
00000F7D 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
00000F7E 804C240801     <1>       or     byte [esp+8], 1  ; set carry bit of eflags register
                        <1>       ; [esp+12] = ESP (user)
                        <1>       ; [esp+16] = SS (User)
00000F83 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
00000F84 8A25[966F0000] <1>       mov   ah, [KB_FLAG_1]           ; GET SYSTEM SHIFT KEY STATUS
00000F8A 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
00000F8D C0E405         <1>         shl ah, 5
00000F90 A0[966F0000]   <1>       mov   al, [KB_FLAG_1]           ; GET SYSTEM SHIFT STATES BACK
00000F95 2473           <1>       and   al, 01110011b       ; ELIMINATE SYS SHIFT, HOLD_STATE AND INS_SHIFT
00000F97 08C4           <1>       or    ah, al                 ; MERGE REMAINING BITS INTO AH
00000F99 A0[986F0000]   <1>       mov   al, [KB_FLAG_3]           ; GET RIGHT CTL AND ALT
00000F9E 240C           <1>       and   al, 00001100b       ; ELIMINATE LC_E0 AND LC_E1
00000FA0 08C4           <1>       or    ah, al              ; OR THE SHIFT FLAGS TOGETHER
                        <1> _K3:
00000FA2 A0[956F0000]   <1>       mov   al, [KB_FLAG]       ; GET THE SHIFT STATUS FLAGS
                        <1>       ;jmp  short _KIO_EXIT           ; RETURN TO CALLER
00000FA7 EB83           <1>       jmp   _KIO_EXIT
                        <1>
                        <1>       ;----- SET TYPAMATIC RATE AND DELAY
                        <1> _K300:
00000FA9 3C05           <1>       cmp   al, 5               ; CORRECT FUNCTION CALL?
                        <1>       ;jne  short _KIO_EXIT           ; NO, RETURN
00000FAB 0F857BFFFFFF   <1>       jne   _KIO_EXIT
00000FB1 F6C3E0         <1>       test  bl, 0E0h            ; TEST FOR OUT-OF-RANGE RATE
00000FB4 0F8572FFFFFF   <1>         jnz   _KIO_EXIT                 ; RETURN IF SO
00000FBA F6C7FC         <1>       test  BH, 0FCh            ; TEST FOR OUT-OF-RANGE DELAY
00000FBD 0F8569FFFFFF   <1>         jnz   _KIO_EXIT                 ; RETURN IF SO
00000FC3 B0F3           <1>       mov   al, KB_TYPA_RD            ; COMMAND FOR TYPAMATIC RATE/DELAY
00000FC5 E8DA060000     <1>       call  SND_DATA            ; SEND TO KEYBOARD
                        <1>       ;mov  cx, 5               ; SHIFT COUNT
                        <1>       ;shl  bh, cl              ; SHIFT DELAY OVER
00000FCA C0E705         <1>       shl   bh, 5
00000FCD 88D8           <1>       mov   al, bl              ; PUT IN RATE
00000FCF 08F8           <1>       or    al, bh              ; AND DELAY
00000FD1 E8CE060000     <1>       call  SND_DATA            ; SEND TO KEYBOARD
00000FD6 E951FFFFFF     <1>         jmp   _KIO_EXIT                 ; RETURN TO CALLER
                        <1>
                        <1>       ;----- WRITE TO KEYBOARD BUFFER
                        <1> _K500:
00000FDB 56             <1>       push  esi                 ; SAVE SI (esi)
00000FDC FA             <1>       cli                       ;
00000FDD 8B1D[A66F0000] <1>       mov   ebx, [BUFFER_TAIL]  ; GET THE 'IN TO' POINTER TO THE BUFFER
00000FE3 89DE           <1>       mov   esi, ebx            ; SAVE A COPY IN CASE BUFFER NOT FULL
00000FE5 E8D3000000     <1>       call  _K4                 ; BUMP THE POINTER TO SEE IF BUFFER IS FULL
00000FEA 3B1D[A26F0000] <1>       cmp   ebx, [BUFFER_HEAD]  ; WILL THE BUFFER OVERRUN IF WE STORE THIS?
00000FF0 740D           <1>       je    short _K502         ; YES - INFORM CALLER OF ERROR
00000FF2 66890E         <1>       mov   [esi], cx           ; NO - PUT ASCII/SCAN CODE INTO BUFFER
00000FF5 891D[A66F0000] <1>       mov   [BUFFER_TAIL], ebx  ; ADJUST 'IN TO' POINTER TO REFLECT CHANGE
00000FFB 28C0           <1>       sub   al, al              ; TELL CALLER THAT OPERATION WAS SUCCESSFUL
00000FFD EB02           <1>       jmp   short _K504         ; SUB INSTRUCTION ALSO RESETS CARRY FLAG
                        <1> _K502:
00000FFF B001           <1>       mov   al, 01h                   ; BUFFER FULL INDICATION
                        <1> _K504:
00001001 FB             <1>       sti
00001002 5E             <1>       pop   esi                 ; RECOVER SI (esi)
00001003 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:
00001008 FA             <1>       cli   ; 03/12/2014
00001009 8B1D[A26F0000] <1>         mov   ebx, [BUFFER_HEAD]    ; GET POINTER TO HEAD OF BUFFER
0000100F 3B1D[A66F0000] <1>         cmp   ebx, [BUFFER_TAIL]    ; TEST END OF BUFFER
                        <1>       ;jne  short _K1U          ; IF ANYTHING IN BUFFER SKIP INTERRUPT
00001015 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
00001017 FB             <1>       sti                       ; INTERRUPTS BACK ON DURING LOOP
00001018 90             <1>       nop                       ; ALLOW AN INTERRUPT TO OCCUR
                        <1> _K1U:
00001019 FA             <1>       cli                       ; INTERRUPTS BACK OFF
0000101A 8B1D[A26F0000] <1>         mov       ebx, [BUFFER_HEAD]  ; GET POINTER TO HEAD OF BUFFER
00001020 3B1D[A66F0000] <1>         cmp     ebx, [BUFFER_TAIL]    ; TEST END OF BUFFER
                        <1> _k1x:
00001026 53             <1>       push  ebx                 ; SAVE ADDRESS
00001027 9C             <1>       pushf                     ; SAVE FLAGS
00001028 E82F070000     <1>       call  MAKE_LED            ; GO GET MODE INDICATOR DATA BYTE
0000102D 8A1D[976F0000] <1>       mov   bl, [KB_FLAG_2]     ; GET PREVIOUS BITS
00001033 30C3           <1>       xor   bl, al              ; SEE IF ANY DIFFERENT
00001035 80E307         <1>       and   bl, 07h      ; KB_LEDS    ; ISOLATE INDICATOR BITS
00001038 7406           <1>       jz    short _K1V          ; IF NO CHANGE BYPASS UPDATE
0000103A E8C9060000     <1>       call  SND_LED1
0000103F FA             <1>       cli                       ; DISABLE INTERRUPTS
                        <1> _K1V:
00001040 9D             <1>       popf                      ; RESTORE FLAGS
00001041 5B             <1>       pop   ebx                 ; RESTORE ADDRESS
00001042 74D3           <1>         je      short _K1T              ; LOOP UNTIL SOMETHING IN BUFFER
                        <1>       ;
00001044 668B03         <1>       mov   ax, [ebx]           ; GET SCAN CODE AND ASCII CODE
00001047 E871000000     <1>         call    _K4                     ; MOVE POINTER TO NEXT POSITION
0000104C 891D[A26F0000] <1>         mov     [BUFFER_HEAD], ebx      ; STORE VALUE IN VARIABLE
00001052 C3             <1>       retn                      ; RETURN
                        <1>
                        <1>       ;----- READ THE KEY TO SEE IF ONE IS PRESENT -----
                        <1> _K2S:
00001053 FA             <1>       cli                       ; INTERRUPTS OFF
```

```
363 00001054 8B1D[A26F0000]      <1>         mov     ebx, [BUFFER_HEAD]      ; GET HEAD POINTER
364 0000105A 3B1D[A66F0000]      <1>         cmp     ebx, [BUFFER_TAIL]      ; IF EQUAL (Z=1) THEN NOTHING THERE
365 00001060 668B03              <1>       mov   ax, [ebx]
366 00001063 9C                  <1>       pushf                     ; SAVE FLAGS
367 00001064 6650                <1>       push  ax                  ; SAVE CODE
368 00001066 E8F1060000          <1>       call  MAKE_LED            ; GO GET MODE INDICATOR DATA BYTE
369 0000106B 8A1D[976F0000]      <1>       mov   bl, [KB_FLAG_2]     ; GET PREVIOUS BITS
370 00001071 30C3                <1>       xor   bl, al              ; SEE IF ANY DIFFERENT
371 00001073 80E307              <1>       and   bl, 07h ; KB_LEDS   ; ISOLATE INDICATOR BITS
372 00001076 7405                <1>       jz    short _K2T          ; IF NO CHANGE BYPASS UPDATE
373 00001078 E874060000          <1>       call  SND_LED                   ; GO TURN ON MODE INDICATORS
374                              <1> _K2T:
375 0000107D 6658                <1>       pop   ax                  ; RESTORE CODE
376 0000107F 9D                  <1>       popf                      ; RESTORE FLAGS
377 00001080 FB                  <1>       sti                       ; INTERRUPTS BACK ON
378 00001081 C3                  <1>       retn                      ; RETURN
379                              <1>
380                              <1>       ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR EXTENDED CALLS -----
381                              <1> _KIO_E_XLAT:
382 00001082 3CF0                <1>       cmp   al, 0F0h            ; IS IT ONE OF THE FILL-INs?
383 00001084 7506                <1>       jne   short _KIO_E_RET    ; NO, PASS IT ON
384 00001086 08E4                <1>         or  ah, ah              ; AH = 0 IS SPECIAL CASE
385 00001088 7402                <1>         jz  short _KIO_E_RET        ; PASS THIS ON UNCHANGED
386 0000108A 30C0                <1>       xor   al, al              ; OTHERWISE SET AL = 0
387                              <1> _KIO_E_RET:
388 0000108C C3                  <1>       retn                      ; GO BACK
389                              <1>
390                              <1>       ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR STANDARD CALLS -----
391                              <1> _KIO_S_XLAT:
392 0000108D 80FCE0              <1>       cmp   ah, 0E0h            ; IS IT KEYPAD ENTER OR / ?
393 00001090 750F                <1>       jne   short _KIO_S2       ; NO, CONTINUE
394 00001092 3C0D                <1>       cmp   al, 0Dh                   ; KEYPAD ENTER CODE?
395 00001094 7408                <1>         je  short _KIO_S1       ; YES, MASSAGE A BIT
396 00001096 3C0A                <1>       cmp   al, 0Ah                   ; CTRL KEYPAD ENTER CODE?
397 00001098 7404                <1>         je  short _KIO_S1       ; YES, MASSAGE THE SAME
398 0000109A B435                <1>       mov   ah, 35h                   ; NO, MUST BE KEYPAD /
399                              <1> _kio_ret: ; 03/12/2014
400 0000109C F8                  <1>       clc
401 0000109D C3                  <1>       retn
402                              <1>       ;jmp  short _KIO_USE            ; GIVE TO CALLER
403                              <1> _KIO_S1:
404 0000109E B41C                <1>       mov   ah, 1Ch                   ; CONVERT TO COMPATIBLE OUTPUT
405                              <1>       ;jmp  short _KIO_USE            ; GIVE TO CALLER
406 000010A0 C3                  <1>       retn
407                              <1> _KIO_S2:
408 000010A1 80FC84              <1>       cmp   ah, 84h                   ; IS IT ONE OF EXTENDED ONES?
409 000010A4 7715                <1>       ja    short _KIO_DIS            ; YES, THROW AWAY AND GET ANOTHER CHAR
410 000010A6 3CF0                <1>       cmp   al, 0F0h            ; IS IT ONE OF THE FILL-INs?
411 000010A8 7506                <1>         jne short _KIO_S3       ; NO, TRY LAST TEST
412 000010AA 08E4                <1>       or    ah, ah              ; AH = 0 IS SPECIAL CASE
413 000010AC 740C                <1>         jz  short _KIO_USE            ; PASS THIS ON UNCHANGED
414 000010AE EB0B                <1>       jmp   short _KIO_DIS            ; THROW AWAY THE REST
415                              <1> _KIO_S3:
416 000010B0 3CE0                <1>       cmp   al, 0E0h            ; IS IT AN EXTENSION OF A PREVIOUS ONE?
417                              <1>       ;jne  short _KIO_USE            ; NO, MUST BE A STANDARD CODE
418 000010B2 75E8                <1>       jne   short _kio_ret
419 000010B4 08E4                <1>       or    ah, ah              ; AH = 0 IS SPECIAL CASE
420 000010B6 7402                <1>         jz  short _KIO_USE            ; JUMP IF AH = 0
421 000010B8 30C0                <1>       xor   al, al              ; CONVERT TO COMPATIBLE OUTPUT
422                              <1>       ;jmp  short _KIO_USE            ; PASS IT ON TO CALLER
423                              <1> _KIO_USE:
424                              <1>       ;clc                      ; CLEAR CARRY TO INDICATE GOOD CODE
425 000010BA C3                  <1>       retn                      ; RETURN
426                              <1> _KIO_DIS:
427 000010BB F9                  <1>       stc                       ; SET CARRY TO INDICATE DISCARD CODE
428 000010BC C3                  <1>       retn                      ; RETURN
429                              <1>
430                              <1>       ;----- INCREMENT BUFFER POINTER ROUTINE -----
431                              <1> _K4:
432 000010BD 43                  <1>       inc     ebx
433 000010BE 43                  <1>       inc   ebx                 ; MOVE TO NEXT WORD IN LIST
434 000010BF 3B1D[9E6F0000]      <1>         cmp     ebx, [BUFFER_END]       ; AT END OF BUFFER?
435                              <1>         ;jne    short _K5               ; NO, CONTINUE
436 000010C5 7206                <1>       jb    short _K5
437 000010C7 8B1D[9A6F0000]      <1>         mov     ebx, [BUFFER_START]     ; YES, RESET TO BUFFER BEGINNING
438                              <1> _K5:
439 000010CD C3                  <1>       retn
440                              <1>
441                              <1> ; 20/02/2015
442                              <1> ; 05/12/2014
443                              <1> ; 26/08/2014
444                              <1> ; KEYBOARD (HARDWARE) INTERRUPT -  IRQ LEVEL 1
445                              <1> ; (INT_09h - Retro UNIX 8086 v1 - U9.ASM, 07/03/2014)
446                              <1> ;
447                              <1> ; Derived from "KB_INT_1" procedure of IBM "pc-at"
448                              <1> ; rombios source code (06/10/1985)
449                              <1> ; 'keybd.asm', HARDWARE INT 09h - (IRQ Level 1)
450                              <1>
451                              <1> ; EQUATES (IBM PC-XT-286 BIOS, 1986, 'POSQEQU.INC')
452                              <1>
453                              <1> ;--------- 8042 COMMANDS --------------------------------------------------------
454                              <1> ENA_KBD             equ   0AEh          ; ENABLE KEYBOARD COMMAND
455                              <1> DIS_KBD             equ   0ADh          ; DISABLE KEYBOARD COMMAND
456                              <1> SHUT_CMD    equ   0FEh          ; CAUSE A SHUTDOWN COMMAND
457                              <1> ;--------- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS ------------
458                              <1> STATUS_PORT  equ   064h         ; 8042 STATUS PORT
459                              <1> INPT_BUF_FULL       equ   00000010b     ; 1 = +INPUT BUFFER FULL
460                              <1> PORT_A              equ   060h          ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
461                              <1> ;---------- 8042 KEYBOARD RESPONSE ---------------------------------------------
462                              <1> KB_ACK              equ   0FAh          ; ACKNOWLEDGE PROM TRANSMISSION
463                              <1> KB_RESEND   equ   0FEh          ; RESEND REQUEST
464                              <1> KB_OVER_RUN  equ   0FFh         ; OVER RUN SCAN CODE
465                              <1> ;---------- KEYBOARD/LED COMMANDS -----------------------------------------------
466                              <1> KB_ENABLE   equ   0F4h          ; KEYBOARD ENABLE
467                              <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:
000010CE FB             <1>      sti                         ; ENABLE INTERRUPTS
                        <1>      ;push  ebp
000010CF 50             <1>      push   eax
000010D0 53             <1>      push   ebx
000010D1 51             <1>      push   ecx
000010D2 52             <1>      push   edx
000010D3 56             <1>      push   esi
000010D4 57             <1>      push   edi
000010D5 1E             <1>      push   ds
000010D6 06             <1>      push   es
000010D7 FC             <1>      cld                         ; FORWARD DIRECTION
000010D8 66B81000       <1>      mov    ax, KDATA
000010DC 8ED8           <1>      mov    ds, ax
000010DE 8EC0           <1>      mov    es, ax
                        <1>      ;
                        <1>      ;-----WAIT FOR KEYBOARD DISABLE COMMAND TO BE ACCEPTED
000010E0 B0AD           <1>      mov    al, DIS_KBD         ; DISABLE THE KEYBOARD COMMAND
000010E2 E8A9050000     <1>      call   SHIP_IT                  ; EXECUTE DISABLE
000010E7 FA             <1>      cli                         ; DISABLE INTERRUPTS
000010E8 B900000100     <1>      mov    ecx, 10000h         ; SET MAXIMUM TIMEOUT
                        <1> KB_INT_01:
000010ED E464           <1>      in     al, STATUS_PORT           ; READ ADAPTER STATUS
000010EF A802           <1>      test   al, INPT_BUF_FULL   ; CHECK INPUT BUFFER FULL STATUS BIT
000010F1 E0FA           <1>      loopnz KB_INT_01           ; WAIT FOR COMMAND TO BE ACCEPTED
                        <1>      ;
```

```
                        <1>       ;----- READ CHARACTER FROM KEYBOARD INTERFACE
000010F3 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
000010F5 FB             <1>       sti                        ; ENABLE INTERRUPTS AGAIN
000010F6 3CFE           <1>       cmp    al, KB_RESEND       ; IS THE INPUT A RESEND
000010F8 7411           <1>         je     short KB_INT_4          ; GO IF RESEND
                        <1>       ;
                        <1>       ;----- CHECK FOR RESPONSE TO A COMMAND TO KEYBOARD
000010FA 3CFA           <1>       cmp    al, KB_ACK          ; IS THE INPUT AN ACKNOWLEDGE
000010FC 751A           <1>         jne    short KB_INT_2          ; GO IF NOT
                        <1>       ;
                        <1>       ;----- A COMMAND TO THE KEYBOARD WAS ISSUED
000010FE FA             <1>       cli                        ; DISABLE INTERRUPTS
000010FF 800D[976F0000]10 <1>     or     byte [KB_FLAG_2], KB_FA ; INDICATE ACK RECEIVED
00001106 E97A020000     <1>         jmp    K26                     ; RETURN IF NOT (ACK RETURNED FOR DATA)
                        <1>       ;
                        <1>       ;----- RESEND THE LAST BYTE
                        <1> KB_INT_4:
0000110B FA             <1>       cli                        ; DISABLE INTERRUPTS
0000110C 800D[976F0000]20 <1>     or     byte [KB_FLAG_2], KB_FE ; INDICATE RESEND RECEIVED
00001113 E96D020000     <1>         jmp    K26                     ; RETURN IF NOT ACK RETURNED FOR DATA)
                        <1>       ;
                        <1> ;-----      UPDATE MODE INDICATORS IF CHANGE IN STATE
                        <1> KB_INT_2:
00001118 6650           <1>       push   ax                  ; SAVE DATA IN
0000111A E83D060000     <1>       call   MAKE_LED            ; GO GET MODE INDICATOR DATA BYTE
0000111F 8A1D[976F0000] <1>       mov    bl, [KB_FLAG_2]     ; GET PREVIOUS BITS
00001125 30C3           <1>       xor    bl, al              ; SEE IF ANY DIFFERENT
00001127 80E307         <1>       and    bl, KB_LEDS         ; ISOLATE INDICATOR BITS
0000112A 7405           <1>       jz     short UP0           ; IF NO CHANGE BYPASS UPDATE
0000112C E8C0050000     <1>       call   SND_LED                   ; GO TURN ON MODE INDICATORS
                        <1> UP0:
00001131 6658           <1>       pop    ax                  ; RESTORE DATA IN
                        <1> ;--------------------------------------------------------------------------
                        <1> ;     START OF KEY PROCESSING                                        ;
                        <1> ;--------------------------------------------------------------------------
00001133 88C4           <1>       mov    ah, al              ; SAVE SCAN CODE IN AH ALSO
                        <1>       ;
                        <1>       ;----- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
00001135 3CFF           <1>       cmp    al, KB_OVER_RUN           ; IS THIS AN OVERRUN CHAR
00001137 0F843F050000   <1>         je     K62                     ; BUFFER_FULL_BEEP
                        <1>       ;
                        <1> K16:
0000113D 8A3D[986F0000] <1>       mov    bh, [KB_FLAG_3]           ; LOAD FLAGS FOR TESTING
                        <1>       ;
                        <1>       ;----- TEST TO SEE IF A READ_ID IS IN PROGRESS
00001143 F6C7C0         <1>       test   bh, RD_ID+LC_AB     ; ARE WE DOING A READ ID?
00001146 7449           <1>       jz     short NOT_ID        ; CONTINUE IF NOT
00001148 7917           <1>       jns    short TST_ID_2            ; IS THE RD_ID FLAG ON?
0000114A 3CAB           <1>       cmp    al, ID_1            ; IS THIS THE 1ST ID CHARACTER?
0000114C 7507           <1>       jne    short RST_RD_ID
0000114E 800D[986F0000]40 <1>     or     byte [KB_FLAG_3], LC_AB ; INDICATE 1ST ID WAS OK
                        <1> RST_RD_ID:
00001155 8025[986F0000]7F <1>     and    byte [KB_FLAG_3], ~RD_ID ; RESET THE READ ID FLAG
                        <1>         ;jmp    short ID_EX           ; AND EXIT
0000115C E924020000     <1>       jmp    K26
                        <1>       ;
                        <1> TST_ID_2:
00001161 8025[986F0000]BF <1>     and    byte [KB_FLAG_3], ~LC_AB ; RESET FLAG
00001168 3C54           <1>       cmp    al, ID_2A           ; IS THIS THE 2ND ID CHARACTER?
0000116A 7419           <1>         je   short KX_BIT        ; JUMP IF SO
0000116C 3C41           <1>       cmp    al, ID_2            ; IS THIS THE 2ND ID CHARACTER?
                        <1>         ;jne short ID_EX         ; LEAVE IF NOT
0000116E 0F8511020000   <1>       jne    K26
                        <1>       ;
                        <1>       ;----- A READ ID SAID THAT IT WAS ENHANCED KEYBOARD
00001174 F6C720         <1>       test   bh, SET_NUM_LK             ; SHOULD WE SET NUM LOCK?
00001177 740C           <1>         jz     short KX_BIT           ; EXIT IF NOT
00001179 800D[956F0000]20 <1>     or     byte [KB_FLAG], NUM_STATE ; FORCE NUM LOCK ON
00001180 E86C050000     <1>       call   SND_LED                   ; GO SET THE NUM LOCK INDICATOR
                        <1> KX_BIT:
00001185 800D[986F0000]10 <1>     or     byte [KB_FLAG_3], KBX     ; INDICATE ENHANCED KEYBOARD WAS FOUND
0000118C E9F4010000     <1> ID_EX:      jmp    K26               ; EXIT
                        <1>       ;
                        <1> NOT_ID:
00001191 3CE0           <1>       cmp    al, MC_E0           ; IS THIS THE GENERAL MARKER CODE?
00001193 750C           <1>       jne    short TEST_E1
00001195 800D[986F0000]12 <1>     or     byte [KB_FLAG_3], LC_E0+KBX ; SET FLAG BIT, SET KBX, AND
                        <1>       ;jmp   short EXIT          ; THROW AWAY THIS CODE
0000119C E9EB010000     <1>       jmp    K26A
                        <1> TEST_E1:
000011A1 3CE1           <1>       cmp    al, MC_E1           ; IS THIS THE PAUSE KEY?
000011A3 750C           <1>       jne    short NOT_HC
000011A5 800D[986F0000]11 <1>     or     byte [KB_FLAG_3], LC_E1+KBX ; SET FLAG BIT, SET KBX, AND
000011AC E9DB010000     <1> EXIT: jmp    K26A                ; THROW AWAY THIS CODE
                        <1>       ;
                        <1> NOT_HC:
000011B1 247F           <1>       and    al, 07Fh            ; TURN OFF THE BREAK BIT
000011B3 F6C702         <1>       test   bh, LC_E0           ; LAST CODE THE E0 MARKER CODE
000011B6 7414           <1>       jz     short NOT_LC_E0           ; JUMP IF NOT
                        <1>       ;
000011B8 BF[826E0000]   <1>       mov    edi, _K6+6          ; IS THIS A SHIFT KEY?
000011BD AE             <1>       scasb
000011BE 0F84C1010000   <1>         je     K26 ; K16B              ; YES, THROW AWAY & RESET FLAG
```

```
                        <1>       scasb
000011C4 AE             <1>       jne    short K16A          ; NO, CONTINUE KEY PROCESSING
000011C5 757C           <1>       ;jmp   short K16B          ; YES, THROW AWAY & RESET FLAG
                        <1>       jmp    K26
000011C7 E9B9010000     <1>       ;
                        <1> NOT_LC_E0:
                        <1>       test   bh, LC_E1           ; LAST CODE THE E1 MARKER CODE?
000011CC F6C701         <1>       jz     short T_SYS_KEY           ; JUMP IF NOT
000011CF 7435           <1>       mov    ecx, 4              ; LENGHT OF SEARCH
000011D1 B904000000     <1>       mov    edi, _K6+4          ; IS THIS AN ALT, CTL, OR SHIFT?
000011D6 BF[806E0000]   <1>       repne  scasb               ; CHECK IT
000011DB F2AE           <1>       ;je    short EXIT          ; THROW AWAY IF SO
                        <1>       je     K26A
000011DD 0F84A9010000   <1>       ;
                        <1>       cmp    al, NUM_KEY         ; IS IT THE PAUSE KEY?
000011E3 3C45           <1>       ;jne   short K16B          ; NO, THROW AWAY & RESET FLAG
                        <1>       jne    K26
000011E5 0F859A010000   <1>       test   ah, 80h                   ; YES, IS IT THE BREAK OF THE KEY?
000011EB F6C480         <1>       ;jnz   short K16B          ; YES, THROW THIS AWAY, TOO
                        <1>       jnz    K26
000011EE 0F8591010000   <1>         ; 20/02/2015
                        <1>       test   byte [KB_FLAG_1],HOLD_STATE ;  NO, ARE WE PAUSED ALREADY?
000011F4 F605[966F0000]08 <1>     ;jnz   short K16B          ;  YES, THROW AWAY
                        <1>       jnz    K26
000011FB 0F8584010000   <1>       jmp     K39P                    ; NO, THIS IS THE REAL PAUSE STATE
00001201 E9E1020000     <1>       ;
                        <1>       ;----- TEST FOR SYSTEM KEY
                        <1> T_SYS_KEY:
                        <1>       cmp    al, SYS_KEY         ; IS IT THE SYSTEM KEY?
00001206 3C54           <1>       jnz    short K16A          ; CONTINUE IF NOT
00001208 7539           <1>       ;
                        <1>       test   ah, 80h                   ; CHECK IF THIS A BREAK CODE
0000120A F6C480         <1>       jnz    short K16C          ; DO NOT TOUCH SYSTEM INDICATOR IF TRUE
0000120D 7524           <1>       ;
                        <1>       test   byte [KB_FLAG_1], SYS_SHIFT ; SEE IF IN SYSTEM KEY HELD DOWN
0000120F F605[966F0000]04 <1>       ;jnz short K16B          ; IF YES, DO NOT PROCESS SYSTEM INDICATOR
                        <1>       jnz     K26
00001216 0F8569010000   <1>       ;
                        <1>       or     byte [KB_FLAG_1], SYS_SHIFT ; INDICATE SYSTEM KEY DEPRESSED
0000121C 800D[966F0000]04 <1>       mov    al, EOI                  ; END OF INTERRUPT COMMAND
00001223 B020           <1>       out    20h, al ;out INTA00, al    ; SEND COMMAND TO INTERRUPT CONTROL PORT
00001225 E620           <1>                                       ; INTERRUPT-RETURN-NO-EOI
                        <1>       mov    al, ENA_KBD         ; INSURE KEYBOARD IS ENABLED
00001227 B0AE           <1>       call   SHIP_IT                   ; EXECUTE ENABLE
00001229 E862040000     <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
0000122E E965010000     <1>         jmp     K27A                    ; END PROCESSING
                        <1>       ;
                        <1> ;K16B:      jmp    K26                ; IGNORE SYSTEM KEY
                        <1>       ;
                        <1> K16C:
00001233 8025[966F0000]FB <1>       and    byte [KB_FLAG_1], ~SYS_SHIFT ; TURN OFF SHIFT KEY HELD DOWN
0000123A B020           <1>       mov    al, EOI                  ; END OF INTERRUPT COMMAND
0000123C 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>       ;
0000123E E94E010000     <1>       jmp     K27                ; IGNORE SYSTEM KEY
                        <1>       ;
                        <1>       ;----- TEST FOR SHIFT KEYS
                        <1> K16A:
00001243 8A1D[956F0000] <1>       mov    bl, [KB_FLAG]       ; PUT STATE FLAGS IN BL
00001249 BF[7C6E0000]   <1>       mov    edi, _K6            ; SHIFT KEY TABLE offset
0000124E B908000000     <1>       mov    ecx,  K6L           ; LENGTH
00001253 F2AE           <1>       repne  scasb               ; LOOK THROUGH THE TABLE FOR A MATCH
00001255 88E0           <1>       mov    al, ah              ; RECOVER SCAN CODE
00001257 0F8510010000   <1>         jne     K25                     ; IF NO MATCH, THEN SHIFT NOT FOUND
                        <1>       ;
                        <1>       ;------      SHIFT KEY FOUND
                        <1> K17:
                        <1>         sub     edi, _K6+1              ; ADJUST PTR TO SCAN CODE MATCH
0000125D 81EF[7D6E0000] <1>             mov     ah, [edi+_K7]           ; GET MASK INTO AH
00001263 8AA7[846E0000] <1>       mov    cl, 2               ; SETUP COUNT FOR FLAG SHIFTS
00001269 B102           <1>       test   al, 80h                  ; TEST FOR BREAK KEY
0000126B A880           <1>         jnz     K23                     ; JUMP OF BREAK
0000126D 0F8596000000   <1>       ;
                        <1>       ;----- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
                        <1> K17C:
00001273 80FC10         <1>       cmp    ah, SCROLL_SHIFT
00001276 732B           <1>       jae    short K18           ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
                        <1>       ;
                        <1>       ;----- PLAIN SHIFT KEY, SET SHIFT ON
00001278 0825[956F0000] <1>       or     [KB_FLAG], ah       ; TURN ON SHIFT BIT
0000127E A80C           <1>         test    al, CTL_SHIFT+ALT_SHIFT ; IS IT ALT OR CTRL?
                        <1>       ;jnz   short K17D          ; YES, MORE FLAGS TO SET
00001280 0F84FF000000   <1>       jz     K26                 ; NO, INTERRUPT RETURN
                        <1> K17D:
00001286 F6C702         <1>       test   bh, LC_E0           ; IS THIS ONE OF NEW KEYS?
00001289 740B           <1>       jz     short K17E          ; NO, JUMP
0000128B 0825[986F0000] <1>       or     [KB_FLAG_3], ah           ; SET BITS FOR RIGHT CTRL, ALT
00001291 E9EF000000     <1>       jmp    K26                 ; INTERRUPT RETURN
                        <1> K17E:
00001296 D2EC           <1>       shr    ah, cl              ; MOVE FLAG BITS TWO POSITIONS
00001298 0825[966F0000] <1>       or     [KB_FLAG_1], ah           ; SET BITS FOR LEFT CTRL, ALT
0000129E E9E2000000     <1>       jmp    K26
                        <1>       ;
                        <1>       ;----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
```

```
                        <1> K18:                            ; SHIFT-TOGGLE
000012A3 F6C304         <1>         test   bl, CTL_SHIFT              ; CHECK CTL SHIFT STATE
                        <1>           ;jz        short K18A               ; JUMP IF NOT CTL STATE
000012A6 0F85C1000000   <1>           jnz     K25                       ; JUMP IF CTL STATE
                        <1> K18A:
000012AC 3C52           <1>         cmp    al, INS_KEY          ; CHECK FOR INSERT KEY
000012AE 7524           <1>         jne    short K22            ; JUMP IF NOT INSERT KEY
000012B0 F6C308         <1>         test   bl, ALT_SHIFT              ; CHECK FOR ALTERNATE SHIFT
                        <1>                ;jz    short K18B           ; JUMP IF NOT ALTERNATE SHIFT
000012B3 0F85B4000000   <1>           jnz     K25                       ; JUMP IF ALTERNATE SHIFT
                        <1> K18B:
000012B9 F6C702         <1>         test   bh, LC_E0 ;20/02/2015      ; IS THIS NEW INSERT KEY?
000012BC 7516           <1>         jnz    short K22            ; YES, THIS ONE'S NEVER A '0'
                        <1> K19:
000012BE F6C320         <1>         test   bl, NUM_STATE              ; CHECK FOR BASE STATE
000012C1 750C           <1>         jnz    short K21            ; JUMP IF NUM LOCK IS ON
000012C3 F6C303         <1>         test   bl, LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
000012C6 740C           <1>         jz     short K22            ; JUMP IF BASE STATE
                        <1> K20:                            ; NUMERIC ZERO, NOT INSERT KEY
000012C8 88C4           <1>         mov    ah, al               ; PUT SCAN CODE BACK IN AH
000012CA E99E000000     <1>           jmp     K25                       ; NUMERAL '0', STNDRD. PROCESSING
                        <1> K21:                            ; MIGHT BE NUMERIC
000012CF F6C303         <1>         test   bl, LEFT_SHIFT+RIGHT_SHIFT
000012D2 74F4           <1>         jz     short K20            ; IS NUMERIC, STD. PROC.
                        <1>         ;
                        <1> K22:                            ; SHIFT TOGGLE KEY HIT; PROCESS IT
000012D4 8425[966F0000] <1>         test   ah, [KB_FLAG_1]      ; IS KEY ALREADY DEPRESSED
000012DA 0F85A5000000   <1>           jnz     K26                       ; JUMP IF KEY ALREADY DEPRESSED
                        <1> K22A:
000012E0 0825[966F0000] <1>           or      [KB_FLAG_1], ah ; INDICATE THAT THE KEY IS DEPRESSED
000012E6 3025[956F0000] <1>         xor    [KB_FLAG], ah        ; TOGGLE THE SHIFT STATE
                        <1>         ;
                        <1>         ;----- TOGGLE LED IF CAPS, NUM  OR SCROLL KEY DEPRESSED
000012EC F6C470         <1>         test   ah, CAPS_SHIFT+NUM_SHIFT+SCROLL_SHIFT ; SHIFT TOGGLE?
000012EF 7409           <1>         jz     short K22B           ; GO IF NOT
                        <1>         ;
000012F1 6650           <1>         push   ax                   ; SAVE SCAN CODE AND SHIFT MASK
000012F3 E8F9030000     <1>         call   SND_LED                    ; GO TURN MODE INDICATORS ON
000012F8 6658           <1>         pop    ax                   ; RESTORE SCAN CODE
                        <1> K22B:
000012FA 3C52           <1>         cmp    al, INS_KEY          ; TEST FOR 1ST MAKE OF INSERT KEY
000012FC 0F8583000000   <1>           jne     K26                       ; JUMP IF NOT INSERT KEY
00001302 88C4           <1>         mov    ah, al               ; SCAN CODE IN BOTH HALVES OF AX
00001304 E999000000     <1>           jmp     K28                       ; FLAGS UPDATED, PROC. FOR BUFFER
                        <1>         ;
                        <1>         ;----- BREAK SHIFT FOUND
                        <1> K23:                            ; BREAK-SHIFT-FOUND
00001309 80FC10         <1>         cmp    ah, SCROLL_SHIFT     ; IS THIS A TOGGLE KEY
0000130C F6D4           <1>         not    ah                   ; INVERT MASK
0000130E 7355           <1>         jae    short K24            ; YES, HANDLE BREAK TOGGLE
00001310 2025[956F0000] <1>         and    [KB_FLAG], ah        ; TURN OFF SHIFT BIT
00001316 80FCFB         <1>         cmp    ah, ~CTL_SHIFT             ; IS THIS ALT OR CTL?
00001319 7730           <1>         ja     short K23D           ; NO, ALL DONE
                        <1>         ;
0000131B F6C702         <1>         test   bh, LC_E0            ; 2ND ALT OR CTL?
0000131E 7408           <1>         jz     short K23A           ; NO, HANSLE NORMALLY
00001320 2025[986F0000] <1>         and    [KB_FLAG_3], ah            ; RESET BIT FOR RIGHT ALT OR CTL
00001326 EB08           <1>         jmp    short K23B           ; CONTINUE
                        <1> K23A:
00001328 D2FC           <1>         sar    ah, cl               ; MOVE THE MASK BIT TWO POSITIONS
0000132A 2025[966F0000] <1>         and    [KB_FLAG_1], ah            ; RESET BIT FOR LEFT ALT AND CTL
                        <1> K23B:
00001330 88C4           <1>         mov    ah, al               ; SAVE SCAN CODE
00001332 A0[986F0000]   <1>         mov    al, [KB_FLAG_3]            ; GET RIGHT ALT & CTRL FLAGS
00001337 D2E8           <1>         shr    al, cl               ; MOVE TO BITS 1 & 0
00001339 0A05[966F0000] <1>         or     al, [KB_FLAG_1]            ; PUT IN LEFT ALŞT & CTL FLAGS
0000133F D2E0           <1>         shl    al, cl               ; MOVE BACK TO BITS 3 & 2
00001341 240C           <1>         and    al, ALT_SHIFT+CTL_SHIFT ; FILTER OUT OTHER GARBAGE
00001343 0805[956F0000] <1>         or     [KB_FLAG], al        ; PUT RESULT IN THE REAL FLAGS
00001349 88E0           <1>         mov    al, ah
                        <1> K23D:
0000134B 3CB8           <1>         cmp    al, ALT_KEY+80h            ; IS THIS ALTERNATE SHIFT RELEASE
0000134D 7536           <1>         jne    short K26           ; INTERRUPT RETURN
                        <1>         ;
                        <1>         ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
0000134F A0[996F0000]   <1>         mov    al, [ALT_INPUT]
00001354 B400           <1>         mov    ah, 0                ; SCAN CODE OF 0
00001356 8825[996F0000] <1>         mov    [ALT_INPUT], ah      ; ZERO OUT THE FIELD
0000135C 3C00           <1>         cmp    al, 0                ; WAS THE INPUT = 0?
0000135E 7425           <1>         je     short K26            ; INTERRUPT_RETURN
                        <1>           ; 29/01/2016
                        <1>         ;jmp     K61                      ; IT WASN'T, SO PUT IN BUFFER
00001360 E9D0020000     <1>         jmp    _K60
                        <1>         ;
                        <1> K24:                            ; BREAK-TOGGLE
00001365 2025[966F0000] <1>         and    [KB_FLAG_1], ah      ; INDICATE NO LONGER DEPRESSED
0000136B EB18           <1>         jmp    short K26            ; INTERRUPT_RETURN
                        <1>         ;
                        <1>         ;----- TEST FOR HOLD STATE
                        <1>                                     ; AL, AH = SCAN CODE
                        <1> K25:                            ; NO-SHIFT-FOUND
0000136D 3C80           <1>         cmp    al, 80h                    ; TEST FOR BREAK KEY
0000136F 7314           <1>         jae    short K26           ; NOTHING FOR BREAK CHARS FROM HERE ON
00001371 F605[966F0000]08 <1>       test   byte [KB_FLAG_1], HOLD_STATE ; ARE WE IN HOLD STATE
00001378 7428           <1>         jz     short K28           ; BRANCH AROUND TEST IF NOT
0000137A 3C45           <1>         cmp    al, NUM_KEY
0000137C 7407           <1>         je     short K26           ; CAN'T END HOLD ON NUM_LOCK
0000137E 8025[966F0000]F7 <1>       and    byte [KB_FLAG_1], ~HOLD_STATE ; TURN OFF THE HOLD STATE BIT
                        <1>         ;
                        <1> K26:
00001385 8025[986F0000]FC <1>       and    byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; RESET LAST CHAR H.C. FLAG
                        <1> K26A:                           ; INTERRUPT-RETURN
0000138C FA             <1>         cli                         ; TURN OFF INTERRUPTS
0000138D B020           <1>         mov    al, EOI                    ; END OF INTERRUPT COMMAND
0000138F E620           <1>         out    20h, al      ;out INTA00, al     ; SEND COMMAND TO INTERRUPT CONTROL PORT
```

```
                        <1> K27:                            ; INTERRUPT-RETURN-NO-EOI
00001391 B0AE           <1>         mov    al, ENA_KBD      ; INSURE KEYBOARD IS ENABLED
00001393 E8F8020000     <1>         call   SHIP_IT                  ; EXECUTE ENABLE
                        <1> K27A:
00001398 FA             <1>         cli                     ; DISABLE INTERRUPTS
                        <1>         ;;mov  byte [intflg], 0 ; 07/01/2017 ;; 15/01/2017
00001399 07             <1>         pop    es               ; RESTORE REGISTERS
0000139A 1F             <1>         pop    ds
0000139B 5F             <1>         pop    edi
0000139C 5E             <1>         pop    esi
0000139D 5A             <1>         pop    edx
0000139E 59             <1>         pop    ecx
0000139F 5B             <1>         pop    ebx
000013A0 58             <1>         pop    eax
                        <1>         ;pop   ebp
000013A1 CF             <1>         iretd                   ; RETURN
                        <1>
                        <1>         ;----- NOT IN HOLD STATE
                        <1> K28:                            ; NO-HOLD-STATE
000013A2 3C58           <1>         cmp    al, 88           ; TEST FOR OUT-OF-RANGE SCAN CODES
000013A4 77DF           <1>         ja     short K26        ; IGNORE IF OUT-OF-RANGE
                        <1>         ;
000013A6 F6C308         <1>         test   bl, ALT_SHIFT            ; ARE WE IN ALTERNATE SHIFT
                        <1>           ;jz  short K28A       ; IF NOT ALTERNATE
000013A9 0F84F1000000   <1>           jz     K38
                        <1>         ;
000013AF F6C710         <1>         test   bh, KBX                  ; IS THIS THE ENCHANCED KEYBOARD?
000013B2 740D           <1>         jz     short K29        ; NO, ALT STATE IS REAL
                        <1>          ;28/02/2015
000013B4 F605[966F0000]04 <1>       test   byte [KB_FLAG_1], SYS_SHIFT ; YES, IS SYSREQ KEY DOWN?
                        <1>         ;jz    short K29        ;  NO, ALT STATE IS REAL
000013BB 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
000013C1 F6C304         <1>         test   bl, CTL_SHIFT            ; ARE WE IN CONTROL SHIFT ALSO?
000013C4 740B           <1>         jz     short K31        ; NO_RESET
000013C6 3C53           <1>         cmp    al, DEL_KEY      ; CTL-ALT STATE, TEST FOR DELETE KEY
000013C8 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.
000013CA B0FE           <1>         mov    al, SHUT_CMD     ; SHUTDOWN COMMAND
000013CC E664           <1>         out    STATUS_PORT, al          ; SEND TO KEYBOARD CONTROL PORT
                        <1> khere:
000013CE F4             <1>         hlt                     ; WAIT FOR 80286 RESET
000013CF EBFD           <1>         jmp    short khere      ; INSURE HALT
                        <1>
                        <1>         ;
                        <1>         ;----- IN ALTERNATE SHIFT, RESET NOT FOUND
                        <1> K31:                            ; NO-RESET
000013D1 3C39           <1>         cmp    al, 57           ; TEST FOR SPACE KEY
000013D3 7507           <1>         jne    short K311       ; NOT THERE
000013D5 B020           <1>         mov    al, ' '                  ; SET SPACE CHAR
000013D7 E948020000     <1>           jmp    K57                    ; BUFFER_FILL
                        <1> K311:
000013DC 3C0F           <1>         cmp    al, 15           ; TEST FOR TAB KEY
000013DE 7509           <1>         jne    short K312       ; NOT THERE
000013E0 66B800A5       <1>         mov    ax, 0A500h       ; SET SPECIAL CODE FOR ALT-TAB
000013E4 E93B020000     <1>           jmp    K57                    ; BUFFER_FILL
                        <1> K312:
000013E9 3C4A           <1>         cmp    al, 74           ; TEST FOR KEY PAD -
000013EB 0F84A2000000   <1>           je     K37B                   ; GO PROCESS
000013F1 3C4E           <1>         cmp    al, 78           ; TEST FOR KEY PAD +
000013F3 0F849A000000   <1>           je     K37B                   ; GO PROCESS
                        <1>         ;
                        <1>         ;----- LOOK FOR KEY PAD ENTRY
                        <1> K32:                            ; ALT-KEY-PAD
000013F9 BF[586E0000]   <1>         mov    edi, K30         ; ALT-INPUT-TABLE offset
000013FE B90A000000     <1>         mov    ecx, 10                  ; LOOK FOR ENTRY USING KEYPAD
00001403 F2AE           <1>         repne  scasb            ; LOOK FOR MATCH
00001405 7525           <1>         jne    short K33        ; NO_ALT_KEYPAD
00001407 F6C702         <1>         test   bh, LC_E0        ; IS THIS ONE OF THE NEW KEYS?
0000140A 0F858A000000   <1>           jnz    K37C                   ; YES, JUMP, NOT NUMPAD KEY
00001410 81EF[596E0000] <1>         sub    edi, K30+1       ; DI NOW HAS ENTRY VALUE
00001416 A0[996F0000]   <1>         mov    al, [ALT_INPUT]  ; GET THE CURRENT BYTE
0000141B B40A           <1>         mov    ah, 10           ; MULTIPLY BY 10
0000141D F6E4           <1>         mul    ah
0000141F 6601F8         <1>         add    ax, di           ; ADD IN THE LATEST ENTRY
00001422 A2[996F0000]   <1>         mov    [ALT_INPUT], al  ; STORE IT AWAY
                        <1> ;K32A:
00001427 E959FFFFFF     <1>           jmp    K26                    ; THROW AWAY THAT KEYSTROKE
                        <1>         ;
                        <1>         ;----- LOOK FOR SUPERSHIFT ENTRY
                        <1> K33:                            ; NO-ALT-KEYPAD
0000142C C605[996F0000]00 <1>         mov    byte [ALT_INPUT], 0      ; ZERO ANY PREVIOUS ENTRY INTO INPUT
00001433 B91A000000     <1>         mov    ecx, 26                  ; (DI),(ES) ALREADY POINTING
00001438 F2AE           <1>         repne  scasb            ; LOOK FOR MATCH IN ALPHABET
0000143A 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
0000143C 3C02           <1>         cmp    al, 2            ; KEY WITH '1' ON IT
0000143E 7253           <1>         jb     short K37B       ; MUST BE ESCAPE
00001440 3C0D           <1>         cmp    al, 13           ; IS IT IN THE REGION
00001442 7705           <1>         ja     short K35        ; NO, ALT SOMETHING ELSE
00001444 80C476         <1>         add    ah, 118                  ; CONVERT PSEUDO SCAN CODE TO RANGE
00001447 EB43           <1>         jmp    short K37A       ; GO FILL THE BUFFER
                        <1>         ;
                        <1>         ;----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
```

```
 993                                    <1> K35:                                ; ALT-FUNCTION
 994 00001449 3C57                      <1>         cmp    al, F11_M            ; IS IT F11?
 995 0000144B 7209                      <1>         jb     short K35A ; 20/02/2015   ; NO, BRANCH
 996 0000144D 3C58                      <1>         cmp    al, F12_M            ; IS IT F12?
 997 0000144F 7705                      <1>         ja     short K35A ; 20/02/2015   ; NO, BRANCH
 998 00001451 80C434                    <1>         add    ah, 52               ; CONVERT TO PSEUDO SCAN CODE
 999 00001454 EB36                      <1>         jmp    short K37A           ; GO FILL THE BUFFER
1000                                    <1> K35A:
1001 00001456 F6C702                    <1>         test   bh, LC_E0            ; DO WE HAVE ONE OF THE NEW KEYS?
1002 00001459 7422                      <1>         jz     short K37            ; NO, JUMP
1003 0000145B 3C1C                      <1>         cmp    al, 28               ; TEST FOR KEYPAD ENTER
1004 0000145D 7509                      <1>           jne     short K35B              ; NOT THERE
1005 0000145F 66B800A6                  <1>         mov    ax, 0A600h           ; SPECIAL CODE
1006 00001463 E9BC010000                <1>         jmp    K57                  ; BUFFER FILL
1007                                    <1> K35B:
1008 00001468 3C53                      <1>         cmp    al, 83               ; TEST FOR DELETE KEY
1009 0000146A 742E                      <1>         je     short K37C           ; HANDLE WITH OTHER EDIT KEYS
1010 0000146C 3C35                      <1>         cmp    al, 53               ; TEST FOR KEYPAD /
1011                                    <1>         ;jne   short K32A           ; NOT THERE, NO OTHER E0 SPECIALS
1012 0000146E 0F8511FFFFFF              <1>           jne     K26
1013 00001474 66B800A4                  <1>         mov    ax, 0A400h           ; SPECIAL CODE
1014 00001478 E9A7010000                <1>         jmp    K57                  ; BUFFER FILL
1015                                    <1> K37:
1016 0000147D 3C3B                      <1>         cmp    al, 59               ; TEST FOR FUNCTION KEYS (F1)
1017 0000147F 7212                      <1>           jb      short K37B             ; NO FN, HANDLE W/OTHER EXTENDED
1018 00001481 3C44                      <1>         cmp    al, 68               ; IN KEYPAD REGION?
1019                                    <1>           ;ja  short K32A           ; IF SO, IGNORE
1020 00001483 0F87FCFEFFFF              <1>           ja      K26
1021 00001489 80C42D                    <1>         add    ah, 45               ; CONVERT TO PSEUDO SCAN CODE
1022                                    <1> K37A:
1023 0000148C B000                      <1>         mov    al, 0                ; ASCII CODE OF ZERO
1024 0000148E E991010000                <1>           jmp     K57                     ; PUT IT IN THE BUFFER
1025                                    <1> K37B:
1026 00001493 B0F0                      <1>         mov    al, 0F0h             ; USE SPECIAL ASCII CODE
1027 00001495 E98A010000                <1>         jmp     K57                     ; PUT IT IN THE BUFFER
1028                                    <1> K37C:
1029 0000149A 0450                      <1>         add    al, 80               ; CONVERT SCAN CODE (EDIT KEYS)
1030 0000149C 88C4                      <1>         mov    ah, al               ; (SCAN CODE NOT IN AH FOR INSERT)
1031 0000149E EBEC                      <1>         jmp     short K37A              ; PUT IT IN THE BUFFER
1032                                    <1>         ;
1033                                    <1>         ;----- NOT IN ALTERNATE SHIFT
1034                                    <1> K38:                                ; NOT-ALT-SHIFT
1035                                    <1>                                     ; BL STILL HAS SHIFT FLAGS
1036 000014A0 F6C304                    <1>         test   bl, CTL_SHIFT              ; ARE WE IN CONTROL SHIFT?
1037                                    <1>         ;jnz   short K38A           ; YES, START PROCESSING
1038 000014A3 0F84B0000000              <1>           jz      K44                     ; NOT-CTL-SHIFT
1039                                    <1>         ;
1040                                    <1>         ;----- CONTROL SHIFT, TEST SPECIAL CHARACTERS
1041                                    <1>         ;----- TEST FOR BREAK
1042                                    <1> K38A:
1043 000014A9 3C46                      <1>         cmp    al, SCROLL_KEY            ; TEST FOR BREAK
1044 000014AB 7531                      <1>         jne    short K39            ; JUMP, NO-BREAK
1045 000014AD F6C710                    <1>         test   bh, KBX                  ; IS THIS THE ENHANCED KEYBOARD?
1046 000014B0 7405                      <1>         jz     short K38B           ; NO, BREAK IS VALID
1047 000014B2 F6C702                    <1>         test   bh, LC_E0            ; YES, WAS LAST CODE AN E0?
1048 000014B5 7427                      <1>         jz     short K39            ; NO-BREAK, TEST FOR PAUSE
1049                                    <1> K38B:
1050 000014B7 8B1D[A26F0000]            <1>         mov    ebx, [BUFFER_HEAD]   ; RESET BUFFER TO EMPTY
1051 000014BD 891D[A66F0000]            <1>         mov    [BUFFER_TAIL], ebx
1052 000014C3 C605[946F0000]80          <1>         mov    byte [BIOS_BREAK], 80h  ; TURN ON BIOS_BREAK BIT
1053                                    <1>         ;
1054                                    <1>         ;----- ENABLE KEYBOARD
1055 000014CA B0AE                      <1>         mov    al, ENA_KBD          ; ENABLE KEYBOARD
1056 000014CC E8BF010000                <1>         call   SHIP_IT                  ; EXECUTE ENABLE
1057                                    <1>         ;
1058                                    <1>         ; CTRL+BREAK code here !!!
1059                                    <1>         ;INT   1BH                  ; BREAK INTERRUPT VECTOR
1060                                    <1>         ; 17/10/2015
1061 000014D1 E80A610000                <1>         call   ctrlbrk ; control+break subroutine
1062                                    <1>         ;
1063 000014D6 6629C0                    <1>         sub    ax, ax               ; PUT OUT DUMMY CHARACTER
1064 000014D9 E946010000                <1>           jmp     K57                     ; BUFFER_FILL
1065                                    <1>         ;
1066                                    <1>         ;----- TEST FOR PAUSE
1067                                    <1> K39:                                ; NO_BREAK
1068 000014DE F6C710                    <1>         test   bh, KBX                  ; IS THIS THE ENHANCED KEYBOARD?
1069 000014E1 7537                      <1>         jnz    short K41            ; YES, THEN THIS CAN'T BE PAUSE
1070 000014E3 3C45                      <1>         cmp    al, NUM_KEY          ; LOOK FOR PAUSE KEY
1071 000014E5 7533                      <1>         jne    short K41            ; NO-PAUSE
1072                                    <1> K39P:
1073 000014E7 800D[966F0000]08          <1>         or     byte [KB_FLAG_1], HOLD_STATE ; TURN ON THE HOLD FLAG
1074                                    <1>         ;
1075                                    <1>         ;----- ENABLE KEYBOARD
1076 000014EE B0AE                      <1>         mov    al, ENA_KBD          ; ENABLE KEYBOARD
1077 000014F0 E89B010000                <1>         call   SHIP_IT                  ; EXECUTE ENABLE
1078                                    <1> K39A:
1079 000014F5 B020                      <1>         mov    al, EOI                  ; END OF INTERRUPT TO CONTROL PORT
1080 000014F7 E620                      <1>         out    20h, al ;out INTA00, al  ; ALLOW FURTHER KEYSTROKE INTERRUPTS
1081                                    <1>         ;
1082                                    <1>         ;----- DURING PAUSE INTERVAL, TURN COLOR CRT BACK ON
1083 000014F9 803D[CA6F0000]07          <1>           cmp     byte [CRT_MODE], 7      ; IS THIS BLACK AND WHITE CARD
1084 00001500 740A                      <1>           je      short K40                  ; YES, NOTHING TO DO
1085 00001502 66BAD803                  <1>         mov    dx, 03D8h            ; PORT FOR COLOR CARD
1086 00001506 A0[CB6F0000]              <1>           mov     al, [CRT_MODE_SET]     ; GET THE VALUE OF THE CURRENT MODE
1087 0000150B EE                        <1>         out    dx, al               ; SET THE CRT MODE, SO THAT CRT IS ON
1088                                    <1>         ;
1089                                    <1> K40:                                ; PAUSE-LOOP
1090 0000150C F605[966F0000]08          <1>           test    byte [KB_FLAG_1], HOLD_STATE ; CHECK HOLD STATE FLAG
1091 00001513 75F7                      <1>         jnz    short K40            ; LOOP UNTIL FLAG TURNED OFF
1092                                    <1>         ;
1093 00001515 E977FEFFFF                <1>           jmp     K27                     ; INTERRUPT_RETURN_NO_EOI
1094                                    <1>           ;
1095                                    <1>         ;----- TEST SPECIAL CASE KEY 55
1096                                    <1> K41:                                ; NO-PAUSE
1097 0000151A 3C37                      <1>         cmp    al, 55               ; TEST FOR */PRTSC KEY
```

```
1098 0000151C 7513                <1>       jne    short K42           ; NOT-KEY-55
1099 0000151E F6C710              <1>       test   bh, KBX                    ; IS THIS THE ENHANCED KEYBOARD?
1100 00001521 7405                <1>       jz     short K41A          ; NO, CTL-PRTSC IS VALID
1101 00001523 F6C702              <1>       test   bh, LC_E0           ; YES, WAS LAST CODE AN E0?
1102 00001526 7421                <1>       jz     short K42B          ; NO, TRANSLATE TO A FUNCTION
1103                              <1> K41A:
1104 00001528 66B80072            <1>       mov    ax, 114*256         ; START/STOP PRINTING SWITCH
1105 0000152C E9F3000000          <1>         jmp     K57                      ; BUFFER_FILL
1106                              <1>       ;
1107                              <1>       ;----- SET UP TO TRANSLATE CONTROL SHIFT
1108                              <1> K42:                              ; NOT-KEY-55
1109 00001531 3C0F                <1>       cmp    al, 15              ; IS IT THE TAB KEY?
1110 00001533 7414                <1>       je     short K42B          ; YES, XLATE TO FUNCTION CODE
1111 00001535 3C35                <1>       cmp    al, 53              ; IS IT THE / KEY?
1112 00001537 750E                <1>       jne    short K42A          ; NO, NO MORE SPECIAL CASES
1113 00001539 F6C702              <1>       test   bh, LC_E0           ; YES, IS IT FROM THE KEY PAD?
1114 0000153C 7409                <1>       jz     short K42A          ; NO, JUST TRANSLATE
1115 0000153E 66B80095            <1>       mov    ax, 9500h           ; YES, SPECIAL CODE FOR THIS ONE
1116 00001542 E9DD000000          <1>       jmp    K57                 ; BUFFER FILL
1117                              <1> K42A:
1118                              <1>       ;;mov  ebx, _K8            ; SET UP TO TRANSLATE CTL
1119 00001547 3C3B                <1>       cmp    al, 59              ; IS IT IN CHARACTER TABLE?
1120                              <1>         ;jb  short K45F              ; YES, GO TRANSLATE CHAR
1121                              <1>       ;;jb   K56 ; 20/02/2015
1122                              <1>       ;;jmp  K64 ; 20/02/2015
1123                              <1> K42B:
1124 00001549 BB[8C6E0000]        <1>       mov    ebx, _K8            ; SET UP TO TRANSLATE CTL
1125 0000154E 0F82AE000000        <1>       jb     K56 ;; 20/02/2015
1126 00001554 E9B9000000          <1>       jmp    K64
1127                              <1>         ;
1128                              <1>       ;----- NOT IN CONTROL SHIFT
1129                              <1> K44:                              ; NOT-CTL-SHIFT
1130 00001559 3C37                <1>       cmp    al, 55              ; PRINT SCREEN KEY?
1131 0000155B 7528                <1>       jne    short K45          ; NOT PRINT SCREEN
1132 0000155D F6C710              <1>       test   bh, KBX                    ; IS THIS ENHANCED KEYBOARD?
1133 00001560 7407                <1>       jz     short K44A          ; NO, TEST FOR SHIFT STATE
1134 00001562 F6C702              <1>       test   bh, LC_E0           ; YES, LAST CODE A MARKER?
1135 00001565 7507                <1>       jnz    short K44B          ; YES, IS PRINT SCREEN
1136 00001567 EB41                <1>       jmp    short K45C          ; NO, TRANSLATE TO '*' CHARACTER
1137                              <1> K44A:
1138 00001569 F6C303              <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT ; NOT 101 KBD, SHIFT KEY DOWN?
1139 0000156C 743C                <1>       jz     short K45C          ; NO, TRANSLATE TO '*' CHARACTER
1140                              <1>       ;
1141                              <1>       ;----- ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
1142                              <1> K44B:
1143 0000156E B0AE                <1>       mov    al, ENA_KBD         ; INSURE KEYBOARD IS ENABLED
1144 00001570 E81B010000          <1>       call   SHIP_IT                    ; EXECUTE ENABLE
1145 00001575 B020                <1>       mov    al, EOI                    ; END OF CURRENT INTERRUPT
1146 00001577 E620                <1>       out    20h, al ;out INTA00, al    ; SO FURTHER THINGS CAN HAPPEN
1147                              <1>       ; Print Screen !!!         ; ISSUE PRINT SCREEN INTERRUPT (INT 05h)
1148                              <1>       ;PUSH  BP                  ; SAVE POINTER
1149                              <1>       ;INT   5H                  ; ISSUE PRINT SCREEN INTERRUPT
1150                              <1>       ;POP   BP                  ; RESTORE POINTER
1151 00001579 8025[986F0000]FC    <1>         and     byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; ZERO OUT THESE FLAGS
1152 00001580 E90CFEFFFF          <1>         jmp     K27                      ; GO BACK WITHOUT EOI OCCURRING
1153                              <1>       ;
1154                              <1>       ;----- HANDLE IN-CORE KEYS
1155                              <1> K45:                              ; NOT-PRINT-SCREEN
1156 00001585 3C3A                <1>       cmp    al, 58              ; TEST FOR IN-CORE AREA
1157 00001587 7734                <1>       ja     short K46           ; JUMP IF NOT
1158 00001589 3C35                <1>       cmp    al, 53              ; IS THIS THE '/' KEY?
1159 0000158B 7505                <1>       jne    short K45A          ; NO, JUMP
1160 0000158D F6C702              <1>       test   bh, LC_E0           ; WAS THE LAST CODE THE MARKER?
1161 00001590 7518                <1>       jnz    short K45C          ; YES, TRANSLATE TO CHARACTER
1162                              <1> K45A:
1163 00001592 B91A000000          <1>       mov    ecx, 26                    ; LENGHT OF SEARCH
1164 00001597 BF[626E0000]        <1>       mov    edi, K30+10         ; POINT TO TABLE OF A-Z CHARS
1165 0000159C F2AE                <1>       repne  scasb               ; IS THIS A LETTER KEY?
1166                              <1>              ; 20/02/2015
1167 0000159E 7505                <1>       jne    short K45B             ; NO, SYMBOL KEY
1168                              <1>       ;
1169 000015A0 F6C340              <1>       test   bl, CAPS_STATE             ; ARE WE IN CAPS LOCK?
1170 000015A3 750C                <1>       jnz    short K45D          ; TEST FOR SURE
1171                              <1> K45B:
1172 000015A5 F6C303              <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
1173 000015A8 750C                <1>       jnz    short K45E          ; YES, UPPERCASE
1174                              <1>                                  ; NO, LOWERCASE
1175                              <1> K45C:
1176 000015AA BB[E46E0000]        <1>       mov    ebx, K10            ; TRANSLATE TO LOWERCASE LETTERS
1177 000015AF EB51                <1>       jmp    short K56
1178                              <1> K45D:                             ; ALMOST-CAPS-STATE
1179 000015B1 F6C303              <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT ; CL ON. IS SHIFT ON, TOO?
1180 000015B4 75F4                <1>       jnz    short K45C          ; SHIFTED TEMP OUT OF CAPS STATE
1181                              <1> K45E:
1182 000015B6 BB[3C6F0000]        <1>       mov    ebx, K11            ; TRANSLATE TO UPPER CASE LETTERS
1183 000015BB EB45                <1> K45F: jmp    short K56
1184                              <1>       ;
1185                              <1>       ;----- TEST FOR KEYS F1 - F10
1186                              <1> K46:                              ; NOT IN-CORE AREA
1187 000015BD 3C44                <1>       cmp    al, 68              ; TEST FOR F1 - F10
1188                              <1>       ;ja    short K47           ; JUMP IF NOT
1189                              <1>       ;jmp   short K53           ; YES, GO DO FN KEY PROCESS
1190 000015BF 7635                <1>       jna    short K53
1191                              <1>       ;
1192                              <1>       ;----- HANDLE THE NUMERIC PAD KEYS
1193                              <1> K47:                              ; NOT F1 - F10
1194 000015C1 3C53                <1>       cmp    al, 83              ; TEST NUMPAD KEYS
1195 000015C3 772D                <1>       ja     short K52           ; JUMP IF NOT
1196                              <1>       ;
1197                              <1>       ;----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
1198                              <1> K48:
1199 000015C5 3C4A                <1>       cmp    al , 74                    ; SPECIAL CASE FOR MINUS
1200 000015C7 74ED                <1>       je     short K45E          ; GO TRANSLATE
1201 000015C9 3C4E                <1>       cmp    al , 78                    ; SPECIAL CASE FOR PLUS
1202 000015CB 74E9                <1>       je     short K45E          ; GO TRANSLATE
```

```
1203 000015CD F6C702             <1>       test   bh, LC_E0           ; IS THIS ONE OFTHE NEW KEYS?
1204 000015D0 750A               <1>       jnz    short K49           ; YES, TRANSLATE TO BASE STATE
1205                             <1>       ;
1206 000015D2 F6C320             <1>       test   bl, NUM_STATE       ; ARE WE IN NUM LOCK
1207 000015D5 7514               <1>       jnz    short K50           ; TEST FOR SURE
1208 000015D7 F6C303             <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
1209                             <1>       ;jnz   short K51           ; IF SHIFTED, REALLY NUM STATE
1210 000015DA 75DA               <1>       jnz    short K45E
1211                             <1>       ;
1212                             <1>       ;----- BASE CASE FOR KEYPAD
1213                             <1> K49:
1214 000015DC 3C4C               <1>       cmp    al, 76              ; SPECIAL CASE FOR BASE STATE 5
1215 000015DE 7504               <1>       jne    short K49A          ; CONTINUE IF NOT KEYPAD 5
1216 000015E0 B0F0               <1>       mov    al, 0F0h            ; SPECIAL ASCII CODE
1217 000015E2 EB40               <1>       jmp    short K57           ; BUFFER FILL
1218                             <1> K49A:
1219 000015E4 BB[E46E0000]       <1>       mov    ebx, K10            ; BASE CASE TABLE
1220 000015E9 EB27               <1>       jmp    short K64           ; CONVERT TO PSEUDO SCAN
1221                             <1>       ;
1222                             <1>       ;----- MIGHT BE NUM LOCK, TEST SHIFT STATUS
1223                             <1> K50:                             ; ALMOST-NUM-STATE
1224 000015EB F6C303             <1>         test    bl, LEFT_SHIFT+RIGHT_SHIFT
1225 000015EE 75EC               <1>       jnz    short K49           ; SHIFTED TEMP OUT OF NUM STATE
1226 000015F0 EBC4               <1> K51:  jmp    short K45E          ; REALLY NUM STATE
1227                             <1>       ;
1228                             <1>       ;----- TEST FOR THE NEW KEYS ON WT KEYBOARDS
1229                             <1> K52:                             ; NOT A NUMPAD KEY
1230 000015F2 3C56               <1>       cmp    al, 86              ; IS IT THE NEW WT KEY?
1231                             <1>       ;jne   short K53           ; JUMP IF NOT
1232                             <1>       ;jmp   short K45B          ; HANDLE WITH REST OF LETTER KEYS
1233 000015F4 74AF               <1>       je     short K45B
1234                             <1>       ;
1235                             <1>       ;----- MUST BE F11 OR F12
1236                             <1> K53:                             ; F1 - F10 COME HERE, TOO
1237 000015F6 F6C303             <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT ; TEST SHIFT STATE
1238 000015F9 74E1               <1>       jz     short K49           ; JUMP, LOWER CASE PSEUDO SC'S
1239                             <1>              ; 20/02/2015
1240 000015FB BB[3C6F0000]       <1>       mov    ebx, K11            ; UPPER CASE PSEUDO SCAN CODES
1241 00001600 EB10               <1>       jmp    short K64           ; TRANSLATE SCAN
1242                             <1>       ;
1243                             <1>       ;----- TRANSLATE THE CHARACTER
1244                             <1> K56:                             ; TRANSLATE-CHAR
1245 00001602 FEC8               <1>       dec    al                  ; CONVERT ORIGIN
1246 00001604 D7                 <1>       xlat                       ; CONVERT THE SCAN CODE TO ASCII
1247 00001605 F605[986F0000]02   <1>       test   byte [KB_FLAG_3], LC_E0   ; IS THIS A NEW KEY?
1248 0000160C 7416               <1>       jz     short K57           ; NO, GO FILL BUFFER
1249 0000160E B4E0               <1>       mov    ah, MC_E0           ; YES, PUT SPECIAL MARKER IN AH
1250 00001610 EB12               <1>       jmp    short K57           ; PUT IT INTO THE BUFFER
1251                             <1>       ;
1252                             <1>       ;----- TRANSLATE SCAN FOR PSEUDO SCAN CODES
1253                             <1> K64:                             ; TRANSLATE-SCAN-ORGD
1254 00001612 FEC8               <1>       dec    al                  ; CONVERT ORIGIN
1255 00001614 D7                 <1>              xlat                        ; CTL TABLE SCAN
1256 00001615 88C4               <1>       mov    ah, al              ; PUT VALUE INTO AH
1257 00001617 B000               <1>       mov    al, 0               ; ZERO ASCII CODE
1258 00001619 F605[986F0000]02   <1>       test   byte [KB_FLAG_3], LC_E0   ; IS THIS A NEW KEY?
1259 00001620 7402               <1>       jz     short K57           ; NO, GO FILL BUFFER
1260 00001622 B0E0               <1>       mov    al, MC_E0           ; YES, PUT SPECIAL MARKER IN AL
1261                             <1>       ;
1262                             <1>       ;----- PUT CHARACTER INTO BUFFER
1263                             <1> K57:                             ; BUFFER_FILL
1264 00001624 3CFF               <1>       cmp    al, -1              ; IS THIS AN IGNORE CHAR
1265                             <1>         ;je  short K59           ; YES, DO NOTHING WITH IT
1266 00001626 0F8459FDFFFF       <1>       je      K26                ; YES, DO NOTHING WITH IT
1267 0000162C 80FCFF             <1>       cmp    ah, -1              ; LOOK FOR -1 PSEUDO SCAN
1268                             <1>         ;jne short K61           ; NEAR_INTERRUPT_RETURN
1269 0000162F 0F8450FDFFFF       <1>       je      K26                ; INTERRUPT_RETURN
1270                             <1> ;K59:                            ; NEAR_INTERRUPT_RETURN
1271                             <1> ;     jmp    K26                 ; INTERRUPT_RETURN
1272                             <1>
1273                             <1> _K60: ; 29/01/2016
1274 00001635 80FC68             <1>       cmp    ah, 68h      ; ALT + F1 key
1275 00001638 721F               <1>       jb     short K61
1276 0000163A 80FC6F             <1>       cmp    ah, 6Fh ; ALT + F8 key
1277 0000163D 771A               <1>       ja     short K61
1278                             <1>       ;
1279 0000163F 8A1D[EE890100]     <1>       mov    bl, [ACTIVE_PAGE]
1280 00001645 80C368             <1>       add    bl, 68h
1281 00001648 38E3               <1>       cmp    bl, ah
1282 0000164A 740D               <1>       je     short K61
1283 0000164C 6650               <1>       push   ax
1284 0000164E 88E0               <1>       mov    al, ah
1285 00001650 2C68               <1>       sub    al, 68h
1286 00001652 E858090000         <1>       call   set_active_page
1287 00001657 6658               <1>       pop    ax
1288                             <1> K61:                             ; NOT-CAPS-STATE
1289 00001659 8B1D[A66F0000]     <1>       mov    ebx, [BUFFER_TAIL]  ; GET THE END POINTER TO THE BUFFER
1290 0000165F 89DE               <1>       mov    esi, ebx            ; SAVE THE VALUE
1291 00001661 E857FAFFFF         <1>       call   _K4                 ; ADVANCE THE TAIL
1292 00001666 3B1D[A26F0000]     <1>       cmp    ebx, [BUFFER_HEAD]  ; HAS THE BUFFER WRAPPED AROUND
1293 0000166C 740E               <1>       je     short K62           ; BUFFER_FULL_BEEP
1294 0000166E 668906             <1>       mov    [esi], ax           ; STORE THE VALUE
1295 00001671 891D[A66F0000]     <1>       mov    [BUFFER_TAIL], ebx  ; MOVE THE POINTER UP
1296 00001677 E909FDFFFF         <1>       jmp    K26
1297                             <1>       ;;cli                      ; TURN OFF INTERRUPTS
1298                             <1>       ;;mov  al, EOI                   ; END OF INTERRUPT COMMAND
1299                             <1>       ;;out  INTA00, al          ; SEND COMMAND TO INTERRUPT CONTROL PORT
1300                             <1>       ;MOV   AL, ENA_KBD         ; INSURE KEYBOARD IS ENABLED
1301                             <1>       ;CALL  SHIP_IT                   ; EXECUTE ENABLE
1302                             <1>       ;MOV   AX, 9102H           ; MOVE IN POST CODE & TYPE
1303                             <1>       ;INT   15H                 ; PERFORM OTHER FUNCTION
1304                             <1>       ;;and  byte [KB_FLAG_3],~(LC_E0+LC_E1) ; RESET LAST CHAR H.C. FLAG
1305                             <1>       ;JMP   K27A                ; INTERRUPT_RETURN
1306                             <1>       ;;jmp   K27
1307                             <1>       ;
```

```
1308                            <1>     ;----- BUFFER IS FULL SOUND THE BEEPER
1309                            <1> K62:
1310 0000167C B020              <1>     mov   al, EOI                   ; ENABLE INTERRUPT CONTROLLER CHIP
1311 0000167E E620              <1>     out   INTA00, al
1312 00001680 66B9A602          <1>     mov   cx, 678                   ; DIVISOR FOR 1760 HZ
1313 00001684 B304              <1>     mov   bl, 4                ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)
1314 00001686 E86D0D0000        <1>     call  beep                 ; GO TO COMMON BEEP HANDLER
1315 0000168B E901FDFFFF        <1>     jmp    K27                 ; EXIT
1316                            <1>
1317                            <1> SHIP_IT:
1318                            <1>     ;--------------------------------------------------------------------------------
1319                            <1>     ; SHIP_IT
1320                            <1>     ;     THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES
1321                            <1>     ;     TO THE KEYBOARD CONTROLLER.
1322                            <1>     ;--------------------------------------------------------------------------------
1323                            <1>     ;
1324 00001690 6650              <1>     push  ax                   ; SAVE DATA TO SEND
1325                            <1>
1326                            <1>     ;----- WAIT FOR COMMAND TO ACCEPTED
1327 00001692 FA                <1>     cli                        ; DISABLE INTERRUPTS TILL DATA SENT
1328                            <1>     ; xor  ecx, ecx            ; CLEAR TIMEOUT COUNTER
1329 00001693 B900000100        <1>     mov    ecx, 10000h
1330                            <1> S10:
1331 00001698 E464              <1>     in     al, STATUS_PORT          ; READ KEYBOARD CONTROLLER STATUS
1332 0000169A A802              <1>     test   al, INPT_BUF_FULL   ; CHECK FOR ITS INPUT BUFFER BUSY
1333 0000169C E0FA              <1>     loopnz S10                 ; WAIT FOR COMMAND TO BE ACCEPTED
1334                            <1>
1335 0000169E 6658              <1>     pop    ax                  ; GET DATA TO SEND
1336 000016A0 E664              <1>     out    STATUS_PORT, al          ; SEND TO KEYBOARD CONTROLLER
1337 000016A2 FB                <1>     sti                        ; ENABLE INTERRUPTS AGAIN
1338 000016A3 C3                <1>     retn                       ; RETURN TO CALLER
1339                            <1>
1340                            <1> SND_DATA:
1341                            <1>     ; --------------------------------------------------------------------------------
1342                            <1>     ; SND_DATA
1343                            <1>     ;     THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES
1344                            <1>     ;     TO THE KEYBOARD AND RECEIPT OF ACKNOWLEDGEMENTS. IT ALSO
1345                            <1>     ;     HANDLES ANY RETRIES IF REQUIRED
1346                            <1>     ; --------------------------------------------------------------------------------
1347                            <1>     ;
1348 000016A4 6650              <1>     push   ax                  ; SAVE REGISTERS
1349 000016A6 6653              <1>     push   bx
1350 000016A8 51                <1>     push   ecx
1351 000016A9 88C7              <1>     mov    bh, al              ; SAVE TRANSMITTED BYTE FOR RETRIES
1352 000016AB B303              <1>     mov    bl, 3               ; LOAD RETRY COUNT
1353                            <1> SD0:
1354 000016AD FA                <1>     cli                        ; DISABLE INTERRUPTS
1355 000016AE 8025[976F0000]CF  <1>     and    byte [KB_FLAG_2], ~(KB_FE+KB_FA) ; CLEAR ACK AND RESEND FLAGS
1356                            <1>     ;
1357                            <1>     ;----- WAIT FOR COMMAND TO BE ACCEPTED
1358 000016B5 B900000100        <1>     mov    ecx, 10000h         ; MAXIMUM WAIT COUNT
1359                            <1> SD5:
1360 000016BA E464              <1>     in     al, STATUS_PORT           ; READ KEYBOARD PROCESSOR STATUS PORT
1361 000016BC A802              <1>     test   al, INPT_BUF_FULL   ; CHECK FOR ANY PENDING COMMAND
1362 000016BE E0FA              <1>     loopnz SD5                 ; WAIT FOR COMMAND TO BE ACCEPTED
1363                            <1>     ;
1364 000016C0 88F8              <1>     mov    al, bh              ; REESTABLISH BYTE TO TRANSMIT
1365 000016C2 E660              <1>     out    PORT_A, al          ; SEND BYTE
1366 000016C4 FB                <1>     sti                        ; ENABLE INTERRUPTS
1367                            <1>     ;mov   cx, 01A00h          ; LOAD COUNT FOR 10 ms+
1368 000016C5 B9FFFF0000        <1>     mov    ecx, 0FFFFh
1369                            <1> SD1:
1370 000016CA F605[976F0000]30  <1>     test   byte [KB_FLAG_2], KB_FE+KB_FA ; SEE IF EITHER BIT SET
1371 000016D1 750F              <1>     jnz    short SD3           ; IF SET, SOMETHING RECEIVED GO PROCESS
1372 000016D3 E2F5              <1>     loop   SD1                 ; OTHERWISE WAIT
1373                            <1> SD2:
1374 000016D5 FECB              <1>     dec    bl                  ; DECREMENT RETRY COUNT
1375 000016D7 75D4              <1>     jnz    short SD0           ; RETRY TRANSMISSION
1376 000016D9 800D[976F0000]80  <1>     or     byte [KB_FLAG_2], KB_ERR ; TURN ON TRANSMIT ERROR FLAG
1377 000016E0 EB09              <1>     jmp    short SD4           ; RETRIES EXHAUSTED FORGET TRANSMISSION
1378                            <1> SD3:
1379 000016E2 F605[976F0000]10  <1>     test   byte [KB_FLAG_2], KB_FA ; SEE IF THIS IS AN ACKNOWLEDGE
1380 000016E9 74EA              <1>     jz     short SD2           ; IF NOT, GO RESEND
1381                            <1> SD4:
1382 000016EB 59                <1>     pop    ecx                 ; RESTORE REGISTERS
1383 000016EC 665B              <1>     pop    bx
1384 000016EE 6658              <1>     pop    ax
1385 000016F0 C3                <1>     retn                       ; RETURN, GOOD TRANSMISSION
1386                            <1>
1387                            <1> SND_LED:
1388                            <1>     ; --------------------------------------------------------------------------------
1389                            <1>     ; SND_LED
1390                            <1>     ;     THIS ROUTINES TURNS ON THE MODE INDICATORS.
1391                            <1>     ;
1392                            <1>     ;--------------------------------------------------------------------------------
1393                            <1>     ;
1394 000016F1 FA                <1>     cli                        ; TURN OFF INTERRUPTS
1395 000016F2 F605[976F0000]40  <1>     test   byte [KB_FLAG_2], KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
1396 000016F9 755F              <1>     jnz    short SL1           ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
1397                            <1>     ;
1398 000016FB 800D[976F0000]40  <1>     or     byte [KB_FLAG_2], KB_PR_LED ; TURN ON UPDATE IN PROCESS
1399 00001702 B020              <1>     mov    al, EOI                  ; END OF INTERRUPT COMMAND
1400 00001704 E620              <1>     out    20h, al ;out INTA00, al   ; SEND COMMAND TO INTERRUPT CONTROL PORT
1401 00001706 EB11              <1>     jmp    short SL0           ; GO SEND MODE INDICATOR COMMAND
1402                            <1> SND_LED1:
1403 00001708 FA                <1>     cli                        ; TURN OFF INTERRUPTS
1404 00001709 F605[976F0000]40  <1>     test   byte [KB_FLAG_2], KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
1405 00001710 7548              <1>     jnz    short SL1           ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
1406                            <1>     ;
1407 00001712 800D[976F0000]40  <1>     or     byte [KB_FLAG_2], KB_PR_LED ; TURN ON UPDATE IN PROCESS
1408                            <1> SL0:
1409 00001719 B0ED              <1>     mov    al, LED_CMD         ; LED CMD BYTE
1410 0000171B E884FFFFFF        <1>     call   SND_DATA            ; SEND DATA TO KEYBOARD
1411 00001720 FA                <1>     cli
1412 00001721 E836000000        <1>     call   MAKE_LED            ; GO FORM INDICATOR DATA BYTE
```

```
1413 00001726 8025[976F0000]F8    <1>       and   byte [KB_FLAG_2], 0F8h    ; ~KB_LEDS ; CLEAR MODE INDICATOR BITS
1414 0000172D 0805[976F0000]      <1>       or    [KB_FLAG_2], al     ; SAVE PRESENT INDICATORS FOR NEXT TIME
1415 00001733 F605[976F0000]80    <1>       test  byte [KB_FLAG_2], KB_ERR ; TRANSMIT ERROR DETECTED
1416 0000173A 750F                <1>       jnz   short SL2           ; IF SO, BYPASS SECOND BYTE TRANSMISSION
1417                              <1>       ;
1418 0000173C E863FFFFFF          <1>       call  SND_DATA            ; SEND DATA TO KEYBOARD
1419 00001741 FA                  <1>       cli                       ; TURN OFF INTERRUPTS
1420 00001742 F605[976F0000]80    <1>       test  byte [KB_FLAG_2], KB_ERR ; TRANSMIT ERROR DETECTED
1421 00001749 7408                <1>       jz    short SL3           ; IF NOT, DON'T SEND AN ENABLE COMMAND
1422                              <1> SL2:
1423 0000174B B0F4                <1>       mov   al, KB_ENABLE       ; GET KEYBOARD CSA ENABLE COMMAND
1424 0000174D E852FFFFFF          <1>       call  SND_DATA            ; SEND DATA TO KEYBOARD
1425 00001752 FA                  <1>       cli                       ; TURN OFF INTERRUPTS
1426                              <1> SL3:
1427 00001753 8025[976F0000]3F    <1>       and   byte [KB_FLAG_2], ~(KB_PR_LED+KB_ERR) ; TURN OFF MODE INDICATOR
1428                              <1> SL1:                            ; UPDATE AND TRANSMIT ERROR FLAG
1429 0000175A FB                  <1>       sti                       ; ENABLE INTERRUPTS
1430 0000175B C3                  <1>       retn                      ; RETURN TO CALLER
1431                              <1>
1432                              <1> MAKE_LED:
1433                              <1>       ;--------------------------------------------------------------------------------
1434                              <1>       ; MAKE_LED
1435                              <1>       ;     THIS ROUTINES FORMS THE DATA BYTE NECESSARY TO TURN ON/OFF
1436                              <1>       ;     THE MODE INDICATORS.
1437                              <1>       ;--------------------------------------------------------------------------------
1438                              <1>       ;
1439                              <1>       ;push  cx                 ; SAVE CX
1440 0000175C A0[956F0000]        <1>       mov   al, [KB_FLAG]       ; GET CAPS & NUM LOCK INDICATORS
1441 00001761 2470                <1>       and   al, CAPS_STATE+NUM_STATE+SCROLL_STATE ; ISOLATE INDICATORS
1442                              <1>       ;mov   cl, 4              ; SHIFT COUNT
1443                              <1>       ;rol   al, cl             ; SHIFT BITS OVER TO TURN ON INDICATORS
1444 00001763 C0C004              <1>       rol   al, 4 ; 20/02/2015
1445 00001766 2407                <1>       and   al, 07h                   ; MAKE SURE ONLY MODE BITS ON
1446                              <1>       ;pop   cx
1447 00001768 C3                  <1>       retn                      ; RETURN TO CALLER
1448                              <1>
1449                              <1> ; % include 'kybdata.s'   ; KEYBOARD DATA
1450                              <1>
1451                              <1>
1452                              <1> ; /// End Of KEYBOARD FUNCTIONS ///
2653
2654                                  %include 'video.s' ; 07/03/2015
   1                              <1> ; ****************************************************************************
   2                              <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.3 - video.s
   3                              <1> ; ----------------------------------------------------------------------------
   4                              <1> ; Last Update: 12/02/2021
   5                              <1> ; ----------------------------------------------------------------------------
   6                              <1> ; Beginning: 16/01/2016
   7                              <1> ; ----------------------------------------------------------------------------
   8                              <1> ; Assembler: NASM version 2.15 (trdos386.s)
   9                              <1> ; ----------------------------------------------------------------------------
  10                              <1> ; Turkish Rational DOS
  11                              <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
  12                              <1> ;
  13                              <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
  14                              <1> ; video.inc (13/08/2015)
  15                              <1> ;
  16                              <1> ; Derived from 'IBM PC-AT' BIOS source code (1985)
  17                              <1> ; ****************************************************************************
  18                              <1>
  19                              <1> ; Retro UNIX 386 v1 Kernel - VIDEO.INC
  20                              <1> ; Last Modification: 13/08/2015
  21                              <1> ;             (Video Data is in 'VIDATA.INC')
  22                              <1> ;
  23                              <1> ; ///////// VIDEO (CGA) FUNCTIONS ///////////////
  24                              <1>
  25                              <1> ; 16/01/2016 (32 bit modifications, TRDOS386 - TRDOS v2.0, video.s)
  26                              <1> ; INT 31H (TRDOS 386) = INT 10H (IBM PC/AT REAL MODE)
  27                              <1>
  28                              <1> ; IBM PC-AT BIOS Source Code
  29                              <1> ; TITLE VIDEO1 --- 06/10/85 VIDEO DISPLAY BIOS
  30                              <1>
  31                              <1> _int10h:
  32                              <1>       ; 23/03/2016
  33                              <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
  34 00001769 9C                  <1>       pushfd
  35 0000176A 0E                  <1>       push  cs
  36 0000176B E851000000          <1>       call  VIDEO_IO_1
  37 00001770 C3                  <1>       retn
  38                              <1>
  39                              <1> ;--- INT 10 H ----------------------------------------------------------------
  40                              <1> ; VIDEO_IO                                                :
  41                              <1> ;     THESE ROUTINES PROVIDE THE CRT DISPLAY INTERFACE                   :
  42                              <1> ;     THE FOLLOWING FUNCTIONS ARE PROVIDED:                             :
  43                              <1> ;                                                                       :
  44                              <1> ;    (AH)= 00H     SET MODE (AL) CONTAINS MODE VALUE                            :
  45                              <1> ;           (AL) = 00H  40X25 BW MODE (POWER ON DEFAULT)                :
  46                              <1> ;           (AL) = 01H  40X25 COLOR                                     :
  47                              <1> ;           (AL) = 02H  80X25 BW                                        :
  48                              <1> ;           (AL) = 03H  80X25 COLOR                                     :
  49                              <1> ;                         GRAPHICS MODES                                :
  50                              <1> ;           (AL) = 04H  320X200 COLOR                                   :
  51                              <1> ;           (AL) = 05H  320X200 BW MODE                                 :
  52                              <1> ;           (AL) = 06H  640X200 BW MODE                                 :
  53                              <1> ;           (AL) = 07H   80X25 MONOCHROME (USED INTERNAL TO VIDEO ONLY)       :
  54                              <1> ;           *** NOTES -BW MODES OPERATE SAME AS COLOR MODES, BUT COLOR :
  55                              <1> ;                      BURST IS NOT ENABLED                             :
  56                              <1> ;                     -CURSOR IS NOT DISPLAYED IN GRAPHICS MODE         :
  57                              <1> ;    (AH)= 01H     SET CURSOR TYPE                                           :
  58                              <1> ;           (CH) = BITS 4-0 = START LINE FOR CURSOR                     :
  59                              <1> ;                  ** HARDWARE WILL ALWAYS CAUSE BLINK                  :
  60                              <1> ;                  ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING    :
  61                              <1> ;                     OR NO CURSOR AT ALL                               :
  62                              <1> ;           (CL) = BITS 4-0 = END LINE FOR CURSOR                       :
  63                              <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>
00001771 [221B0000]     <1> M1:   dd    SET_MODE     ; TABLE OF ROUTINES WITHIN VIDEO I/O
00001775 [EF1E0000]     <1>       dd    SET_CTYPE
00001779 [231F0000]     <1>       dd    SET_CPOS
0000177D [4B1F0000]     <1>       dd    READ_CURSOR
                        <1>       ;dd   VIDEO_RETURN ; READ_LPEN
00001781 [201B0000]     <1>       dd    set_mode_ncm ; Set mode without clearing video memory
00001785 [911F0000]     <1>       dd    ACT_DISP_PAGE
00001789 [23200000]     <1>       dd    SCROLL_UP
0000178D [4D210000]     <1>       dd    SCROLL_DOWN
00001791 [CD210000]     <1>       dd    READ_AC_CURRENT
00001795 [2A220000]     <1>       dd    WRITE_AC_CURRENT
00001799 [50220000]     <1>       dd    WRITE_C_CURRENT
0000179D [952B0000]     <1>       dd    SET_COLOR
000017A1 [002C0000]     <1>       dd    WRITE_DOT
000017A5 [CB2B0000]     <1>       dd    READ_DOT
000017A9 [E0220000]     <1>       dd    WRITE_TTY
000017AD [081B0000]     <1>       dd    VIDEO_STATE
000017B1 [94360000]     <1>       dd    vga_pal_funcs; 10/08/2016 (TRDOS 386)
000017B5 [0D310000]     <1>       dd    font_setup   ; 10/07/2016 (TRDOS 386)
000017B9 [571B0000]     <1>       dd    VIDEO_RETURN ; RESERVED
000017BD [5B240000]     <1>       dd    WRITE_STRING ; 23/06/2016 (TRDOS 386)
                        <1> M1L   EQU   $ - M1
                        <1>
                        <1> ; 06/12/2020
                        <1> ; 05/12/2020
                        <1> ; 03/12/2020
                        <1> ; 27/11/2020 - TRDOS 386 v2.0.3
                        <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
000017C1 FC             <1>       cld                       ; SET DIRECTION FORWARD
                        <1>
                        <1>       ;cmp   ah, M1L/4          ; TEST FOR WITHIN TABLE RANGE
                        <1>       ;jnb   short M4           ; BRANCH TO EXIT IF NOT A VALID COMMAND
                        <1>
                        <1>       ; 26/11/2020
000017C2 80FC14         <1>       cmp    ah, M1L/4
000017C5 7205           <1>       jb     short VGA_func
                        <1>
000017C7 80FC4F         <1>       cmp    ah, 4Fh
000017CA 7532           <1>       jne    short M4 ; invalid !
                        <1>
                        <1> VGA_func: ; 26/11/2020
000017CC 06             <1>       push   es ; *
000017CD 1E             <1>       push   ds ; **                   ; SAVE WORK AND PARAMETER REGISTERS
                        <1>
                        <1>       ; 26/11/2020
000017CE 50             <1>       push   eax ; -
                        <1>
000017CF 66B81000       <1>       mov    ax, KDATA          ; POINT DS: TO DATA SEGMENT
000017D3 8ED8           <1>       mov    ds, ax
000017D5 8EC0           <1>       mov    es, ax
                        <1>
                        <1>       ; 26/11/2020
000017D7 58             <1>       pop    eax ; +
                        <1>       ;
000017D8 FB             <1>       sti
000017D9 80FC4F         <1>       cmp    ah, 4Fh
000017DC 747A           <1>       je     short VBE_func
                        <1>
                        <1>       ; 04/12/2020
000017DE A3[48960100]   <1>       mov    [video_eax], eax
                        <1>
                        <1>       ; 21/12/2020
000017E3 803D[CA6F0000]FF <1>     cmp    byte [CRT_MODE], 0FFh     ; Current mode is a VESA VBE mode ?
000017EA 7213           <1>       jb     short VGA_func_std
                        <1>
000017EC 08E4           <1>       or     ah, ah             ; set mode ?
000017EE 750F           <1>       jnz    short VGA_func_std  ; no
                        <1>
000017F0 803D[54090000]03 <1>     cmp    byte [vbe3], 3            ; (real) VESA VBE 3 video bios ?
000017F7 7506           <1>       jne    short VGA_func_std  ; no
                        <1>
```

```
000017F9 E989010000          <1>       jmp    vesa_vbe3_pmi
                             <1>
                             <1>       ; 21/12/2020
                             <1> M4:                           ; COMMAND NOT VALID
000017FE CF                  <1>       iretd                   ; DO NOTHING IF NOT IN VALID RANGE
                             <1>
                             <1> VGA_func_std:
                             <1>       ; 06/12/2020
                             <1>       ; 03/12/2020
000017FF 80FC0F              <1>       cmp    ah, 0Fh
00001802 773D                <1>       ja     short VGA_funcs_0   ; only CGA funcs will be handled by
                             <1>       ; 06/12/2020              ; vga bios firmware
                             <1>       ; 03/12/2020
                             <1>       ;test  ah, 7Fh ; set mode ?
                             <1>       ;;or   ah, ah             ; only 'set mode' will be handled by
                             <1>       ;jnz   short VGA_funcs_0  ; vga bios firmware
                             <1>       ;jz    short vbe_pmi32_0
                             <1>
                             <1>       ; 28/11/2020
00001804 803D[1C120300]00    <1>       cmp    byte [pmi32], 0          ; 32 bit protected mode interface for
0000180B 7634                <1>       jna    short VGA_funcs_0   ; video hardware's vga bios firmware
                             <1>                                  ; ([pmi32] > 0 if it is activated)
                             <1>                                  ; note:
                             <1>                                  ; [vbe3] = 3 is required to activate
                             <1>       ; 07/12/2020
0000180D 20E4                <1>       and    ah, ah ; is this set mode ?
0000180F 7413                <1>       jz     short vbe_pmi32_2 ; yes
                             <1>
00001811 80FC04              <1>       cmp    ah, 04h       ; set mode ('no clear memory' option)
00001814 742B                <1>       je     short VGA_funcs_0
                             <1>
                             <1>       ; 07/12/2020
00001816 803D[CA6F0000]07    <1>       cmp    byte [CRT_MODE], 7 ; current mode > 7 ?
0000181D 7622                <1>       jna    short VGA_funcs_0  ; no
                             <1>
                             <1>       ; when mode 3 is active,
                             <1>       ; video bios functions are not redirected
                             <1>       ; to VESA VBE3 PMI except 'set mode' function
                             <1>
                             <1> vbe_pmi32_0:
                             <1>       ; 06/12/2020
                             <1>       ;or    ah, ah
                             <1>       ;jnz   short vbe_pmi32_2
                             <1>       ; ah = 0 ; 'set mode'
                             <1>       ;cmp   al, 3 ; 80x25 text mode, 16 colors, default mode for MainProg
                             <1>       ;jne   short vbe_pmi32_1
                             <1>       ;cmp   byte [CRT_MODE], 3 ; If current video mode <> 3 and requested
                             <1>       ;                         ; video mode is 3, use internal 'set mode';
                             <1>       ;                         ; otherwise, use vesa vbe 3 bios's 'set mode'.
                             <1>       ;jne   short VGA_funcs_0
                             <1> vbe_pmi32_1:
0000181F E963010000          <1>       jmp    vesa_vbe3_pmi
                             <1> ;vbe_pmi32_2:
                             <1>       ;cmp   ah, 04h ; set mode (no clear mem option)
                             <1>       ;jne   short vbe_pmi32_1
                             <1>
                             <1> vbe_pmi32_2:
                             <1>       ; 07/12/2020
00001824 803D[CA6F0000]07    <1>       cmp    byte [CRT_MODE], 7 ; current mode > 7 ?
0000182B 77F2                <1>       ja     short vbe_pmi32_1  ; yes
                             <1>
0000182D 3C07                <1>       cmp    al, 7  ; requested mode > 7 ?
0000182F 7610                <1>       jna    short VGA_funcs_0  ; no (CGA)
                             <1>
00001831 3C13                <1>       cmp    al, 13h
00001833 76EA                <1>       jna    short vbe_pmi32_1
                             <1>
00001835 A880                <1>       test   al, 80h
00001837 7408                <1>       jz     short VGA_funcs_0  ; unknown or special
                             <1>
00001839 3C87                <1>       cmp    al, 87h       ; requested mode > 7 ?
                             <1>                            ; (with no clear mem ops)
0000183B 7604                <1>       jna    short VGA_funcs_0  ; no (CGA)
                             <1>
0000183D 3C93                <1>       cmp    al, 93h
0000183F 76DE                <1>       jna    short vbe_pmi32_1
                             <1>
                             <1>       ; > 13h video modes are unknown or special
                             <1>       ; they must be handled by kernel
                             <1>
                             <1>       ; CGA video modes will be handled by kernel
                             <1>
                             <1> VGA_funcs_0:
00001841 52                  <1>       push   edx ; ***
00001842 51                  <1>       push   ecx ; ****
00001843 53                  <1>       push   ebx ; *****
00001844 56                  <1>       push   esi ; ******
00001845 57                  <1>       push   edi ; *******
00001846 55                  <1>       push   ebp ; ********
                             <1>
                             <1>       ;mov   [video_eax], eax ; 12/05/2016
00001847 BF00800B00          <1>       mov    edi, 0B8000h       ; GET offset FOR COLOR CARD
                             <1>
                             <1>       ; 23/03/2016
0000184C C0E402              <1>       shl    ah, 2  ; dword            ; TIMES 2 FOR WORD TABLE LOOKUP
0000184F 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
                             <1>       ;sti ; 26/11/2020
                             <1>       ;
                             <1>
```

```
00001852 FFA6[71170000]      <1>       JMP    dword [esi+M1]            ; GO TO SELECTED FUNCTION
                             <1>
                             <1> VBE_func:
                             <1>       ; 26/11/2020
                             <1>       ;sti
00001858 55                  <1>       push   ebp ; *** ; 27/11/2020
00001859 56                  <1>       push   esi ; ****
                             <1>
                             <1>       ; Note:
                             <1>       ; ebx, ecx, edx, edi, ebp registers
                             <1>       ; must be saved by VBE functions and
                             <1>       ; eax register must be set
                             <1>       ; (according to VBE 3 standard)
                             <1>       ; before return from this interrupt
                             <1>       ; (every function must restore and set
                             <1>       ; registers except esp, esi, es, ds)
                             <1>
0000185A 803D[54090000]02    <1>       cmp    byte [vbe3], 2
00001861 7741                <1>       ja     short VESA_VBE3_PMI_CALL ; VBE3 video bios ('PMID')
                             <1>       ;je    short VBE_func_0  ; Bochs/Qemu/VirtualBox emulator
00001863 726A                <1>       jb     short VBE_unknown ; invalid ([vbe3] = 0)
                             <1>
                             <1>       ;jmp   VESA_VBE3_PMI_CALL
                             <1>
                             <1> VBE_func_0:
                             <1>       ; Bochs/Plex86 VGAbios VBE extension
                             <1>       ; (TRDOS 386 v2.0.3 can use VBE graphics modes on emulators)
                             <1>       ; BOCHS/QEMU/VIRTUALBOX
                             <1>
00001865 8A25[55090000]      <1>       mov    ah, [vbe2bios]
0000186B 80FCC0              <1>       cmp    ah, 0C0h
0000186E 725F                <1>       jb     short VBE_unknown
00001870 80FCC5              <1>       cmp    ah, 0C5h
00001873 775A                <1>       ja     short VBE_unknown
                             <1>
                             <1>       ; TRDOS 386 is running on BOCHS or QEMU
00001875 B44F                <1>       mov    ah, 4Fh
                             <1> VBE_func_1:
00001877 0FB6F0              <1>       movzx  esi, al ; VESA VBE function number
0000187A 66C1E602            <1>       shl    si, 2 ; dword
0000187E 6683FE14            <1>       cmp    si, N1L
00001882 734B                <1>       jnb    short VBE_unknown
                             <1>       ;sti
                             <1>
00001884 FF96[90180000]      <1>       call   dword [esi+N1] ; call VBE function
                             <1>
                             <1>       ;jmp   short VBE_bios_return
                             <1>
                             <1> VBE_bios_return:
0000188A FA                  <1>       cli
0000188B 5E                  <1>       pop    esi ; ****
0000188C 5D                  <1>       pop    ebp ; *** ; 27/11/2020
0000188D 07                  <1>       pop    es  ; **
0000188E 1F                  <1>       pop    ds  ; *
0000188F CF                  <1>       iretd
                             <1>
                             <1>       ; 26/11/2020
                             <1> N1:
00001890 [F3180000]          <1>       dd     vbe_biosfn_return_ctrl_info
00001894 [203A0000]          <1>       dd     vbe_biosfn_return_mode_info
00001898 [DB3A0000]          <1>       dd     vbe_biosfn_set_mode
0000189C [AB3B0000]          <1>       dd     vbe_biosfn_return_current_mode
000018A0 [CA3B0000]          <1>       dd     vbe_biosfn_save_restore_state
                             <1>       ;dd    vbe_biosfn_display_window_ctrl
                             <1>       ;dd    vbe_biosfn_set_get_log_scanline
                             <1>       ;dd    vbe_biosfn_set_get_disp_start
                             <1>       ;dd    vbe_biosfn_set_get_dac_pal_frm
                             <1>       ;dd    vbe_biosfn_set_get_palette_data
                             <1>
                             <1> N1L   EQU    $ - N1
                             <1>
                             <1> VESA_VBE3_PMI_CALL: ; VESA VBE video bios (firmware) functions
                             <1>                     ; by using VESA VBE3 Protected Mode Interface
                             <1>
                             <1>       ; 29/11/2020
                             <1>       ; 26/11/2020 - TRDOS 386 v2.0.3 video.s
                             <1>
                             <1>       ; We are here because..
                             <1>       ; 'PMID' has been verified by TRDOS 386 v2.0.3 kernel.
                             <1>       ; (Otherwise bochs/plex86 compatible VBE functions and
                             <1>       ;  modes would be used on BOCHS/QEMU/VIRTUALBOX emulators
                             <1>       ;  or only standard/old VGA graphics modes would be used.)
                             <1>
                             <1>       ; (TRDOS 386 v2.0.3 can use VESA VBE graphics modes if
                             <1>       ;  the video bios is full compatible with VBE3 standard)
                             <1>
                             <1>       ; 29/11/2020
                             <1>
000018A4 0FB6F0              <1>       movzx  esi, al ; VESA VBE 3 function number
000018A7 66C1E602            <1>       shl    si, 2 ; dword
000018AB 6683FE14            <1>       cmp    si, P1L
000018AF 731E                <1>       jnb    short VBE_unknown
                             <1>       ;sti
                             <1>
000018B1 57                  <1>       push   edi ; *****
                             <1>
000018B2 FF96[BB180000]      <1>       call   dword [esi+P1] ; call VBE 3 function
                             <1>
000018B8 5F                  <1>       pop    edi ; *****
                             <1>
000018B9 EBCF                <1>       jmp    short VBE_bios_return
                             <1>
                             <1> P1:
000018BB [D5180000]          <1>       dd     vbe3_pmfn_return_ctrl_info
```

```
000018BF [F9180000]           <1>       dd    vbe3_pmfn_return_mode_info
000018C3 [36190000]           <1>       dd    vbe3_pmfn_set_mode
000018C7 [31190000]           <1>       dd    vbe3_pmfn_return_current_mode
000018CB [281A0000]           <1>       dd    vbe3_pmfn_save_restore_state
                              <1>       ;dd   vbe3_pmfn_display_window_ctrl
                              <1>       ;dd   vbe3_pmfn_set_get_log_scanline
                              <1>       ;dd   vbe3_pmfn_set_get_disp_start
                              <1>       ;dd   vbe3_pmfn_set_get_dac_pal_frm
                              <1>       ;dd   vbe3_pmfn_set_get_palette_data
                              <1>       ;dd   vbe3_pmfn_return_pmi ; invalid for TRDOS 386 v2
                              <1>       ;dd   vbe3_pmfn_set_get_pixel_clock
                              <1>
                              <1> P1L   EQU   $ - P1
                              <1>
                              <1> ;     ; 29/11/2020
                              <1> ;     mov   edi, VBE3MODEINFOBLOCK >> 4 ; / 16
                              <1> ;
                              <1> ;     cmp   al, 04h
                              <1> ;     jb    short vbe3_pm_f ; function: 4F00h to 4F03h
                              <1> ;     ja    short vbe3_pmi_f5B ; function: 4F05h to 4F0Bh
                              <1> ;
                              <1> ;     ; check buffer length (must be <= 2048 bytes)
                              <1> ;
                              <1> ;     and   dl, dl ; 0
                              <1> ;     jz    short vbe3_pm_f
                              <1> ;                 ; Return Save/Restore State buffer size
                              <1> ;
                              <1> ;     push  ebx  ; buffer address
                              <1> ;     push  edx  ; function: save (01h) or restore (02h)
                              <1> ;     call
                              <1> ;
                              <1> ;vbe3_pm_f03:
                              <1> ;     cmp   al, 2
                              <1> ;     ja    short vbe3_pm_f ; function 4F03h
                              <1> ;     jb    short vbe3_pm_f1
                              <1> ;
                              <1> ;
                              <1> ;vbe3_pm_f1:
                              <1> ;
                              <1> ;
                              <1> ;vbe3_pmi_f5B:
                              <1> ;     cmp   al, 09h
                              <1> ;     jna   short vbe3_pm_f ; funcs 05h to 09h are usable
                              <1> ;
                              <1> ;     cmp   al, 0Bh ; Get/Set pixel clock, last function
                              <1> ;     jne   short VBE_unknown
                              <1> ;                 ; (do not use 'uncertain' functions
                              <1> ;                 ; because of system-user buff transfers)
                              <1> ;vbe3_pm_f:
                              <1> ;     mov   byte [vbe3_pm_fn], al     ; set
                              <1> ;     ; prepare 16 bit pm segments & registers for pmi call
                              <1> ;     call  VESA_VBE3_PM_FUNCTION
                              <1> ;
                              <1> ;
                              <1>
                              <1>       ; 26/11/2020
                              <1> VBE_unknown:
000018CF 66B80001             <1>       mov   ax, 0100h  ; ah = 1 : Function call failed
                              <1>                        ; al = 0 : Function is not supported
                              <1>
000018D3 EBB5                 <1>       jmp   short VBE_bios_return
                              <1>
                              <1> vbe3_pmfn_return_ctrl_info:
                              <1>       ; 12/12/2020
                              <1>       ;
                              <1>       ; VBE function 4F00h - Return VBE Controller Information
                              <1>       ;
                              <1>       ; Input:
                              <1>       ;      EDI = Pointer to buffer in which to place
                              <1>       ;           VbeInfoBlock structure
                              <1>       ;
                              <1>       ;     AX = 4F00h
                              <1>       ; Output:
                              <1>       ;      AX = VBE return status
                              <1>       ;           AX = 004Fh -> succeeded
                              <1>       ;           AX <> 004Fh -> failed
                              <1>       ;
                              <1>       ; Modified registers: eax (+ edi for kernel's own call)
                              <1>
                              <1>       ; NOTE: TRDOS 386 v2 (v2.0.3) kernel calls this function
                              <1>       ; during startup while cpu is in real mode
                              <1>       ; (by using int 10h, 4F02h) and saves VbeInfoBlock at
                              <1>       ; VBE3INFOBLOCK address (97E00h for TRDOS 386 v2.0.3).
                              <1>       ;
                              <1>       ; So...
                              <1>       ; This VBE function is adjusted to return/move same info
                              <1>       ; from VBE3INFOBLOCK to user's buffer in EDI.
                              <1>
                              <1>       ; int 31h (int 10h) entrance
                              <1>
000018D5 21FF                 <1>       and   edi, edi
000018D7 7424                 <1>       jz    short vbe3_func_fail ; invalid buffer address !
                              <1>
                              <1> ;_vbe3_pmfn_return_ctrl_info:
                              <1>       ;or   edi, edi
                              <1>       ;jnz  short _vbe_biosfn_return_ctrl_info
                              <1>       ;
                              <1>       ;; this option may not be necessary - 12/12/2020
                              <1>       ;
                              <1>       ;; edi = 0, kernel forces to get ctrl info again
                              <1>       ;;       by using VESA VBE3 video bios's pmi
                              <1>       ;
                              <1>       ;;push edi
                              <1>       ;; far call to VESA VBE3 PMI
                              <1>       ;;mov  ax, 4F00h ; Return VBE Controller Info
```

```
                        <1>       ;mov   edi, VBE3INFOBLOCK-VBE3SAVERESTOREBLOCK
                        <1>       ;; ES selector base address = VBE3SAVERESTOREBLOCK
                        <1>       ;call  int10h_32bit_pmi
                        <1>       ;;pop  edi
                        <1>       ;mov   edi, VBE3INFOBLOCK ; retn to the kernel sub
                        <1>       ;cmp   ax, 004Fh
                        <1>       ;je    short vbe_ctrl_info_retn
                        <1>       ;stc
                        <1>       ;retn
                        <1>
                        <1> _vbe_biosfn_return_ctrl_info:
000018D9 57             <1>       push   edi
000018DA 51             <1>       push   ecx
000018DB BE007E0900     <1>       mov    esi, VBE3INFOBLOCK
000018E0 B900020000     <1>       mov    ecx, 512
000018E5 E808020100     <1>       call   transfer_to_user_buffer
000018EA 59             <1>       pop    ecx
000018EB 5F             <1>       pop    edi
000018EC 720F           <1>       jc     short vbe3_func_fail
                        <1>
000018EE 31C0           <1>       xor    eax, eax
000018F0 B04F           <1>       mov    al, 4Fh ; successful
                        <1> ;vbe_ctrl_info_retn:
000018F2 C3             <1>       retn
                        <1>
                        <1> vbe_biosfn_return_ctrl_info:
                        <1>       ; 12/12/2020
                        <1>       ;
                        <1>       ; VBE function 4F00h - Return VBE Controller Information
                        <1>       ;
                        <1>       ; Input:
                        <1>       ;      EDI = Pointer to buffer in which to place
                        <1>       ;           VbeInfoBlock structure
                        <1>       ;
                        <1>       ;      AX = 4F00h
                        <1>       ; Output:
                        <1>       ;      AX = VBE return status
                        <1>       ;           AX = 004Fh -> succeeded
                        <1>       ;           AX <> 004Fh -> failed
                        <1>       ;
                        <1>       ; Modified registers: eax
                        <1>
000018F3 21FF           <1>       and    edi, edi
000018F5 7406           <1>       jz     short vbe3_func_fail ; invalid buffer addr !
000018F7 EBE0           <1>       jmp    short _vbe_biosfn_return_ctrl_info
                        <1>
                        <1> vbe3_pmfn_return_mode_info:
                        <1>       ; 21/12/2020
                        <1>       ; 12/12/2020
                        <1>       ;
                        <1>       ; VBE function 4F01h - Return VBE Mode Information
                        <1>       ;
                        <1>       ; Input:
                        <1>       ;      CX = Mode number (VESA VBE mode number)
                        <1>       ;      EDI = Pointer to ModeInfoBlock structure
                        <1>       ;           (256 bytes) -User's buffer address-
                        <1>       ;      EDI = 0 -> kernel call
                        <1>       ;               (do not transfer ModeInfoBlock
                        <1>       ;               to user's buffer address)
                        <1>       ;      AX = 4F01h
                        <1>       ; Output:
                        <1>       ;      AX = VBE return status
                        <1>       ;           AX = 004Fh -> succeeded
                        <1>       ;           AX <> 004Fh -> failed
                        <1>       ;
                        <1>       ; Modified registers: eax, esi, edi
                        <1>
                        <1>       ; int 31h (int 10h) entrance
                        <1>
000018F9 09FF           <1>       or     edi, edi
000018FB 7506           <1>       jnz    short _vbe3_pmfn_return_mode_info
                        <1>
                        <1> vbe3_func_fail:
000018FD B84F010000     <1>       mov    eax, 014Fh ; ah = 1 : Function call failed
                        <1>                         ; al = 4Fh : Function is supported
00001902 C3             <1>       retn
                        <1>
                        <1>       ; jump from '_vbe_biosfn_return_mode_info'
                        <1> _vbe3_pmfn_return_mode_info:
00001903 57             <1>       push   edi
                        <1>
                        <1>       ;; clear vbe3 'mode info block' buffer
                        <1>       ;push  ecx
                        <1>       ;xor   eax, eax
                        <1>       ;mov   ecx, 256/4
                        <1>       ;mov   edi, VBE3MODEINFOBLOCK
                        <1>       ;rep   stosd
                        <1>       ;pop   ecx
                        <1>
                        <1>       ; far call to VESA VBE3 PMI
                        <1>       ;mov   ax, 4F01h ; Return VBE Mode Information
00001904 BF00060000     <1>       mov    edi, VBE3MODEINFOBLOCK-VBE3SAVERESTOREBLOCK
                        <1>       ; ES selector base address = VBE3SAVERESTOREBLOCK
00001909 E8FC000000     <1>       call   int10h_32bit_pmi
                        <1>
0000190E 5F             <1>       pop    edi
                        <1>
                        <1>       ;cmp   ax, 004Fh
                        <1>       ;jne   short vbe3_func_retn  ; failed !
                        <1>
0000190F 21FF           <1>       and    edi, edi
                        <1>       ;jz    short vbe3_func_success
                        <1>       ; 21/12/2020
00001911 741D           <1>       jz     short vbe3_func_retn
                        <1>
```

```
00001913 6683F84F            <1>       cmp    ax, 004Fh
00001917 7517                <1>       jne    short vbe3_func_retn  ; failed !
                             <1>
00001919 51                  <1>       push   ecx
0000191A BE007C0900          <1>       mov    esi, VBE3MODEINFOBLOCK
0000191F B900010000          <1>       mov    ecx, 256
00001924 E8C9010100          <1>       call   transfer_to_user_buffer
00001929 59                  <1>       pop    ecx
0000192A 72D1                <1>       jc     short vbe3_func_fail
                             <1>
0000192C 31C0                <1>       xor    eax, eax
0000192E B04F                <1>       mov    al, 4Fh ; successful
                             <1> vbe3_func_success:
                             <1> vbe3_func_retn:
00001930 C3                  <1>       retn
                             <1>
                             <1> vbe3_pmfn_return_current_mode:
                             <1>       ; 12/12/2020
                             <1>       ;
                             <1>       ; VBE function 4F03h - Return Current VBE Mode
                             <1>       ;
                             <1>       ; Input:
                             <1>       ;      none (AX = 4F03h)
                             <1>       ; Output:
                             <1>       ;      AX = VBE return status
                             <1>       ;           AX = 004Fh -> succeeded
                             <1>       ;           AX <> 004Fh -> failed
                             <1>       ;      BX = Current VBE mode
                             <1>       ;        bit 0-13 = Mode number
                             <1>       ;        bit 14 = 0 Windowed frame buffer model
                             <1>       ;                 = 1 Linear frame buffer model
                             <1>       ;        bit 15
                             <1>       ;            = 0 Memory cleared at last mode set
                             <1>       ;            = 1 Memory not cleared at last mode set
                             <1>       ;
                             <1>       ; Modified registers: eax, ebx
                             <1>
                             <1>       ; int 31h (int 10h) entrance
                             <1>
                             <1>       ; far call to VESA VBE3 PMI
                             <1>
                             <1>       ;mov   eax, 4F03h ; Return Current VBE Mode
                             <1> vbe3_pmfn_far_call:
                             <1>       ; ES selector base address = VBE3SAVERESTOREBLOCK
                             <1>       ;call  int10h_32bit_pmi
                             <1>       ;retn
00001931 E9D4000000          <1>       jmp    int10h_32bit_pmi
                             <1>
                             <1> vbe3_pmfn_set_mode:
                             <1>       ; 22/12/2020
                             <1>       ; 21/12/2020
                             <1>       ; 12/12/2020
                             <1>       ;
                             <1>       ; VBE function 4F02h - Set VBE Mode
                             <1>       ;
                             <1>       ; Input:
                             <1>       ;      BX = Desired Mode to set
                             <1>       ;        bit 0-13 = Mode number
                             <1>       ;        bit 14 = 0 Windowed frame buffer model
                             <1>       ;                 = 1 Linear frame buffer model
                             <1>       ;        bit 15
                             <1>       ;            = 0 Memory cleared at last mode set
                             <1>       ;            = 1 Memory not cleared at last mode set
                             <1>       ; Output:
                             <1>       ;      AX = VBE return status
                             <1>       ;           AX = 004Fh -> succeeded
                             <1>       ;           AX <> 004Fh -> failed
                             <1>       ;
                             <1>       ; Modified registers: eax, ebx, esi (21/12/2020)
                             <1>
                             <1>       ; int 31h (int 10h) entrance
                             <1>
                             <1>       ; 22/12/2020 (VESA VBE3 feature)
                             <1>       ; BX bit 11 is flag for
                             <1>       ;      user specified CRTC values for refresh rate
                             <1>       ; 'test bh, 8'
                             <1>       ; if bit 11 is set, EDI points to 'CRTCInfoBlock'
                             <1>
                             <1>       ; 22/12/2020
                             <1>       ;; test bx for VBE video mode
                             <1>       ;test  bh, 1
                             <1>       ;jnz   short vbe3_sm_0
                             <1>
                             <1>       ;; use internal VBE mode set procedure
                             <1>       ;; for non-vbe (std VGA/CGA) modes
                             <1>       ;
                             <1>       ;; (it is useful -as 4F02h function-
                             <1>       ;;  to jump 'vbe_biosfn_set_mode'
                             <1>       ;;  instead of direct jump to '_set_mode')
                             <1>       ;; ((eliminates additional push-pops and settings))
                             <1>
                             <1>       ;jmp   vbe_biosfn_set_mode
                             <1>
                             <1> vbe3_sm_0:
                             <1>       ;;push ds  ; *
                             <1>       ;;push es  ; **
                             <1>       ;;push ebp ; ***
                             <1>       ;;push esi ; ****
                             <1>
                             <1>       ; Fit bx to VESA VBE2 type mode setting
                             <1>       ; (bx bit 11 is used for custom CRTC values in VBE3)
                             <1>       ; clear bit 9 to 11 (clear bh bit 1 to bit 3)
                             <1>
                             <1>       ; 22/12/2020
00001936 57                  <1>       push   edi ; *****
```

```
00001937 F6C708             <1>       test   bh, 8  ; Use user specified CRTC values
0000193A 7530               <1>       jnz    short vbe3_sm_3  ; for refresh rate
                            <1> vbe3_sm_4:
0000193C 80E7C1             <1>       and    bh, 0C1h ; use bit 15, 14, 8 only (for bh)
                            <1>       ;mov   [vbe_mode_x], bh
                            <1>
0000193F 803D[CA6F0000]03   <1>       cmp    byte [CRT_MODE], 3 ; is current mode 03h ?
00001946 7509               <1>       jne    short vbe3_sm_1    ; no
                            <1>
                            <1>       ; save mode 03h video pages and cursor positions
00001948 57                 <1>       push   edi ; **!***
00001949 51                 <1>       push   ecx ; ******
                            <1>       ;push  esi
                            <1>
0000194A E8CE040000         <1>       call   save_mode3_multiscreen
                            <1>
                            <1>       ;pop   esi
0000194F 59                 <1>       pop    ecx ; ******
00001950 5F                 <1>       pop    edi ; **!***
                            <1> vbe3_sm_1:
                            <1>       ; ax = 4F02h
                            <1>       ; bx = video mode number (vbe2 type)
00001951 E8B4000000         <1>       call   int10h_32bit_pmi
                            <1>                     ; call to far call to VBE3 PMI
                            <1>
00001956 6683F84F           <1>       cmp    ax, 004Fh ; succeeded ?
0000195A 750E               <1>       jne    short vbe3_sm_2
                            <1>       ; set current mode byte/sign to extended (SVGA) mode
0000195C C605[CA6F0000]FF   <1>       mov    byte [CRT_MODE], 0FFh ; VESA VBE mode
                            <1>       ; set current VBE mode word to bx input
00001963 66891D[1E120300]   <1>       mov    [video_mode], bx
                            <1> vbe3_sm_2:
                            <1>       ; 22/12/2020
0000196A 5F                 <1>       pop    edi ; *****
0000196B C3                 <1>       retn
                            <1>
                            <1> vbe3_sm_3:
                            <1>       ; 22/12/2020
                            <1>       ; copy user's CRTCInfoBlock to the buffer
0000196C 51                 <1>       push   ecx
0000196D 89FE               <1>       mov    esi, edi
0000196F BF807D0900         <1>       mov    edi, VBE3CRTCINFOBLOCK
00001974 B940000000         <1>       mov    ecx, 64
00001979 E8BE010100         <1>       call   transfer_from_user_buffer
0000197E 59                 <1>       pop    ecx
                            <1>       ; set offset (es base addr is VBE3SAVERESTOREBLOCK)
0000197F 81EF00760900       <1>       sub    edi, VBE3SAVERESTOREBLOCK
00001985 EBB5               <1>       jmp    short vbe3_sm_4
                            <1>
                            <1> vesa_vbe3_pmi:
                            <1>       ; 12/12/2020
                            <1>       ; 08/12/2020
                            <1>       ; 07/12/2020
                            <1>       ; 05/12/2020, 06/12/2020
                            <1>       ; 03/12/2020, 04/12/2020
                            <1>       ; 28/11/2020 (TRDOS 386 v2.0.3)
                            <1>       ; VGA BIOS functions via
                            <1>       ; VESA VBE3 Protected Mode Inface
                            <1>       ; [vbe3] = 3 and [pmi32] > 0
                            <1>
                            <1>       ; 04/12/2020
                            <1>       ; Only 'set mode' will be redirected to vbe3 video bios
                            <1>       ; (by setting mode 3 multiscreen paraters before and after)
                            <1>
                            <1>       ; 06/12/2020
00001987 20E4               <1>       and    ah, ah ; 0 = set mode function
00001989 7402               <1>       jz     short vbe3_pmi_0
0000198B EB76               <1>       jmp    vbe3_pmi_9
                            <1>
                            <1> vbe3_pmi_0:
                            <1>       ; 07/12/2020
0000198D 88C4               <1>       mov    ah, al
0000198F 80E480             <1>       and    ah, 80h ; 0 or 80h
00001992 30E0               <1>       xor    al, ah ; 8?h -> 0?h
                            <1>
                            <1>       ;cmp   al, 13h ; mode number above 13h is returned
                            <1>       ;jna   short vbe3_pmi_1
                            <1>       ;             ; back to default code due to uncertainty
                            <1>       ;             ; (>13h is not std for all svga bioses)
                            <1>       ;jmp   VGA_funcs_0
                            <1> vbe3_pmi_1:
                            <1>       ; 07/12/2020
                            <1>       ; Possible cases for VBE3 (PMI, ah=0) set mode:
                            <1>       ; current mode > 07h and requested mode: any
                            <1>       ; current mode <= 07h and requested mode > 07h
                            <1>
                            <1>       ; 06/12/2020
00001994 8825[57960100]     <1>       mov    byte [noclearmem], ah ; 0 or 80h
                            <1>       ; check current video mode if it is 03h
0000199A 803D[CA6F0000]03   <1>       cmp    byte [CRT_MODE], 3 ; current mode
000019A1 750B               <1>       jne    short vbe3_pmi_3
                            <1>       ; 07/12/2020
                            <1>       ; check new video mode if it is 03h also
                            <1>       ;cmp   al, 3
                            <1>       ;jne   short vbe3_pmi_2
                            <1>       ;mov   byte [p_crt_mode], 80h    ; clear video memory
                            <1>       ;jmp   short vbe3_pmi_5
                            <1> vbe3_pmi_2:
                            <1>       ; case 1:
                            <1>       ; Current mode is 03h and new mode is not 03h
                            <1>
                            <1>       ; save video pages and cursor positions
000019A3 56                 <1>       push   esi
000019A4 57                 <1>       push   edi
000019A5 51                 <1>       push   ecx
```

```
                                <1>
                                <1>       ; 12/12/2020
                                <1>       ;mov   esi, 0B8000h ; mode 3 video memory
                                <1>       ;mov   edi, 98000h  ; backup location
                                <1>       ;mov   ecx, (0B8000h-0B0000h)/4
                                <1>       ;rep   movsd
                                <1>       ;
                                <1>       ;mov   byte [p_crt_mode], 3 ; previous mode, backup sign
                                <1>       ;xchg  cl, [ACTIVE_PAGE]
                                <1>       ;mov   [p_crt_page], cl  ; save as previous active page
                                <1>       ;
                                <1>       ;; save cursor positions
                                <1>       ;mov   esi, CURSOR_POSN
                                <1>       ;mov   edi, cursor_pposn ; cursor positions backup
                                <1>       ;mov   cl, 4
                                <1>       ;rep   movsd
                                <1>
                                <1>       ; 12/12/2020
000019A6 E872040000             <1>       call   save_mode3_multiscreen
                                <1>
000019AB 59                     <1>       pop    ecx
000019AC 5F                     <1>       pop    edi
000019AD 5E                     <1>       pop    esi
                                <1> vbe3_pmi_3:
                                <1>       ; 08/12/2020
                                <1>       ; 07/12/2020
                                <1>       ; case 3 or case 4
000019AE A2[CA6F0000]           <1>       mov    [CRT_MODE], al
000019B3 3C03                   <1>       cmp    al, 3
000019B5 7407                   <1>       je     short vbe3_pmi_4
                                <1>       ; case 4:
                                <1>       ; Current mode is not 03h and also new mode is not 03h
000019B7 800D[55960100]80       <1>       or     byte [p_crt_mode], 80h  ; 83h (case 1 -> case 4)
                                <1>       ;jmp   short vbe3_pmi_5
                                <1> vbe3_pmi_4:
                                <1>       ; case 3:
                                <1>       ;
                                <1>       ; Current mode is not 03h and new mode is 03h
                                <1>
                                <1> ;vbe3_pmi_5:
                                <1>       ;mov   [CRT_MODE], al
                                <1>
000019BE E847000000             <1>       call   int10h_32bit_pmi
                                <1>
000019C3 803D[CA6F0000]03       <1>       cmp    byte [CRT_MODE], 3  ; new video mode
                                <1>       ;jne   vbe3_pmi_8   ; video mode <> 03h
000019CA 7532                   <1>       jne    short vbe3_pmi_8
                                <1>
                                <1>       ;push  eax ; 04/12/2020
000019CC 53                     <1>       push   ebx
000019CD 51                     <1>       push   ecx
000019CE 52                     <1>       push   edx
000019CF 57                     <1>       push   edi ; 03/12/2020
                                <1>
                                <1>       ; 12/12/2020
000019D0 56                     <1>       push   esi
000019D1 E87A040000             <1>       call   restore_mode3_multiscreen
000019D6 5E                     <1>       pop    esi
                                <1>       ; AL = active video page
                                <1>
                                <1>       ; 12/12/2020
                                <1>       ;mov   al, [p_crt_page] ; previous mode 3 active page
                                <1>       ;
                                <1>       ;;test byte [p_crt_mode], 7Fh ; 83h or 80h or 03h
                                <1>       ;;jz   short vbe3_pmi_6 ; do not restore video pages
                                <1>       ;                   ; clear current video page
                                <1>       ;; case 3
                                <1>       ;
                                <1>       ;; ([p_crt_mode] = 03h)
                                <1>       ;
                                <1>       ;; New video mode is 3 while current video mode is not 3
                                <1>       ;; (multi screen) video pages will be restored from 098000h
                                <1>       ;
                                <1>       ;; restore video pages and cursor positions
                                <1>       ;
                                <1>       ;mov   [ACTIVE_PAGE], al ; current mode 3 active page
                                <1>       ;
                                <1>       ;push  esi
                                <1>       ;
                                <1>       ;; restore video pages
                                <1>       ;mov   esi, 98000h
                                <1>       ;mov   edi, 0B8000h
                                <1>       ;;mov  ecx, 2000h
                                <1>       ;mov   cx, 2000h ; 8K dwords (32K)
                                <1>       ;rep   movsd
                                <1>       ;
                                <1>       ;mov   [p_crt_mode], cl ; reset ('case 3' end condition)
                                <1>       ;
                                <1>       ;; restore cursor positions
                                <1>       ;mov   esi, cursor_pposn
                                <1>       ;mov   edi, CURSOR_POSN
                                <1>       ;;mov  ecx, 4 ; restore all cursor positions (16 bytes)
                                <1>       ;mov   cl, 4
                                <1>       ;rep   movsd
                                <1>       ;
                                <1>       ;pop   esi
                                <1>       ;
                                <1>       ;; 07/12/2020
                                <1>       ;; restore CRT_START according to ACTIVE_PAGE
                                <1>       ;mov   [CRT_START], cx ; 0
                                <1>       ;
                                <1>       ;; check active page and set it again if it is not 0
                                <1>       ;or    al, al
                                <1>       ;jz    short vbe3_pmi_7
                                <1>       ;
```

```
1009                                    <1>       ;mov   cl, al
1010                                    <1> ;vbe3_pmi_5:
1011                                    <1>       ;add   word [CRT_START], 4096
1012                                    <1>       ;dec   cl
1013                                    <1>       ;jnz   short vbe3_pmi_5
1014                                    <1>
1015 000019D7 B405                      <1>       mov    ah, 05h ; set current video page
1016                                    <1>       ;al = video page
1017 000019D9 E82C000000                <1>       call   int10h_32bit_pmi
1018                                    <1>
1019                                    <1>       ; check current cursor position & set it again if not 0,0
1020                                    <1>       ;movzx ebx, byte [ACTIVE_PAGE]
1021 000019DE 0FB6D8                    <1>       movzx  ebx, al
1022 000019E1 D0E3                      <1>       shl    bl, 1
1023 000019E3 81C3[DE890100]            <1>       add    ebx, CURSOR_POSN
1024 000019E9 668B13                    <1>       mov    dx, [ebx]
1025 000019EC 6621D2                    <1>       and    dx, dx
1026 000019EF 7409                      <1>       jz     short vbe3_pmi_7
1027                                    <1>
1028                                    <1>       ;dx = cursor position (dl = column, dh = row)
1029                                    <1>       ;mov   bh, [ACTIVE_PAGE] ; 06/12/2020
1030 000019F1 88C7                      <1>       mov    bh, al
1031 000019F3 B402                      <1>       mov    ah, 02h ; set cursor position
1032 000019F5 E810000000                <1>       call   int10h_32bit_pmi
1033                                    <1>
1034                                    <1>       ;jmp   short vbe3_pmi_7
1035                                    <1>
1036                                    <1> ;vbe3_pmi_6:
1037                                    <1> ;     ; 07/12/2020
1038                                    <1> ;     ; case 1, previous mode is 03h, current mode is 03h
1039                                    <1> ;     ; 03/12/2020
1040                                    <1> ;     cmp    byte [noclearmem], 0
1041                                    <1> ;     jna    short vbe3_pmi_7 ; do not clear memory
1042                                    <1> ;     ; clear video page
1043                                    <1> ;     mov    ecx, 1024 ; 4096/4
1044                                    <1> ;     mov    eax, 07200720h
1045                                    <1> ;     mov    edi, 0B8000h ; [crt_base]
1046                                    <1> ;     add    di, [CRT_START]
1047                                    <1> ;     rep    stosd  ; FILL THE REGEN BUFFER WITH BLANKS
1048                                    <1>
1049                                    <1> vbe3_pmi_7:
1050 000019FA 5F                        <1>       pop    edi
1051 000019FB 5A                        <1>       pop    edx
1052 000019FC 59                        <1>       pop    ecx
1053 000019FD 5B                        <1>       pop    ebx
1054                                    <1>       ;pop   eax ; 04/12/2020
1055                                    <1> vbe3_pmi_8:
1056                                    <1>       ; 04/12/2020
1057                                    <1>       ;(TRDOS 386 v2.0.3, INT 31h, ah=0 return)
1058 000019FE 31C0                      <1>       xor    eax, eax  ; eax = 0 -> succesful
1059                                    <1> vesa_vbe3_pmi_retn:
1060 00001A00 07                        <1>       pop    es  ; **
1061 00001A01 1F                        <1>       pop    ds  ; *
1062 00001A02 CF                        <1>       iretd
1063                                    <1>
1064                                    <1> vbe3_pmi_9:
1065                                    <1>       ; 06/12/2020
1066                                    <1>       ;cmp   ah, 10h ; Set/Get Palette Registers
1067                                    <1>       ;jnb   short vbe3_pmi_10
1068                                    <1>       ; 05/12/2020
1069 00001A03 E802000000                <1>       call   int10h_32bit_pmi
1070 00001A08 EBF6                      <1>       jmp    short vesa_vbe3_pmi_retn
1071                                    <1>
1072                                    <1> ;vbe3_pmi_10:
1073                                    <1>       ; 06/12/2020
1074                                    <1>       ;jmp   VGA_funcs_0
1075                                    <1>
1076                                    <1> int10h_32bit_pmi:
1077                                    <1>       ; 03/12/2020
1078                                    <1>       ; 28/11/2020
1079                                    <1>       ; calling standard VGA Bios (INT 10h) functions
1080                                    <1>       ; by using 32 bit protected mode interface of
1081                                    <1>       ; VESA VBE3 Video Bios (with 'PMID' signature)
1082                                    <1>
1083                                    <1>       ; 03/12/2020
1084                                    <1>       ; eax, ebx, ecx, edx, edi will be used by vbios pmi
1085                                    <1>       ; (esi and ebp will not be used)
1086                                    <1>
1087                                    <1>       ; 03/12/2020
1088 00001A0A 56                        <1>       push   esi
1089 00001A0B C1E010                    <1>       shl    eax, 16  ; move function number (ax) to hw
1090 00001A0E 8B35[24120300]            <1>       mov    esi, [pmid_addr] ; linear address of
1091                                    <1>                               ; PMInfo.Entrypoint pointer
1092                                    <1>       ;mov   ax, [esi+PMInfo.EntryPoint]
1093 00001A14 668B06                    <1>       mov    ax, [esi]
1094 00001A17 C1C010                    <1>       rol    eax, 16 ; move PM entry address to hw
1095                                    <1>                      ; and move function number to lw (ax)
1096 00001A1A 5E                        <1>       pop    esi
1097                                    <1>
1098                                    <1>       ; top of stack: ; (*)
1099                                    <1>       ; return (the caller) address of "int10h_32bit_pmi"
1100                                    <1>
1101 00001A1B E97CEDFFFF                <1>       jmp    _VBE3PMI_fcall ; will return to the caller (*)
1102                                    <1>
1103                                    <1> _vbe3_pmfn_srs_8:
1104                                    <1>       ; 17/01/2021
1105 00001A20 31DB                      <1>       xor    ebx, ebx ; points to VBE3SAVERESTOREBLOCK
1106                                    <1> _vbe3_pmfn_srs_9: ; 24/01/2021
1107 00001A22 66B8044F                  <1>       mov    ax, 4F04h
1108 00001A26 EBE2                      <1>       jmp    short int10h_32bit_pmi
1109                                    <1>
1110                                    <1> vbe3_pmfn_save_restore_state:
1111                                    <1>       ; 24/01/2021
1112                                    <1>       ; 23/01/2021
1113                                    <1>       ; 16/01/2021, 17/01/2021
```

```
1114                                    <1>       ; 14/01/2021
1115                                    <1>       ;
1116                                    <1>       ; VBE function 4F04h - Save/Restore Video State
1117                                    <1>       ;
1118                                    <1>       ; Input:
1119                                    <1>       ;      DL = sub function
1120                                    <1>       ;      CL = requested state
1121                                    <1>       ;       EBX = pointer to buffer (if DL<>00h)
1122                                    <1>       ;      AX = 4F04h
1123                                    <1>       ; Output:
1124                                    <1>       ;      AX = 004Fh (successful)
1125                                    <1>       ;      AH > 0 -> error
1126                                    <1>       ;      BX = Number of 64-byte blocks
1127                                    <1>       ;           to hold the state buffer (if DL=00h)
1128                                    <1>
1129                                    <1>       ; Modified registers: eax, ebx, esi, edi
1130                                    <1>
1131 00001A28 21DB                     <1>       and    ebx, ebx ; user's buffer address
1132 00001A2A 750A                     <1>       jnz    short _vbe3_pmfn_save_restore_state
1133                                    <1>
1134 00001A2C 08D2                     <1>       or     dl, dl
1135 00001A2E 740C                     <1>       jz     short _vbe3_pmfn_srs_0
1136                                    <1>
1137                                    <1>       ; function failed
1138                                    <1>       ;;mov  eax, 0100h
1139                                    <1>       ;sub   eax, eax
1140                                    <1>       ;inc   ah  ; eax = 0100h
1141                                    <1>       ;retn
1142                                    <1>       ; 16/01/2021
1143                                    <1> _vbe3_pmfn_srs_fail:
1144 00001A30 B84F010000               <1>       mov    eax, 014Fh ; ah = 1 : Function call failed
1145                                    <1>                         ; al = 4Fh : Function is supported
1146                                    <1> _vbe3_srs_retn:
1147 00001A35 C3                       <1>       retn
1148                                    <1>
1149                                    <1> _vbe3_pmfn_save_restore_state:
1150 00001A36 20D2                     <1>       and    dl, dl
1151 00001A38 7559                     <1>       jnz    short _vbe3_pmfn_srs_2
1152                                    <1> _vbe3_pmfn_srs:
1153 00001A3A 31DB                     <1>       xor    ebx, ebx
1154                                    <1> _vbe3_pmfn_srs_0:
1155                                    <1>       ; 24/01/2021
1156 00001A3C 83F90F                   <1>       cmp    ecx, 0Fh
1157                                    <1>       ;ja    short _vbe3_pmfn_srs_1
1158 00001A3F 77EF                     <1>       ja     short _vbe3_pmfn_srs_fail
1159                                    <1>
1160                                    <1>       ; !!! CLEAR CL BIT 2 !!!
1161                                    <1>       ; (when bit 2 is set, function causes cpu exception)
1162                                    <1>       ; BIOS data will not be saved and restored
1163                                    <1>       ; (to prevent protected mode page fault error)
1164 00001A41 80E1FD                   <1>       and    cl, ~2 ; and cl, not 2
1165                                    <1>
1166                                    <1>       ; 24/01/2021
1167                                    <1>       ;mov   bl, 1
1168 00001A44 FEC3                     <1>       inc    bl ; = 1
1169 00001A46 66D3E3                   <1>       shl    bx, cl
1170 00001A49 66231D[28120300]         <1>       and    bx, [vbe3stbsflags]
1171 00001A50 7416                     <1>       jz     short _vbe3_pmfn_srs_1
1172                                    <1>       ;mov   bx, cx
1173 00001A52 89CB                     <1>       mov    ebx, ecx ; <= 15
1174 00001A54 D0E3                     <1>       shl    bl, 1 ; 0, 2, 8 .. 30
1175 00001A56 668B9B[7B3C0000]         <1>       mov    bx, [vbestatebufsize+ebx]
1176 00001A5D 89DF                     <1>       mov    edi, ebx
1177                                    <1>       ; edi = state buffer size in bytes
1178 00001A5F 66C1EB06                 <1>       shr    bx, 6 ; / 64
1179 00001A63 66B84F00                 <1>       mov    ax, 4Fh
1180 00001A67 C3                       <1>       retn
1181                                    <1> _vbe3_pmfn_srs_1:
1182                                    <1>       ; ax = 4F04h
1183                                    <1>       ;call  int10h_32bit_pmi
1184                                    <1>       ; 24/01/2021
1185                                    <1>       ;call  _vbe3_pmfn_srs_8
1186                                    <1>       ; ebx = 0
1187 00001A68 E8B5FFFFFF               <1>       call   _vbe3_pmfn_srs_9
1188 00001A6D 6683F84F                 <1>       cmp    ax, 004Fh
1189 00001A71 75C2                     <1>       jne    short _vbe3_srs_retn
1190                                    <1>       ; 24/01/2021
1191                                    <1>       ;cmp   ecx, 0Fh
1192                                    <1>       ;ja    short _vbe3_srs_retn
1193                                    <1>       ; 24/01/2021
1194                                    <1>       ;mov   ax, 1
1195 00001A73 B001                     <1>       mov    al, 1
1196 00001A75 66D3E0                   <1>       shl    ax, cl
1197 00001A78 660905[28120300]         <1>       or     [vbe3stbsflags], ax ; set flag for state option
1198                                    <1>       ; 23/01/2021
1199 00001A7F 89DF                     <1>       mov    edi, ebx
1200 00001A81 89C8                     <1>       mov    eax, ecx
1201 00001A83 D0E0                     <1>       shl    al, 1
1202 00001A85 66C1E706                 <1>       shl    di, 6 ; * 64
1203 00001A89 6689B8[7B3C0000]         <1>       mov    [vbestatebufsize+eax], di
1204                                    <1>                           ; save buf size for option
1205                                    <1>       ;xchg  edi, ebx
1206                                    <1>       ; edi = state buffer size in bytes
1207 00001A90 B04F                     <1>       mov    al, 4Fh
1208 00001A92 C3                       <1>       retn
1209                                    <1>
1210                                    <1> _vbe3_pmfn_srs_2:
1211                                    <1>       ; 24/01/2021
1212                                    <1>       ; !!! CLEAR CL BIT 2 !!!
1213                                    <1>       ; (when bit 2 is set, function causes cpu exception)
1214                                    <1>       ; BIOS data will not be saved and restored
1215                                    <1>       ; (to prevent protected mode page fault error)
1216                                    <1>
1217 00001A93 F6C1FD                   <1>       test   cl, ~2 ; test cl, not 2
1218 00001A96 7498                     <1>       jz     short _vbe3_pmfn_srs_fail
```

```
1219                                    <1>
1220 00001A98 80FA02                    <1>       cmp   dl, 2
1221 00001A9B 7748                      <1>       ja    short _vbe3_pmfn_srs_5
1222                                    <1>
1223                                    <1>       ;and  cl, ~2 ; and cl, not 2
1224                                    <1>
1225 00001A9D 53                        <1>       push  ebx ; * ; buffer address
1226                                    <1>       ; save or restore state
1227                                    <1>       ; (get required buffer size at first)
1228 00001A9E 52                        <1>       push  edx ; **
1229 00001A9F 28D2                      <1>       sub   dl, dl ; 0
1230 00001AA1 E894FFFFFF                <1>       call  _vbe3_pmfn_srs
1231 00001AA6 5A                        <1>       pop   edx ; **
1232                                    <1>       ; 24/01/2021
1233 00001AA7 5B                        <1>       pop   ebx ; *
1234 00001AA8 08E4                      <1>       or    ah, ah
1235 00001AAA 7538                      <1>       jnz   short _vbe3_pmfn_srs_4 ; error
1236                                    <1>
1237                                    <1>       ; edi = buffer size in bytes
1238 00001AAC 81FF00080000              <1>       cmp   edi, 2048
1239 00001AB2 772B                      <1>       ja    short _vbe3_pmfn_srs_3
1240                                    <1>
1241 00001AB4 80FA01                    <1>       cmp   dl, 1
1242 00001AB7 7531                      <1>       jne   short _vbe3_pmfn_srs_6 ; restore state
1243                                    <1>
1244                                    <1>       ; save video state
1245                                    <1>       ;xor  ebx, ebx ; points to VBE3SAVERESTOREBLOCK
1246                                    <1>       ;mov  ax, 4F04h
1247                                    <1>       ;call int10h_32bit_pmi
1248                                    <1>
1249                                    <1>       ; 24/01/2021
1250 00001AB9 E842000000                <1>       call  _vbe3_pmfn_srs_7
1251                                    <1>
1252 00001ABE 6683F84F                  <1>       cmp   ax, 004Fh
1253 00001AC2 7520                      <1>       jne   short _vbe3_pmfn_srs_4
1254                                    <1>
1255 00001AC4 09DB                      <1>       or    ebx, ebx  ; kernel ('sysvideo') ?
1256 00001AC6 741C                      <1>       jz    short _vbe3_pmfn_srs_4 ; yes
1257                                    <1>
1258                                    <1>       ; the caller is user
1259 00001AC8 51                        <1>       push  ecx ; *
1260 00001AC9 89F9                      <1>       mov   ecx, edi ; state buffer size
1261 00001ACB BE00760900                <1>       mov   esi, VBE3SAVERESTOREBLOCK ; source
1262                                    <1>                                 ; (vbe3 pmi buff)
1263 00001AD0 89DF                      <1>       mov   edi, ebx     ; destination (user buff)
1264 00001AD2 E81B000100                <1>       call  transfer_to_user_buffer
1265 00001AD7 59                        <1>       pop   ecx ; *
1266 00001AD8 7205                      <1>       jc    short _vbe3_pmfn_srs_3
1267                                    <1>
1268 00001ADA 29C0                      <1>       sub   eax, eax
1269 00001ADC B04F                      <1>       mov   al, 4Fh
1270 00001ADE C3                        <1>       retn
1271                                    <1>
1272                                    <1>       ; 24/01/2021
1273                                    <1> _vbe3_pmfn_srs_3:
1274 00001ADF B84F010000                <1>       mov   eax, 014Fh
1275                                    <1> _vbe3_pmfn_srs_4:
1276 00001AE4 C3                        <1>       retn
1277                                    <1> _vbe3_pmfn_srs_5:
1278 00001AE5 31C0                      <1>       xor   eax, eax
1279 00001AE7 FEC4                      <1>       inc   ah
1280                                    <1>       ; eax = 0100h, function is not supported
1281 00001AE9 C3                        <1>       retn
1282                                    <1>
1283                                    <1> _vbe3_pmfn_srs_6:
1284                                    <1>       ; restore video state
1285                                    <1>       ; 24/01/2021
1286                                    <1>       ;pop  ebx ; *
1287                                    <1>       ; 23/01/2021
1288 00001AEA 09DB                      <1>       or    ebx, ebx ; 0 ?
1289 00001AEC 7412                      <1>       jz    short _vbe3_pmfn_srs_7 ; 'sysvideo' call
1290                                    <1>       ; 24/01/2021
1291                                    <1>       ;jz   _vbe3_pmfn_srs_8
1292 00001AEE 89DE                      <1>       mov   esi, ebx
1293                                    <1>       ; esi = user's video state buffer
1294 00001AF0 51                        <1>       push  ecx ; *
1295 00001AF1 89F9                      <1>       mov   ecx, edi ; state buffer size
1296 00001AF3 BF00760900                <1>       mov   edi, VBE3SAVERESTOREBLOCK ; destination
1297                                    <1>                                 ; (vbe3 pmi buff)
1298                                    <1>       ;mov  esi, ebx     ; source (user buff)
1299 00001AF8 E83F000100                <1>       call  transfer_from_user_buffer
1300 00001AFD 59                        <1>       pop   ecx ; *
1301 00001AFE 72DF                      <1>       jc    short _vbe3_pmfn_srs_3
1302                                    <1> _vbe3_pmfn_srs_7:
1303 00001B00 53                        <1>       push  ebx ; *
1304                                    <1>       ; restore video state
1305                                    <1>       ;xor  ebx, ebx ; points to VBE3SAVERESTOREBLOCK
1306                                    <1>       ;mov  ax, 4F04h
1307                                    <1>       ;call int10h_32bit_pmi
1308                                    <1>       ; 17/01/2021
1309 00001B01 E81AFFFFFF                <1>       call  _vbe3_pmfn_srs_8
1310 00001B06 5B                        <1>       pop   ebx ; *
1311 00001B07 C3                        <1>       retn
1312                                    <1>
1313                                    <1> VIDEO_STATE:
1314                                    <1>       ; 26/06/2016
1315                                    <1>       ; 12/05/2016
1316                                    <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
1317                                    <1>
1318                                    <1> ;--------------------------------------------------
1319                                    <1> ; VIDEO STATE
1320                                    <1> ;  RETURNS THE CURRENT VIDEO STATE IN AX
1321                                    <1> ;  AH = NUMBER OF COLUMNS ON THE SCREEN
1322                                    <1> ;  AL = CURRENT VIDEO MODE
1323                                    <1> ;  BH = CURRENT ACTIVE PAGE
```

```
1324                                    <1> ;--------------------------------------------------
1325                                    <1>
1326 00001B08 8A25[CC6F0000]            <1>         mov    ah, [CRT_COLS]      ; GET NUMBER OF COLUMNS
1327 00001B0E A0[CA6F0000]              <1>         mov    al, [CRT_MODE]      ; CURRENT MODE
1328                                    <1>         ;movzx esi, al
1329                                    <1>         ;mov   ah, [esi+M6]
1330                                    <1>         ; BH = active page
1331 00001B13 8A3D[EE890100]            <1>         mov    bh, [ACTIVE_PAGE] ; GET CURRENT ACTIVE PAGE
1332 00001B19 FA                        <1>         cli    ; 02/01/2017
1333 00001B1A 5D                        <1>         pop    ebp          ; RECOVER REGISTERS
1334 00001B1B 5F                        <1>         pop    edi
1335 00001B1C 5E                        <1>         pop    esi
1336 00001B1D 59                        <1>         pop    ecx          ; DISCARD SAVED BX
1337 00001B1E EB41                      <1>         jmp    short M15    ; RETURN TO CALLER
1338                                    <1>
1339                                    <1> set_mode_ncm:
1340                                    <1>         ; 17/11/2020 (TRDOS 386 v2.0.3)
1341                                    <1>         ; 04/07/2016 - TRDOS 386 (TRDOS v2.0)
1342                                    <1>         ; set mode without clearing the video memory
1343                                    <1>         ; (ony for graphics modes)
1344                                    <1>
1345                                    <1>         ;cmp   al, 7 ; IBM PC CGA modes
1346                                    <1>         ;jna   short SET_MODE ; normal function (clear)
1347                                    <1>         ;; do not clear memory
1348                                    <1>         ;;mov  [noclearmem], al ; > 0
1349                                    <1>         ;mov   byte [noclearmem], 80h ; 17/11/2020
1350                                    <1>         ;call  _set_mode
1351                                    <1>         ;mov   byte [noclearmem], 0
1352                                    <1>           ;jmp     short VIDEO_RETURN
1353                                    <1>
1354                                    <1>         ; 17/11/2020 (TRDOS v2.0.3)
1355 00001B20 0C80                     <1>         or     al, 80h ; not clear memory option
1356                                    <1>
1357                                    <1>         ; 05/12/2020
1358                                    <1>         ; 27/11/2020
1359                                    <1>         ; 17/11/2020
1360                                    <1>         ; 08/08/2016, 10/08/2016
1361                                    <1>         ; 29/07/2016, 30/07/2016
1362                                    <1>         ; 25/07/2016, 26/07/2016, 27/07/2016
1363                                    <1>         ; 02/07/2016, 18/07/2016, 23/07/2016
1364                                    <1>         ; 24/06/2016, 26/06/2016
1365                                    <1>         ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
1366                                    <1> SET_MODE:
1367                                    <1>         ; For 32 bit TRDOS and Retro UNIX 386:
1368                                    <1>         ;      valid video mode: 03h only!
1369                                    <1>         ;      (VGA modes will be selected with another routine)
1370                                    <1>         ;
1371                                    <1>         ; set_txt_mode ; 80*25 (16 fore colors, 8 back colors)
1372                                    <1>
1373                                    <1>         ; 27/11/2020
1374                                    <1>
1375                                    <1>         ; Check if current mode is
1376                                    <1>         ; Bochs/Plex86 VBE graphics mode
1377 00001B22 803D[CA6F0000]FF          <1>         cmp    byte [CRT_MODE], 0FFh ; VESA VBE graphics mode
1378 00001B29 7220                      <1>         jb     short _set_mode_      ; signature
1379                                    <1>                                      ; VBE mode number is in
1380                                    <1>                                      ; [video_mode] bit 0to8
1381 00001B2B 88C3                      <1>         mov    bl, al ; save video mode
1382 00001B2D E89D240000                <1>         call   dispi_get_enable
1383 00001B32 50                        <1>         push   eax ; save current VBE dispi status
1384                                    <1>         ; Disable Bochs/Plex86 VBE dispi
1385                                    <1>         ;mov   ax, 0 ; VBE_DISPI_DISABLED
1386 00001B33 31C0                      <1>         xor    eax, eax ; 0
1387 00001B35 E869240000                <1>         call   dispi_set_enable
1388 00001B3A 88D8                      <1>         mov    al, bl ; restore video mode
1389 00001B3C E827000000                <1>         call   _set_mode
1390 00001B41 58                        <1>         pop    eax ; restore current VBE dispi status
1391 00001B42 7313                      <1>         jnc    short VIDEO_RETURN
1392                                    <1>         ; ! unimplemented or invalid video mode number !
1393                                    <1>         ; VBE dispi must be enabled again
1394                                    <1>         ; (return to run on current VBE graphics mode)
1395                                    <1>         ;;mov  al, [video_mode+1] ; bit 8 to 15
1396                                    <1>         ;;and  al, 0C0h ; isolate bit 14 and bit 15
1397                                    <1>         ;;or   al, 1 ; VBE_DISPI_ENABLED
1398 00001B44 E85A240000                <1>         call   dispi_set_enable
1399 00001B49 EB07                      <1>         jmp    short _video_func_err
1400                                    <1>
1401                                    <1> _set_mode_:
1402                                    <1>         ; VGA bios (non-VBE) 'setmode' procedure
1403                                    <1>
1404                                    <1>         ; 26/11/2020 (TRDOS v2.0.3)
1405                                    <1>
1406                                    <1> ;--------------------------------------------------
1407                                    <1> ; SET MODE                                       :
1408                                    <1> ;     THIS ROUTINE INITIALIZES THE ATTACHMENT TO    :
1409                                    <1> ;     THE SELECTED MODE, THE SCREEN IS BLANKED.     :
1410                                    <1> ; INPUT                                            :
1411                                    <1> ;     (AL) - MODE SELECTED (RANGE 0-7)             :
1412                                    <1> ; OUTPUT                                         :
1413                                    <1> ;     NONE                                       :
1414                                    <1> ;--------------------------------------------------
1415                                    <1>
1416 00001B4B E818000000                <1>         call   _set_mode ; 24/06/2016 (set_txt_mode)
1417                                    <1>         ; 26/11/2020
1418 00001B50 7305                      <1>         jnc    short VIDEO_RETURN
1419                                    <1>
1420                                    <1>         ; 26/11/2020
1421                                    <1> _video_func_err:
1422 00001B52 31C0                      <1>         xor    eax, eax ; function call failed
1423 00001B54 48                        <1>         dec    eax  ; 0FFFFFFFFh ; - 1
1424 00001B55 EB05                      <1>         jmp    short _video_return
1425                                    <1>
1426                                    <1> ; 12/05/2016
1427                                    <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
1428                                    <1>
```

```
1429                                <1> ;-----      NORMAL RETURN FROM ALL VIDEO RETURNS
1430                                <1>
1431                                <1> VIDEO_RETURN:
1432 00001B57 A1[48960100]         <1>       mov   eax, [video_eax] ; 12/05/2016
1433                                <1> _video_return:
1434 00001B5C FA                   <1>       cli ; 02/01/2017
1435 00001B5D 5D                   <1>       pop   ebp ; ******** ; 26/11/2020
1436 00001B5E 5F                   <1>       pop   edi ; *******
1437 00001B5F 5E                   <1>       pop   esi ; ******
1438 00001B60 5B                   <1>       pop   ebx ; *****
1439                                <1> M15:  ; VIDEO_RETURN_C
1440                                <1>       ;;15/01/2017
1441                                <1>       ; 02/01/2017
1442                                <1>       ;;mov  byte [intflg], 0
1443                                <1>       ;
1444 00001B61 59                   <1>       pop   ecx ; **** ; 26/11/2020
1445 00001B62 5A                   <1>       pop   edx ; ***
1446 00001B63 1F                   <1>       pop   ds  ; **
1447 00001B64 07                   <1>       pop   es  ; *      ; RECOVER SEGMENTS
1448 00001B65 CF                   <1>       iretd       ; ALL DONE
1449                                <1>
1450                                <1> set_txt_mode:
1451                                <1>
1452                                <1>       ; 29/07/2016
1453                                <1>       ; 27/06/2016
1454 00001B66 B003                 <1>       mov   al, 3 ; 26/11/2020 (bit 7 = 0)
1455                                <1>
1456                                <1>       ; 17/11/2020 (TRDOS v2.0.3)
1457                                <1>       ;mov  byte [noclearmem], 0
1458                                <1>
1459                                <1> ; 10/08/2016
1460                                <1> ; 08/08/2016
1461                                <1> ; 30/07/2016
1462                                <1> ; 29/07/2016
1463                                <1> ; 25/07/2016, 26/07/2016, 27/07/2016
1464                                <1> ; 07/07/2016, 18/07/2016, 23/07/2016
1465                                <1> ; 02/07/2016, 03/07/2016, 04/07/2016
1466                                <1> ; 26/06/2016
1467                                <1> ; 24/06/2016 (set_txt_mode -> _set_mode)
1468                                <1> ; 17/06/2016
1469                                <1> ; 29/05/2016
1470                                <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
1471                                <1>
1472                                <1> _set_mode:
1473                                <1>       ; 12/12/2020
1474                                <1>       ; 26/11/2020
1475                                <1>       ; call from 'biosfn_set_video_mode'
1476                                <1>       ;     (bochs/plex86 video bios code)
1477                                <1>       ; call from 'SET_MODE'
1478                                <1>       ;     (TRDOS 386 v2 default, IBM PC/AT rom bios code)
1479                                <1>       ; continue from 'set_txt_mode'
1480                                <1>
1481                                <1>       ; INPUT:
1482                                <1>       ;     al = VGA video mode
1483                                <1>       ; RETURN:
1484                                <1>       ;     cf = 1 -> video mode not implemented
1485                                <1>       ;     cf = 0 -> OK
1486                                <1>       ;
1487                                <1>       ; Modified registers: eax, bx, ecx, esi, edi, (ebp)
1488                                <1>
1489                                <1>       ; 17/11/2020 (TRDOS v2.0.3)
1490                                <1>       ; no clear memory option
1491                                <1>       ; (from mode number byte bit 7)
1492 00001B68 88C4                 <1>       mov   ah, al
1493 00001B6A 80E480               <1>       and   ah, 80h
1494                                <1>       ;mov  [noclearmem], al
1495 00001B6D 8825[57960100]       <1>       mov   [noclearmem], ah
1496                                <1>       ;and  al, 7Fh ; clear bit 7
1497                                <1>       ;;xor  [noclearmem], al ; clear bit 0 to 6
1498                                <1>       ; 26/11/2020
1499 00001B73 30E0                 <1>       xor   al, ah ; and al, 7Fh
1500                                <1>
1501                                <1>       ; 19/11/2020
1502                                <1>
1503                                <1>       ; Video mode 03h action principle:
1504                                <1>       ;
1505                                <1>       ; for case 1:
1506                                <1>       ; Current mode is 03h and next/requested mode is not 03h
1507                                <1>       ;     - save mode (set mode 03h flag)
1508                                <1>       ;     - save 8 video pages (which are will be restored)
1509                                <1>       ;     - save active page number (which will be reactivated)
1510                                <1>       ;     - set active page to 0 always (no multi screen)
1511                                <1>       ;     - save 8 cursor positions (which will be restored)
1512                                <1>       ;     - use 'noclearmem' option
1513                                <1>       ;     [p_crt_mode] = 0 -> 03h
1514                                <1>       ;
1515                                <1>       ; for case 2:
1516                                <1>       ; Current mode is 03h and next/requested mode is also 03h
1517                                <1>       ;     - clear active video page if 'noclearmem' is not set
1518                                <1>       ;     [p_crt_mode] = 0 -> 80h -> 0
1519                                <1>       ;
1520                                <1>       ; for case 3:
1521                                <1>       ; Current mode is not 03h and next/requested mode is 03h
1522                                <1>       ;     - restore video pages (8 video pages were saved)
1523                                <1>       ;     - restore active page number (which were saved)
1524                                <1>       ;     - restore 8 cursor positions (which were saved)
1525                                <1>       ;     - reset/clear mode 03h flag
1526                                <1>       ;     [p_crt_mode] = 03h -> 0
1527                                <1>       ;
1528                                <1>       ; for case 4:
1529                                <1>       ; Current mode is not 03h and next/requested mode is not 03h
1530                                <1>       ;     - use 'noclearmem' option
1531                                <1>       ;     - set active page to 0 always
1532                                <1>       ;     [p_crt_mode] = 03h -> 83h -> 03h
1533                                <1>       ;
```

```
1534                                    <1>       ; initial (boot time) values:
1535                                    <1>       ;      [p_crt_mode] = 0 ("there isn't a page backup, yet")
1536                                    <1>       ;        [CRT_MODE] = 3 (kernel's starting mode)
1537                                    <1>
1538                                    <1>       ; 26/11/2020
1539 00001B75 3C03                      <1>       cmp    al, 03h          ; mode 3, 80x25 text, 16 colors
1540 00001B77 7515                      <1>       jne    short _sm_0 ; (default mode for TRDOS 386 mainprog)
1541                                    <1>
1542                                    <1>       ; case 2 or case 3
1543                                    <1>
1544                                    <1>       ; check current video mode if it is 03h
1545 00001B79 08E4                      <1>       or     ah, ah ; 80h or 0 ('noclearmem' option)
1546 00001B7B 7521                      <1>       jnz    short _sm_1 ; do not clear display page
1547                                    <1>
1548                                    <1>       ; 26/11/2020
1549                                    <1>       ; Note:
1550                                    <1>       ; [CRT_MODE] = 0FFh for VESA VBE video modes
1551                                    <1>       ; [video_mode] = standard VGA and VESA VBE video modes
1552                                    <1>
1553 00001B7D 3805[CA6F0000]            <1>       cmp    [CRT_MODE], al      ; 03h
1554 00001B83 7520                      <1>       jne    short _sm_2  ; case 3 ([p_crt_mode] = 03h)
1555                                    <1>
1556                                    <1>       ; case 2
1557                                    <1>
1558                                    <1>       ; [p_crt_mode] = 0
1559                                    <1>
1560                                    <1>       ; 19/11/2020
1561                                    <1>       ; If '_set_mode' procedure is called for video mode 3
1562                                    <1>       ;     while video mode is 3, video page will be cleared
1563                                    <1>       ;     and cursor position of video page will be reset.
1564                                    <1>
1565                                    <1>       ; clear display page
1566 00001B85 C605[55960100]80            <1>       mov    byte [p_crt_mode], 80h ; clear page sign
1567 00001B8C EB1C                      <1>       jmp    short _sm_3  ; bypass save video page routine
1568                                    <1> _sm_0:
1569                                    <1>       ; case 1 or case 4
1570                                    <1>
1571                                    <1>       ; 05/12/2020
1572 00001B8E 803D[CA6F0000]03          <1>       cmp    byte [CRT_MODE], 3 ; is current mode 03h?
1573 00001B95 7507                      <1>       jne    short _sm_1  ; case 4 ; [p_crt_mode] = 03h
1574                                    <1>
1575                                    <1>       ; case 1
1576                                    <1>       ; [p_crt_mode] = 0
1577                                    <1>
1578                                    <1>       ; 19/11/2020
1579                                    <1>       ; If '_set_mode' procedure is called for a video mode
1580                                    <1>       ;     except video mode 3 while current video mode
1581                                    <1>       ;     is 3, all video pages of mode 3 will be copied
1582                                    <1>       ;     to 98000h address as backup, before mode change.
1583                                    <1>
1584                                    <1> _sm_save_pm:
1585                                    <1>       ; 12/12/2020
1586                                    <1>       ;; 03/07/2016
1587                                    <1>       ;; save video pages
1588                                    <1>       ;mov   esi, 0B8000h
1589                                    <1>       ;mov   edi, 98000h ; 30/07/2016
1590                                    <1>       ;mov   ecx, (0B8000h-0B0000h)/4
1591                                    <1>       ;rep   movsd
1592                                    <1>
1593                                    <1>       ;mov   byte [p_crt_mode], 3 ; previous mode, backup sign
1594                                    <1>       ;; 26/11/2020
1595                                    <1>       ;xchg  cl, [ACTIVE_PAGE]
1596                                    <1>       ;mov   [p_crt_page], cl  ; save as previous active page
1597                                    <1>       ;
1598                                    <1>       ;; save cursor positions
1599                                    <1>       ;mov   esi, CURSOR_POSN
1600                                    <1>       ;mov   edi, cursor_pposn ; cursor positions backup
1601                                    <1>       ;mov   cl, 4
1602                                    <1>       ;rep   movsd
1603                                    <1>
1604                                    <1>       ; 12/12/2020
1605 00001B97 E881020000               <1>       call   save_mode3_multiscreen
1606                                    <1>
1607                                    <1>       ; 29/07/2016
1608                                    <1>       ;;mov  [ACTIVE_PAGE], cl ; 0
1609                                    <1>       ;xchg  cl, [ACTIVE_PAGE]
1610                                    <1>       ;mov   [p_crt_page], cl  ; previous page (for mode 3)
1611                                    <1>
1612                                    <1>       ; [ACTIVE_PAGE] = 0
1613                                    <1>
1614 00001B9C EB07                      <1>       jmp    short _sm_2  ; case 1 - 19/11/2020
1615                                    <1> _sm_1:
1616                                    <1>       ; 26/11/2020
1617                                    <1>       ; 19/11/2020
1618                                    <1>
1619                                    <1>       ; case 4
1620 00001B9E 800D[55960100]80          <1>       or     byte [p_crt_mode], 80h
1621                                    <1>                           ; here [p_crt_mode] must be 83h
1622                                    <1>                           ;       (for case 4)
1623                                    <1>                           ; (because video mode 03h
1624                                    <1>                           ; was changed before as in case 1)
1625                                    <1>
1626                                    <1> _sm_2:       ; case 4 (jump to _sm_2) - 19/11/2020
1627                                    <1>
1628                                    <1>       ; 19/11/2020
1629                                    <1>       ; case 3
1630                                    <1>       ; If '_set_mode' procedure is called for video mode 3
1631                                    <1>       ;     while video mode is not 3 and if there is video
1632                                    <1>       ;     page backup for video mode 3, all (of 8) mode 3
1633                                    <1>       ;     video pages will be restored from 98000h.
1634                                    <1>
1635 00001BA5 A2[CA6F0000]               <1>       mov    [CRT_MODE], al  ; save mode in global variable
1636                                    <1> _sm_3:
1637                                    <1>       ; 30/07/2016
1638                                    <1>       ; 26/07/2016
```

```
1639                            <1>       ; 25/07/2016
1640                            <1>       ; set_mode_vga:
1641                            <1>       ; 18/07/2016
1642                            <1>       ; 14/07/2016
1643                            <1>       ; 09/07/2016
1644                            <1>       ; 04/07/2016
1645                            <1>       ; 03/07/2016 (TRDOS 386 = TRDOS v2.0)
1646                            <1>       ; /// video mode 13h ///
1647                            <1>       ; derived from 'Plex86/Bochs VGABios' source code
1648                            <1>       ; vgabios-0.7a (2011)
1649                            <1>       ; by the LGPL VGABios developers Team (2001-2008)
1650                            <1>       ; 'vgabios.c', 'vgatables.h'
1651                            <1>       ;
1652                            <1>       ; Oracle VirtualBox 5.0.24 VGABios Source Code
1653                            <1>       ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
1654                            <1>       ;
1655 00001BAA 88C4             <1>       mov    ah, al
1656 00001BAC B910000000       <1>       mov    ecx, vga_mode_count
1657 00001BB1 BE[E66F0000]     <1>       mov    esi, vga_modes
1658 00001BB6 31DB             <1>       xor    ebx, ebx
1659                            <1> _sm_4:
1660 00001BB8 AC               <1>       lodsb
1661 00001BB9 38C4             <1>       cmp    ah, al
1662 00001BBB 7406             <1>       je     short _sm_5
1663 00001BBD FEC3             <1>       inc    bl
1664 00001BBF E2F7             <1>       loop   _sm_4
1665                            <1>
1666                            <1>       ; UNIMPLEMENTED VIDEO MODE !
1667                            <1>       ;xor   eax, eax
1668                            <1>         ;mov [video_eax], eax ; 0
1669                            <1>
1670                            <1>       ; 26/11/2020
1671 00001BC1 F9               <1>       stc    ; unimplemented video mode ! (cf=1)
1672                            <1>
1673 00001BC2 C3               <1>       retn
1674                            <1>
1675                            <1> ;-----      eBX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
1676                            <1>
1677                            <1> _sm_5:      ; 25/07/2016
1678                            <1>       ;mov   esi, ebx
1679                            <1>       ;add   esi, vga_memmodel
1680                            <1>       ;mov   al, [esi]
1681                            <1>       ; 19/11/2020
1682 00001BC3 8A83[36700000]   <1>       mov    al, [ebx+vga_memmodel]
1683 00001BC9 A2[6E960100]     <1>       mov    [VGA_MTYPE], al
1684                            <1>
1685 00001BCE 89DF             <1>       mov    edi, ebx
1686 00001BD0 81C7[46700000]   <1>       add    edi, vga_dac_s
1687 00001BD6 C0E302           <1>       shl    bl, 2 ; byte -> dword
1688 00001BD9 81C3[F66F0000]   <1>       add    ebx, vga_mode_tbl_ptr
1689                            <1>
1690                            <1>       ;mov   dword [VGA_BASE], 0B8000h
1691                            <1>       ;cmp   ah, 0Dh ; [CRT_MODE]
1692                            <1>       ;jb    short M9
1693                            <1>       ;mov   dword [VGA_BASE], 0A0000h
1694                            <1> ;M9:
1695 00001BDF 8B33             <1>       mov    esi, [ebx]
1696 00001BE1 89F3             <1>       mov    ebx, esi
1697 00001BE3 83C614           <1>       add    esi, vga_p_cm_pos ; ebx + 20
1698 00001BE6 668B06           <1>       mov    ax, [esi]       ; get the cursor mode from the table
1699 00001BE9 66A3[E36F0000]   <1>       mov    [CURSOR_MODE], ax ; save cursor mode (initial value)
1700                            <1>       ; al = 6, ah = 7
1701                            <1>       ; al = 0Dh, ah = 0Eh ; 25/07/2016
1702 00001BEF E8A8020000       <1>       call   cursor_shape_fix
1703                            <1>       ; al = 14, ah = 15 (If [CHAR_HEIGHT] = 16)
1704 00001BF4 668906           <1>       mov    [esi], ax
1705                            <1>
1706 00001BF7 56               <1>       push   esi ; *
1707                            <1>
1708 00001BF8 8A25[D16F0000]   <1>       mov    ah, [VGA_MODESET_CTL]
1709 00001BFE 80E408           <1>       and    ah, 8 ; default palette loading ?
1710 00001C01 7524             <1>       jnz    short _sm_6
1711 00001C03 66BAC603         <1>       mov    dx, 3C6h ; VGAREG_PEL_MASK (DAC mask register)
1712 00001C07 B0FF             <1>       mov    al, 0FFh ; PEL mask
1713 00001C09 EE               <1>       out    dx, al
1714 00001C0A 8A27             <1>       mov    ah, [edi] ; DAC model (selection number)
1715 00001C0C E87E100000       <1>       call   load_dac_palette
1716                            <1>       ; ecx = 0
1717 00001C11 F605[D16F0000]02 <1>       test   byte [VGA_MODESET_CTL], 2 ; gray scale summing
1718 00001C18 740D             <1>       jz     short _sm_6
1719 00001C1A 53               <1>       push   ebx
1720 00001C1B 29DB             <1>       sub    ebx, ebx ; sub bl, bl
1721 00001C1D 66B90001         <1>         mov     cx, 256
1722 00001C21 E8BC100000       <1>       call   gray_scale_summing
1723 00001C26 5B               <1>       pop    ebx
1724                            <1> _sm_6:
1725                            <1>       ; Reset Attribute Ctl flip-flop
1726 00001C27 66BADA03         <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
1727 00001C2B EC               <1>       in     al, dx
1728                            <1>       ; Set Attribute Ctl
1729 00001C2C 89DE             <1>       mov    esi, ebx ; addr of params tbl for selected mode
1730 00001C2E 83C623           <1>       add    esi, 35  ; actl regs
1731 00001C31 30E4             <1>       xor    ah, ah ; 0
1732 00001C33 66BAC003         <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
1733                            <1> _sm_7:
1734 00001C37 88E0             <1>       mov    al, ah
1735 00001C39 EE               <1>       out    dx, al ; index
1736 00001C3A AC               <1>       lodsb
1737                            <1>       ; DX = 3C0h = VGAREG_ACTL_WRITE_DATA
1738 00001C3B EE               <1>       out    dx, al ; value
1739 00001C3C FEC4             <1>       inc    ah
1740 00001C3E 80FC14           <1>       cmp    ah, 20 ; number of actl registers
1741 00001C41 72F4             <1>       jb     short _sm_7
1742                            <1>       ;
1743 00001C43 88E0             <1>       mov    al, ah ; 20
```

```
1744 00001C45 EE                <1>       out    dx, al ; index
1745 00001C46 28C0              <1>       sub    al, al ; 0
1746 00001C48 EE                <1>       out    dx, al ; value
1747                            <1>       ;
1748                            <1>       ; Set Sequencer Ctl
1749 00001C49 89DE              <1>       mov    esi, ebx ; addr of params tbl for selected mode
1750 00001C4B 83C605            <1>       add    esi, 5 ; sequ regs
1751                            <1>       ;
1752 00001C4E 66BAC403          <1>       mov    dx, 3C4h  ; VGAREG_SEQU_ADDRESS
1753 00001C52 EE                <1>       out    dx, al ; 0
1754 00001C53 6642              <1>       inc    dx ; 3C5h ; VGAREG_SEQU_DATA
1755 00001C55 B003              <1>       mov    al, 3
1756 00001C57 EE                <1>       out    dx, al
1757 00001C58 B401              <1>       mov    ah, 1
1758                            <1> _sm_8:
1759 00001C5A 88E0              <1>       mov    al, ah
1760                            <1>       ;mov   dx, 3C4h ; VGAREG_SEQU_ADDRESS
1761 00001C5C 664A              <1>       dec    dx
1762 00001C5E EE                <1>       out    dx, al ; index
1763 00001C5F AC                <1>       lodsb
1764 00001C60 6642              <1>       inc    dx ; 3C5h ; VGAREG_SEQU_DATA
1765 00001C62 EE                <1>       out    dx, al
1766 00001C63 80FC04            <1>       cmp    ah, 4 ; number of sequ regs
1767 00001C66 7304              <1>       jnb    short _sm_9
1768 00001C68 FEC4              <1>       inc    ah
1769 00001C6A EBEE              <1>       jmp    short _sm_8
1770                            <1> _sm_9:
1771                            <1>       ; Set Grafx Ctl
1772 00001C6C 89DE              <1>       mov    esi, ebx ; addr of params tbl for selected mode
1773 00001C6E 83C637            <1>       add    esi, 55 ; grdc regs
1774 00001C71 30E4              <1>       xor    ah, ah ; 0
1775                            <1> _sm_10:
1776 00001C73 88E0              <1>       mov    al, ah
1777 00001C75 66BACE03          <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
1778 00001C79 EE                <1>       out    dx, al
1779 00001C7A AC                <1>       lodsb
1780 00001C7B 6642              <1>       inc    dx ; 3CFh ; VGAREG_GRDC_DATA
1781 00001C7D EE                <1>       out    dx, al
1782 00001C7E FEC4              <1>       inc    ah
1783 00001C80 80FC09            <1>       cmp    ah, 9 ; number of grdc regs
1784 00001C83 72EE              <1>       jb     short _sm_10
1785                            <1>       ;
1786                            <1>       ; Disable CRTC write protection
1787 00001C85 66BAD403          <1>       mov    dx, 3D4h  ; VGAREG_VGA_CRTC_ADDRESS
1788                            <1>       ;mov   al, 11h
1789                            <1>       ;our   dx, al
1790                            <1>       ;inc   dx
1791                            <1>       ;sub   al, al
1792                            <1>       ;out   dx, al
1793 00001C89 66B81100          <1>       mov    ax, 11h
1794 00001C8D 66EF              <1>       out    dx, ax
1795 00001C8F 89DE              <1>       mov    esi, ebx ; addr of params tbl for selected mode
1796 00001C91 83C60A            <1>       add    esi, 10 ; crtc regs
1797                            <1>       ; ah = 0
1798                            <1> _sm_11:
1799 00001C94 88E0              <1>       mov    al, ah
1800                            <1>       ; dx = 3D4h = VGAREG_VGA_CRTC_ADDRESS
1801 00001C96 EE                <1>       out    dx, al ; index
1802 00001C97 AC                <1>       lodsb
1803 00001C98 6642              <1>       inc    dx  ; VGAREG_VGA_CRTC_ADDRESS + 1
1804 00001C9A EE                <1>       out    dx, al ; value
1805 00001C9B 80FC18            <1>       cmp    ah, 24 ; number of crtc registers - 1
1806 00001C9E 7306              <1>       jnb    short _sm_12
1807 00001CA0 FEC4              <1>       inc    ah
1808 00001CA2 664A              <1>       dec    dx ; 3D4h
1809 00001CA4 EBEE              <1>       jmp    short _sm_11
1810                            <1> _sm_12:
1811                            <1>       ; Set the misc register
1812 00001CA6 66BACC03          <1>       mov    dx, 3CCh ; VGAREG_READ_MISC_OUTPUT
1813 00001CAA 8A4309            <1>       mov    al, [ebx+9] ; misc reg
1814 00001CAD EE                <1>       out    dx, al
1815                            <1>       ;
1816                            <1>       ; Enable video
1817 00001CAE 66BAC003          <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
1818 00001CB2 B020              <1>       mov    al, 20h
1819 00001CB4 EE                <1>         out     dx, al   ; set bit 5 to 1
1820 00001CB5 66BADA03          <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
1821 00001CB9 EC                <1>       in     al, dx
1822                            <1>       ;
1823                            <1>       ; 17/11/2020
1824                            <1>       ;cmp   byte [noclearmem], 0
1825                            <1>         ;ja  short _sm_15
1826                            <1>
1827 00001CBA F605[57960100]80  <1>       test   byte [noclearmem], 80h
1828 00001CC1 7540              <1>       jnz    short _sm_15
1829                            <1>
1830                            <1>       ; 29/07/2016
1831 00001CC3 31C0              <1>       xor    eax, eax
1832 00001CC5 B900400000        <1>       mov    ecx, 4000h ; 16K words (32K)
1833 00001CCA 803D[6E960100]02  <1>       cmp     byte [VGA_MTYPE], 2  ; CTEXT, MTEXT, CGA
1834 00001CD1 7715              <1>       ja     short _sm_14    ; no ? (0A0000h)
1835 00001CD3 BF00800B00        <1>       mov    edi, 0B8000h
1836 00001CD8 7409              <1>       je     short _sm_13 ; CGA graphics mode
1837                            <1>       ; 08/08/2016
1838 00001CDA A3[6A960100]      <1>       mov    [VGA_INT43H], eax ; 0 ; default font
1839 00001CDF 66B82007          <1>       mov    ax, 0720h    ; CGA text mode
1840                            <1> _sm_13:
1841 00001CE3 F366AB            <1>       rep    stosw
1842 00001CE6 EB1B              <1>       jmp    short _sm_15
1843                            <1>
1844                            <1> _sm_14:
1845 00001CE8 BF00000A00        <1>       mov    edi, 0A0000h
1846                            <1>       ; ecx = 16384 dwords (64K)
1847 00001CED 66BAC403          <1>       mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
1848 00001CF1 B002              <1>       mov    al, 2
```

```
1849 00001CF3 EE                <1>       out    dx, al
1850                            <1>       ;mov   dx, 3C5h ; VGAREG_SEQU_DATA
1851 00001CF4 6642              <1>       inc    dx
1852 00001CF6 EC                <1>       in     al, dx ; mmask
1853 00001CF7 6650              <1>       push   ax
1854 00001CF9 B00F              <1>       mov    al, 0Fh ; all planes
1855 00001CFB EE                <1>       out    dx, al
1856 00001CFC 30C0              <1>       xor    al, al ; 0
1857 00001CFE F3AB              <1>       rep    stosd  ; ecx = 163684 (64K)
1858 00001D00 6658              <1>       pop    ax
1859 00001D02 EE                <1>       out    dx, al  ; mmask
1860                            <1> _sm_15:
1861                            <1>       ; ebx = addr of params tbl for selected mode
1862                            <1>       ; 10/08/2016
1863 00001D03 668B03            <1>       mov    ax, [ebx] ; num of columns, 'twidth'
1864 00001D06 A2[CC6F0000]      <1>       mov    [CRT_COLS], al
1865                            <1>       ;; 26/07/2016
1866                            <1>       ;; CRTC_ADDRESS = 3D4h (always)
1867                            <1>       ;mov   ah, [ebx+1] ; num of rows, 'theightm1'
1868 00001D0B FEC4              <1>       inc    ah ; 09/07/2016
1869 00001D0D 8825[D26F0000]    <1>       mov    [VGA_ROWS], ah
1870                            <1>       ; 10/08/2016
1871 00001D13 8A4302            <1>       mov    al, [ebx+2]
1872 00001D16 A2[CE6F0000]      <1>       mov    [CHAR_HEIGHT], al
1873                            <1>       ; 29/07/2016
1874                            <1>       ; length of regen buffer in bytes
1875 00001D1B 668B4B03          <1>       mov    cx, [ebx+3] ; 'slength_l'
1876 00001D1F 66890D[58960100]  <1>       mov    [CRT_LEN], cx
1877                            <1>       ;
1878                            <1>       ; 27/07/2016
1879 00001D26 30E4              <1>       xor    ah, ah
1880 00001D28 A0[EE890100]      <1>       mov    al, [ACTIVE_PAGE] ; may be > 0 for mode 3
1881                            <1>       ;mul   word [CRT_LEN] ; 4096 for mode 3
1882 00001D2D 66F7E1            <1>       mul    cx ; 29/07/2016
1883 00001D30 66A3[DC890100]    <1>       mov    [CRT_START], ax
1884                            <1>       ;
1885 00001D36 B060              <1>       mov    al, 60h
1886                            <1>       ;cmp   byte [noclearmem], 0
1887                            <1>       ;jna   short _sm_16
1888                            <1>       ;add   al, 80h
1889 00001D38 0A05[57960100]    <1>       or     al, [noclearmem] ; 17/11/2020
1890                            <1> _sm_16:
1891 00001D3E A2[CF6F0000]      <1>       mov    [VGA_VIDEO_CTL], al
1892 00001D43 C605[D06F0000]F9  <1>       mov    byte [VGA_SWITCHES], 0F9h
1893 00001D4A 8025[D16F0000]7F  <1>       and    byte [VGA_MODESET_CTL], 7Fh
1894                            <1>
1895 00001D51 5E                <1>       pop    esi ; *
1896                            <1>
1897                            <1>       ; 26/07/2016
1898                            <1>       ; 07/07/2016
1899 00001D52 668B0D[E36F0000]  <1>       mov    cx, [CURSOR_MODE] ; restore cursor mode (initial value)
1900 00001D59 66870E            <1>       xchg   cx, [esi] ; cl = start line, ch = end line
1901                            <1>                        ; reset to initial value
1902 00001D5C 86E9              <1>       xchg   ch, cl  ; ch = start line, cl = end line
1903 00001D5E 66890D[E36F0000]  <1>       mov    [CURSOR_MODE], cx ; save (fixed) cursor mode
1904                            <1>
1905                            <1>       ; 27/07/2016
1906 00001D65 803D[6E960100]02  <1>       cmp    byte [VGA_MTYPE], 2 ; CTEXT, MTEXT
1907 00001D6C 7317              <1>       jnb    short _sm_17
1908                            <1>
1909                            <1>       ; Set cursor shape
1910                            <1>       ;mov   cx, 0607h
1911                            <1>       ;call  _set_ctype
1912                            <1>
1913                            <1>       ; 29/07/2016
1914 00001D6E B40A              <1>       mov    ah, 10 ; 6845 register for cursor set
1915 00001D70 E84D060000        <1>       call   m16    ; output cx register
1916                            <1>
1917                            <1>       ; 25/07/2016
1918 00001D75 803D[CA6F0000]03  <1>         cmp     byte [CRT_MODE], 03h
1919 00001D7C 7507              <1>       jne    short _sm_17
1920                            <1>       ; 26/07/2016
1921                            <1>
1922 00001D7E A0[EE890100]      <1>       mov    al, [ACTIVE_PAGE]
1923 00001D83 EB0B              <1>       jmp    short _sm_18
1924                            <1> _sm_17:
1925                            <1>       ; Set cursor pos for page 0..7
1926                            <1>       ;sub   ax, ax ; eax = 0
1927 00001D85 29C0              <1>       sub    eax, eax ; 17/11/2020
1928 00001D87 BF[DE890100]      <1>       mov    edi, CURSOR_POSN
1929 00001D8C AB                <1>       stosd
1930 00001D8D AB                <1>       stosd
1931 00001D8E AB                <1>       stosd
1932 00001D8F AB                <1>       stosd
1933                            <1>       ;; Set active page 0
1934                            <1>       ;mov   [ACTIVE_PAGE], al ; 0
1935                            <1> _sm_18:
1936                            <1>       ; 29/07/2016
1937 00001D90 803D[6E960100]02  <1>       cmp    byte [VGA_MTYPE], 2 ; CTEXT, MTEXT
1938 00001D97 737F              <1>         jnb  _sm_23
1939                            <1>
1940                            <1>       ;cmp   byte [CHAR_HEIGHT], 16
1941                            <1>       ;je    short _sm_19
1942                            <1>
1943                            <1>       ;; copy and activate 8x16 font
1944                            <1>
1945                            <1>       ; 26/07/2016
1946 00001D99 B004              <1>       mov    al, 04h
1947                            <1>       ;sub   bl, bl
1948                            <1>       ; AX = 1104H ; Load ROM 8x16 Character Set
1949                            <1>       ; (BL = font block to load (EGA: 0-3; VGA: 0-7))
1950 00001D9B E8E8160000        <1>       call   load_text_8_16_pat
1951                            <1>
1952                            <1>       ; video_func_1103h:
1953                            <1>       ; biosfn_set_text_block_specifier:
```

```
1954                                    <1>       ; BL = font block selector code
1955                                    <1>       ; NOTE: TRDOS 386 only uses and sets font block 0
1956                                    <1>       ; (It is as BL = 0 for TRDOS 386)
1957 00001DA0 66BAC403                 <1>       mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
1958                                    <1>       ;;mov  ah, bl
1959                                    <1>       ;sub   ah, ah ; 0
1960                                    <1>       ;mov   al, 03h
1961                                    <1>       ; 19/11/2020
1962 00001DA4 66B80300                 <1>       mov    ax, 03h
1963 00001DA8 66EF                     <1>       out    dx, ax
1964                                    <1> _sm_19:
1965                                    <1>       ; 29/07/2016
1966                                    <1>       ; 26/07/2016
1967                                    <1>       ; 24/06/2016
1968                                    <1>       ;mov   edi, 0B8000h
1969                                    <1>       ;mov   cx, 4000h ; 16K words (32K)
1970                                    <1>       ;
1971 00001DAA 30C0                     <1>       xor    al, al
1972 00001DAC 3805[55960100]           <1>         cmp     [p_crt_mode], al ; 0
1973 00001DB2 7705                     <1>         ja      short _sm_20 ; 03h, 80h or 83h
1974                                    <1>
1975                                    <1>       ; case 1 - 19/11/2020
1976                                    <1>       ;
1977                                    <1>       ; If [pc_crt_mode] = 0, that means, previous mode is 03h
1978                                    <1>       ; and current mode is not 03h (case 1)
1979                                    <1>
1980                                    <1>       ; 30/07/2016
1981                                    <1>       ; 24/06/2016
1982                                    <1>       ; TRDOS 386 (TRDOS v2) 'set mode' modification
1983                                    <1>       ; (for multiscreen feature):
1984                                    <1>       ; If '_set_mode' procedure is called for video mode 3
1985                                    <1>       ;     while video mode is 3, video page will be cleared
1986                                    <1>       ;     and cursor position of video page will be reset.
1987                                    <1>       ; If '_set_mode' procedure is called for a video mode
1988                                    <1>       ;     except video mode 3, while current video mode
1989                                    <1>       ;     is 3, all video pages of mode 3 will be copied
1990                                    <1>       ;     to 98000h address as backup, before mode change.
1991                                    <1>       ; If '_set_mode' procedure is called for video mode 3
1992                                    <1>       ;     while video mode is not 3 and if there is video
1993                                    <1>       ;     page backup for video mode 3, all (of 8) mode 3
1994                                    <1>       ;     video pages will be restored from 98000h.
1995                                    <1>
1996                                    <1>       ; 19/11/2020
1997                                    <1>       ;mov   [ACTIVE_PAGE], al ; 0
1998                                    <1>
1999                                    <1>       ; Here,
2000                                    <1>       ; video memory already cleared if [noclearmem] <> 80h
2001                                    <1>
2002                                    <1>       ;mov   ax, 0720h
2003                                    <1>       ;mov   cx, 4000h ; 16K words (32K)
2004                                    <1>       ;mov   edi, 0B8000h
2005                                    <1>       ;rep   stosw
2006                                    <1>       ;sub   al, al
2007                                    <1>
2008                                    <1>       ;jmp   short _sm_23
2009                                    <1>
2010                                    <1>       ; Set hardware side for the new active video page
2011                                    <1>
2012 00001DB4 E9FB010000               <1>       jmp    _set_active_page ; 19/11/2020
2013                                    <1>
2014                                    <1> _sm_20:
2015                                    <1>       ; 19/11/2020
2016                                    <1>       ; case 2 or case 3 or case 4 - 19/11/2020
2017                                    <1>
2018                                    <1>       ; 19/11/2020
2019 00001DB9 803D[CA6F0000]03         <1>       cmp    byte [CRT_MODE], 3  ; new video mode
2020 00001DC0 754F                     <1>       jne    short _sm_22 ; al = 0 (& video mode <> 03h)
2021                                    <1>                           ; case 4 - 19/11/2020
2022                                    <1>                           ; ([p_crt_mode] = 83h)
2023                                    <1>
2024                                    <1>       ; case 2 or case 3 - 19/11/2020
2025                                    <1>
2026                                    <1>       ;movzx ebx, byte [ACTIVE_PAGE]
2027                                    <1>       ; 19/11/2020
2028 00001DC2 A0[56960100]             <1>       mov    al, [p_crt_page] ; previous mode 3 active page
2029                                    <1>       ;movzx ebx, al
2030                                    <1>       ;shl   bl, 1 ; * 2
2031                                    <1>       ;add   ebx, CURSOR_POSN
2032                                    <1>
2033                                    <1>       ; 29/07/2016
2034 00001DC7 F605[55960100]7F         <1>       test    byte [p_crt_mode], 7Fh ; 83h or 80h or 03h
2035 00001DCE 740F                     <1>       jz     short _sm_21 ; do not restore video pages
2036                                    <1>                           ; case 2 - 19/11/2020
2037                                    <1>       ; case 3 - 19/11/2020
2038                                    <1>
2039                                    <1>       ; ([p_crt_mode] = 03h)
2040                                    <1>
2041                                    <1>       ; New video mode is 3 while current video mode is not 3
2042                                    <1>       ; (multi screen) video pages will be restored from 098000h
2043                                    <1>
2044                                    <1>       ; 19/11/2020
2045 00001DD0 A2[EE890100]             <1>       mov    [ACTIVE_PAGE], al ; current mode 3 active page
2046                                    <1>
2047                                    <1>       ; 12/12/2020
2048                                    <1>       ;; restore video pages
2049                                    <1>       ;mov   esi, 98000h ; 30/07/2016
2050                                    <1>       ;mov   edi, 0B8000h
2051                                    <1>       ;mov   cx, 2000h ; 8K dwords (32K)
2052                                    <1>       ;rep   movsd
2053                                    <1>       ;
2054                                    <1>       ;; 19/11/2020
2055                                    <1>       ;mov   [p_crt_mode], cl ; reset ('case 3' end condition)
2056                                    <1>       ;
2057                                    <1>       ;; restore cursor positions
2058                                    <1>       ;mov   esi, cursor_pposn
```

```
2059                                    <1>       ;mov   edi, CURSOR_POSN
2060                                    <1>       ;;mov  ecx, 4 ; restore all cursor positions (16 bytes)
2061                                    <1>       ;mov   cl, 4
2062                                    <1>       ;rep   movsd
2063                                    <1>
2064                                    <1>       ; 12/12/2020
2065 00001DD5 E89C000000              <1>       call   _restore_mode3_multiscreen
2066                                    <1>
2067                                    <1>       ;jmp   short _sm_23 ; do not clear current video pages
2068                                    <1>
2069                                    <1>       ; 19/11/2020
2070 00001DDA E9D5010000              <1>       jmp    _set_active_page
2071                                    <1>
2072                                    <1> _sm_21:
2073                                    <1>       ; 19/11/2020
2074                                    <1>       ; case 2
2075                                    <1>       ;
2076                                    <1>       ; ([p_crt_mode] = 80h)
2077                                    <1>       ;
2078                                    <1>       ; User has requested to set video mode 3 again while
2079                                    <1>       ; current video mode is 3.. that means, set mode 03h
2080                                    <1>       ; parameters again and clear video page.
2081                                    <1>       ; ('noclearmem' option effects the result)
2082                                    <1>
2083                                    <1>       ; 19/11/2020
2084 00001DDF F605[57960100]80         <1>       test   byte [noclearmem], 80h
2085 00001DE6 7529                    <1>       jnz    short _sm_22 ; 'do not clear video memory'
2086                                    <1>                           ; continue with current text
2087                                    <1>                           ; (user's option/choice)
2088                                    <1>       ; clear video page
2089                                    <1>       ;mov   cx, [CRT_LEN] ; 4096
2090                                    <1>       ;shr   cx, 1 ; 2048 ; 16/11/2020
2091 00001DE8 66B82007                <1>       mov    ax, 0720h
2092                                    <1>       ; 26/11/2020
2093 00001DEC B900080000              <1>       mov    ecx, 2048 ; 4096/2
2094 00001DF1 BF00800B00              <1>       mov    edi, 0B8000h ; [crt_base]
2095 00001DF6 66033D[DC890100]        <1>       add    di, [CRT_START]
2096 00001DFD F366AB                  <1>       rep    stosw  ; FILL THE REGEN BUFFER WITH BLANKS
2097                                    <1>       ;
2098                                    <1>       ; 19/11/2020
2099 00001E00 A0[EE890100]            <1>       mov    al, [ACTIVE_PAGE] ; 0 to 7 (for video mode 3)
2100 00001E05 0FB6D8                  <1>       movzx  ebx, al
2101 00001E08 D0E3                    <1>       shl    bl, 1
2102 00001E0A 66898B[DE890100]        <1>       mov    [ebx+CURSOR_POSN], cx ; reset cursor position
2103                                    <1> _sm_22:
2104                                    <1>       ;mov   [p_crt_mode], al ; 0 ; reset
2105                                    <1>       ; 19/11/2020
2106                                    <1>       ;and   byte [p_crt_mode], 3 ; 83h -> 3, 80h -> 0
2107 00001E11 8025[55960100]7F        <1>       and    byte [p_crt_mode], 7Fh ; 83h -> 3, 80h -> 0
2108                                    <1> _sm_23:
2109                                    <1>       ; al = video page number
2110                                    <1>       ; [CRT_LEN] = length of regen buffer in bytes
2111                                    <1>       ;call _set_active_page
2112                                    <1>       ; 16/11/2020
2113 00001E18 E997010000              <1>       jmp    _set_active_page
2114                                    <1>
2115                                    <1> ;-----       NORMAL RETURN FROM ALL VIDEO RETURNS
2116                                    <1>       ;retn
2117                                    <1>
2118                                    <1> save_mode3_multiscreen:
2119                                    <1>       ; 12/12/2020 (TRDOS v2.0.3)
2120                                    <1>       ; save mode 03h video pages and cursor positions
2121                                    <1>       ;
2122                                    <1>       ; Modified registers: ecx (=0), esi, edi
2123                                    <1>
2124                                    <1>       ; 12/12/2020
2125                                    <1>       ; moved here from '_set_mode'
2126                                    <1>       ; 03/07/2016
2127                                    <1>       ; save video pages
2128 00001E1D BE00800B00              <1>       mov    esi, 0B8000h
2129 00001E22 BF00800900              <1>       mov    edi, 98000h ; 30/07/2016
2130 00001E27 B900200000              <1>       mov    ecx, (0B8000h-0B0000h)/4
2131 00001E2C F3A5                    <1>       rep    movsd
2132                                    <1>
2133 00001E2E C605[55960100]03        <1>       mov    byte [p_crt_mode], 3 ; previous mode, backup sign
2134                                    <1>       ; 26/11/2020
2135 00001E35 860D[EE890100]          <1>       xchg   cl, [ACTIVE_PAGE]
2136 00001E3B 880D[56960100]          <1>       mov    [p_crt_page], cl  ; save as previous active page
2137                                    <1>
2138                                    <1>       ; save cursor positions
2139 00001E41 BE[DE890100]            <1>       mov    esi, CURSOR_POSN
2140 00001E46 BF[5A960100]            <1>       mov    edi, cursor_pposn ; cursor positions backup
2141 00001E4B B104                    <1>       mov    cl, 4
2142 00001E4D F3A5                    <1>       rep    movsd
2143 00001E4F C3                      <1>       retn
2144                                    <1>
2145                                    <1> restore_mode3_multiscreen:
2146                                    <1>       ; 12/12/2020 (TRDOS v2.0.3)
2147                                    <1>       ; restore mode 03h video pages and cursor positions
2148                                    <1>       ;
2149                                    <1>       ; Input:
2150                                    <1>       ;    settings from the last 'save_mode3_multiscreen'
2151                                    <1>       ;
2152                                    <1>       ; Output:
2153                                    <1>       ;    AL = active video page = [ACTIVE_PAGE]
2154                                    <1>       ;
2155                                    <1>       ; Modified registers: al, ecx (=0), esi, edi
2156                                    <1>
2157 00001E50 A0[56960100]            <1>       mov    al, [p_crt_page] ; previous mode 3 active page
2158 00001E55 A2[EE890100]            <1>       mov    [ACTIVE_PAGE], al ; current mode 3 active page
2159                                    <1>
2160                                    <1>       ; 12/12/2020
2161                                    <1>       ; moved here from 'vesa_vbe3_pmi'
2162                                    <1>
2163                                    <1>       ; 07/12/2020
```

```
2164                                <1>       ; restore CRT_START according to ACTIVE_PAGE
2165                                <1>       ;mov   [CRT_START], cx ; 0
2166                                <1>       ; 12/12/2020
2167 00001E5A 66C705[DC890100]00-  <1>       mov    word [CRT_START], 0
2167 00001E62 00                   <1>
2168                                <1>
2169                                <1>       ; check active page and set it again if it is not 0
2170 00001E63 08C0                 <1>       or     al, al
2171                                <1>       ;;jz   short vbe3_pmi_7
2172                                <1>       ;jz    short _restore_mode3_multiscreen
2173 00001E65 740F                 <1>       jz     short r_m3_ms_1
2174 00001E67 88C1                 <1>       mov    cl, al
2175                                <1> ;vbe3_pmi_5:
2176                                <1> r_m3_ms_0:
2177 00001E69 668105[DC890100]00-  <1>       add    word [CRT_START], 4096
2177 00001E71 10                   <1>
2178 00001E72 FEC9                 <1>       dec    cl
2179                                <1>       ;jnz   short vbe3_pmi_5
2180 00001E74 75F3                 <1>       jnz    short r_m3_ms_0
2181                                <1> r_m3_ms_1:
2182                                <1>       ; 12/12/2020
2183                                <1>       ; moved here from '_set_mode'
2184                                <1> _restore_mode3_multiscreen:
2185                                <1>       ; Modified registers: ecx, esi, edi
2186                                <1>
2187                                <1>       ; restore video pages
2188 00001E76 BE00800900           <1>       mov    esi, 98000h ; 30/07/2016
2189 00001E7B BF00800B00           <1>       mov    edi, 0B8000h
2190                                <1>       ;mov   cx, 2000h ; 8K dwords (32K)
2191 00001E80 B900200000           <1>       mov    ecx, 2000h
2192 00001E85 F3A5                 <1>       rep    movsd
2193                                <1>
2194                                <1>       ; 19/11/2020
2195 00001E87 880D[55960100]       <1>       mov    [p_crt_mode], cl ; reset ('case 3' end condition)
2196                                <1>
2197                                <1>       ; restore cursor positions
2198 00001E8D BE[5A960100]         <1>       mov    esi, cursor_pposn
2199 00001E92 BF[DE890100]         <1>       mov    edi, CURSOR_POSN
2200                                <1>       ;mov   ecx, 4 ; restore all cursor positions (16 bytes)
2201 00001E97 B104                 <1>       mov    cl, 4
2202 00001E99 F3A5                 <1>       rep    movsd
2203 00001E9B C3                   <1>       retn
2204                                <1>
2205                                <1> cursor_shape_fix:
2206                                <1>       ; 07/07/2016
2207                                <1>       ; (Cursor start and cursor end line values -6,7-
2208                                <1>       ; will be fixed depending on character height)
2209                                <1>       ;
2210                                <1>       ; derived from 'Plex86/Bochs VGABios' source code
2211                                <1>       ; vgabios-0.7a (2011)
2212                                <1>       ; by the LGPL VGABios developers Team (2001-2008)
2213                                <1>       ; 'vgabios.c', ' biosfn_set_cursor_shape (CH,CL)'
2214                                <1>       ;
2215                                <1>       ; INPUT ->
2216                                <1>       ;      AL = cursor start line (=6)
2217                                <1>       ;      AH = cursor end line (=7)
2218                                <1>       ; OUTPUT ->
2219                                <1>       ;      AL = cursor start line (=14)
2220                                <1>       ;      AH = cursor end line (=15)
2221                                <1>       ;
2222                                <1>       ;; if((modeset_ctl&0x01)&&(cheight>8)&&(CL<8)&&(CH<0x20))
2223                                <1>
2224                                <1>       ;test  byte [VGA_MODESET_CTL], 1 ; VGA active
2225                                <1>       ;jz    short csf_3
2226 00001E9C 803D[CE6F0000]08     <1>       cmp    byte [CHAR_HEIGHT], 8
2227 00001EA3 7649                 <1>       jna    short csf_3
2228 00001EA5 80FC08               <1>       cmp    ah, 8
2229 00001EA8 7344                 <1>       jnb    short csf_3
2230 00001EAA 3C20                 <1>       cmp    al, 20h
2231 00001EAC 7340                 <1>       jnb    short csf_3
2232                                <1>       ;
2233 00001EAE 6650                 <1>       push   ax
2234                                <1>       ; {
2235                                <1>       ; if(CL!=(CH+1))
2236 00001EB0 FEC0                 <1>       inc    al
2237 00001EB2 38C4                 <1>       cmp    ah, al   ; ah != al + 1
2238 00001EB4 740F                 <1>         je     short csf_1
2239                                <1>       ; CH = ((CH+1) * cheight / 8) -1;
2240 00001EB6 8A25[CE6F0000]       <1>       mov    ah, [CHAR_HEIGHT]
2241 00001EBC F6E4                 <1>       mul    ah
2242 00001EBE C0E803               <1>       shr    al, 3 ; / 8
2243 00001EC1 FEC8                 <1>       dec    al ; - 1
2244 00001EC3 EB0E                 <1>       jmp    short csf_2
2245                                <1> csf_1:
2246                                <1>       ; }
2247                                <1>       ; else       ; ah = al + 1
2248                                <1>       ; {
2249 00001EC5 FEC4                 <1>       inc    ah     ; ah = ah + 1
2250                                <1>       ; CH = ((CL+1) * cheight / 8) - 2;
2251 00001EC7 A0[CE6F0000]         <1>       mov    al, [CHAR_HEIGHT]
2252 00001ECC F6E4                 <1>       mul    ah
2253 00001ECE C0E803               <1>       shr    al, 3 ; / 8
2254 00001ED1 2C02                 <1>       sub    al, 2 ; - 2
2255                                <1>       ; al = 14 (if [CHAR_HEIGHT] = 16)
2256                                <1> csf_2:
2257 00001ED3 880424               <1>       mov    [esp], al
2258 00001ED6 8A642401             <1>       mov    ah, [esp+1]
2259                                <1>       ; CL = ((CL+1) * cheight / 8) - 1;
2260 00001EDA FEC4                 <1>       inc    ah
2261 00001EDC A0[CE6F0000]         <1>       mov    al, [CHAR_HEIGHT]
2262 00001EE1 F6E4                 <1>       mul    ah
2263 00001EE3 C0E803               <1>       shr    al, 3 ; / 8
2264 00001EE6 FEC8                 <1>       dec    al ; - 1
2265 00001EE8 88442401             <1>       mov    [esp+1], al
2266                                <1>       ; ah = 15 (if [CHAR_HEIGHT] = 16)
```

```
                                <1>       ;
00001EEC 6658                   <1>       pop    ax
                                <1> csf_3:
00001EEE C3                     <1>       retn
                                <1>
                                <1> SET_CTYPE:
                                <1>       ; 12/09/2016
                                <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
00001EEF 803D[CA6F0000]07       <1>       cmp    byte [CRT_MODE], 7
00001EF6 0F875BFCFFFF           <1>       ja     VIDEO_RETURN ; 12/09/2016
00001EFC E805000000             <1>       call   _set_ctype
00001F01 E951FCFFFF             <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.)
00001F06 6689C8                 <1>       mov    ax, cx
00001F09 86C4                   <1>       xchg   al, ah
                                <1>       ; AL = start line, AH = stop line
00001F0B E88CFFFFFF             <1>       call   cursor_shape_fix
                                <1>       ; AL = start line (fixed), AH = stop line (fixed)
00001F10 6689C1                 <1>       mov    cx, ax
00001F13 86E9                   <1>       xchg   ch, cl
                                <1>       ; CH = start line (fixed), CL = stop line (fixed)
                                <1>       ;
00001F15 B40A                   <1>       mov    ah, 10 ; 6845 register for cursor set
00001F17 66890D[E36F0000]       <1>       mov    [CURSOR_MODE], cx ; save in data area
                                <1>       ;call  m16    ; output cx register
                                <1>       ;retn
00001F1E E99F040000             <1>         jmp     m16
                                <1>
                                <1> SET_CPOS:
                                <1>       ; 12/09/2016
                                <1>       ; 07/07/2016
                                <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
00001F23 80FF07                 <1>       cmp    bh, 7 ; video page > 7 ; 07/07/2016
00001F26 0F872BFCFFFF           <1>       ja     VIDEO_RETURN
                                <1>       ;
00001F2C 803D[CA6F0000]07       <1>       cmp    byte [CRT_MODE], 7
00001F33 770A                   <1>       ja     short vga_set_cpos ; 12/09/2016
00001F35 E85D040000             <1>       call   _set_cpos
00001F3A E918FCFFFF             <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>
00001F3F 668915[DE890100]       <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
00001F46 E90CFCFFFF             <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
00001F4B 80FF07                 <1>       cmp    bh, 7 ; video page > 7 (invalid)
00001F4E 7606                   <1>       jna    short read_cursor_1
                                <1>       ; invalid video page (input)
00001F50 31C9                   <1>       xor    ecx, ecx ; 0
00001F52 31D2                   <1>       xor    edx, edx ; 0
00001F54 EB15                   <1>       jmp    short read_cursor_2
                                <1> read_cursor_1:
                                <1>       ; 12/09/2016
00001F56 803D[CA6F0000]07       <1>       cmp    byte [CRT_MODE], 7 ; vga mode
00001F5D 7727                   <1>       ja     short vga_get_cpos
                                <1>       ;
00001F5F E815000000             <1>       call   get_cpos
00001F64 0FB70D[E36F0000]       <1>       movzx  ecx, word [CURSOR_MODE]
                                <1> read_cursor_2:
00001F6B 5D                     <1>       pop    ebp
00001F6C 5F                     <1>       pop    edi
00001F6D 5E                     <1>       pop    esi
00001F6E 5B                     <1>       pop    ebx
00001F6F 58                     <1>       pop    eax  ; DISCARD SAVED CX AND DX
00001F70 58                     <1>       pop    eax
00001F71 A1[48960100]           <1>       mov    eax, [video_eax] ; 12/05/2016
                                <1>       ;;15/01/2017
                                <1>       ;;mov  byte [intflg], 0 ; 07/01/2017
00001F76 1F                     <1>       pop    ds
00001F77 07                     <1>       pop    es
00001F78 CF                     <1>       iretd
                                <1>
                                <1> get_cpos:
                                <1>       ; 12/05/2016
                                <1>       ; 16/01/2016
                                <1>       ; BH = Video page number (0 to 7)
                                <1>       ;
00001F79 D0E7                   <1>       shl    bh, 1 ; WORD OFFSET
00001F7B 0FB6F7                 <1>       movzx  esi, bh
00001F7E 0FB796[DE890100]       <1>       movzx  edx, word [esi+CURSOR_POSN]
00001F85 C3                     <1>       retn
                                <1>
                                <1> vga_get_cpos:
                                <1>       ; 12/09/2016
                                <1>       ; get cursor position (vga)
00001F86 0FB715[DE890100]       <1>       movzx  edx, word [CURSOR_POSN] ; cursor pos for pg 0
00001F8D 31C9                   <1>       xor    ecx, ecx ; Cursor Mode = 0 (invalid)
00001F8F 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
00001F91 3C07                   <1>       cmp    al, 7  ; > 7 = invalid video page number
                                <1>       ;ja    VIDEO_RETURN
00001F93 7715                   <1>         ja   short adp_2 ; 18/11/2020
                                <1>       ;cmp   byte [CRT_MODE], 3
                                <1>       ;je    short adp_1
                                <1>       ; 18/11/2020
00001F95 8A25[CA6F0000]         <1>       mov    ah, [CRT_MODE]
00001F9B 80FC03                 <1>       cmp    ah, 3
00001F9E 7605                   <1>       jna    short adp_1 ; mode 01h, 00h (01h), 02h (03h), 03h
00001FA0 80FC07                 <1>       cmp    ah, 7      ; mode 07h (03h)
00001FA3 7505                   <1>       jne    short adp_2
                                <1>       ;and   al, al
                                <1>         ;jnz VIDEO_RETURN
                                <1>       ;;sub  al, al ; 0 ; force to page 0
                                <1> adp_1:
00001FA5 E805000000             <1>       call   set_active_page
                                <1> adp_2:
00001FAA E9A8FBFFFF             <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>       ;
00001FAF A2[EE890100]           <1>       mov    [ACTIVE_PAGE], al ; save active page value ; [ptty]
                                <1> _set_active_page:
                                <1>       ; 27/06/2015
00001FB4 0FB6D8                 <1>       movzx  ebx, al
                                <1>       ;
                                <1>       ;cbw   ; 07/09/2014 (ah=0)
00001FB7 28E4                   <1>       sub    ah, ah ; 09/12/2017
00001FB9 66F725[58960100]       <1>       mul    word [CRT_LEN]  ; get saved length of regen buffer
                                <1>                              ; display page times regen length
                                <1>       ; 10/12/2013
00001FC0 66A3[DC890100]         <1>       mov    [CRT_START], ax ; save start address for later
00001FC6 6689C1                 <1>       mov    cx, ax ; start address to cx
                                <1> _M16:
```

```
                                <1>       ;sar   cx, 1
00001FC9 66D1E9                 <1>       shr    cx, 1  ; divide by 2 for 6845 handling
00001FCC B40C                   <1>       mov    ah, 12 ; 6845 register for start address
00001FCE E8EF030000             <1>       call   m16
                                <1>       ;sal   bx, 1
                                <1>       ; 01/09/2014
00001FD3 D0E3                   <1>       shl    bl, 1  ; *2 for word offset
00001FD5 81C3[DE890100]         <1>       add    ebx, CURSOR_POSN
00001FDB 668B13                 <1>       mov    dx, [ebx] ; get cursor for this page
                                <1>       ; 16/01/2016
                                <1>       ;call  m18
                                <1>       ;retn
00001FDE E9CB030000             <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
00001FE3 0FB605[CC6F0000]       <1>       movzx  eax, byte [CRT_COLS] ; 24/06/2016
00001FEA F6E6                   <1>       mul    dh      ; row value
00001FEC 30F6                   <1>       xor    dh, dh   ; 0
00001FEE 6601D0                 <1>       add    ax, dx  ; add column value to the result
00001FF1 66D1E0                 <1>       shl    ax, 1  ; * 2 for attribute bytes
                                <1>              ; EAX = AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
00001FF4 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>
00001FF5 0FB6CF                 <1>       movzx  ecx, bh ; video page number
00001FF8 89CE                   <1>       mov    esi, ecx
00001FFA 66D1E6                 <1>       shl    si, 1
00001FFD 668B96[DE890100]       <1>       mov    dx, [esi+CURSOR_POSN]
00002004 740C                   <1>       jz     short p21
00002006 6631F6                 <1>       xor    si, si
                                <1> p20:
00002009 660335[58960100]       <1>       add    si, [CRT_LEN] ; 24/06/2016
                                <1>       ;add   si, 80*25*2 ; add length of buffer for one page
00002010 E2F7                   <1>       loop   p20
                                <1> p21:
00002012 6621D2                 <1>       and    dx, dx
00002015 7407                   <1>       jz     short p22
00002017 E8C7FFFFFF             <1>       call   position ; determine location in regen in page
0000201C 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
0000201E 66BADA03               <1>       mov    dx, 03DAh ; status port
                                <1>       ; cx = 0
00002022 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>
```

```
2582                                    <1>       ; 07/07/2016
2583 00002023 38F5                      <1>       cmp    ch, dh
2584 00002025 0F872CFBFFFF              <1>       ja     VIDEO_RETURN
2585 0000202B 38D1                      <1>       cmp    cl, dl
2586 0000202D 0F8724FBFFFF              <1>       ja     VIDEO_RETURN
2587                                    <1>       ;
2588 00002033 E805000000                <1>       call   _scroll_up
2589 00002038 E91AFBFFFF                <1>         jmp     VIDEO_RETURN
2590                                    <1>
2591                                    <1> _scroll_up:  ; from 'write_tty'
2592                                    <1>       ;
2593                                    <1>       ; cl = left upper column
2594                                    <1>       ; ch = left upper row
2595                                    <1>       ; dl = right lower column
2596                                    <1>       ; dh = right lower row
2597                                    <1>       ;
2598                                    <1>       ; al = line count
2599                                    <1>       ; bl = attribute to be used on blanked line
2600                                    <1>       ; bh = video page number (0 to 7)
2601                                    <1>
2602 0000203D E89C000000                <1>       call   test_line_count ; 16/01/2016
2603                                    <1>
2604 00002042 8A25[CA6F0000]            <1>       mov    ah, [CRT_MODE] ; current video mode
2605                                    <1>       ;;cmp  byte [CRT_MODE], 4
2606                                    <1>       ;cmp   ah, 4 ; 07/07/2016
2607                                    <1>       ;jnb   GRAPHICS_UP ; 26/06/2016
2608                                    <1>       ; 18/11/2020
2609 00002048 80FC04                    <1>       cmp    ah, 4
2610 0000204B 720A                      <1>       jb     short n0
2611 0000204D 80FC07                    <1>       cmp    ah, 7 ; TEST FOR BW CARD
2612                                    <1>                    ;      (80x25 text, mono)
2613 00002050 7405                      <1>       je     short n0 ; same with mode 3 for TRDOS 386
2614 00002052 E942050000                <1>       jmp    GRAPHICS_UP
2615                                    <1> n0:
2616                                    <1>       ; 07/07/2016
2617 00002057 80FF07                    <1>       cmp    bh, 7 ; video page number
2618 0000205A 7606                      <1>       jna    short n1
2619 0000205C 8A3D[EE890100]            <1>       mov    bh, [ACTIVE_PAGE]
2620                                    <1> n1:
2621 00002062 88DC                      <1>       mov    ah, bl ; attribute
2622 00002064 6650                      <1>       push   ax ; *
2623                                    <1>       ;mov   esi, [CRT_BASE]
2624 00002066 BE00800B00                <1>         mov     esi, 0B8000h
2625 0000206B 3A3D[EE890100]            <1>         cmp     bh, [ACTIVE_PAGE]
2626 00002071 750B                      <1>       jne    short n2
2627                                    <1>       ;
2628 00002073 66A1[DC890100]            <1>         mov     ax, [CRT_START]
2629 00002079 6601C6                    <1>         add     si, ax
2630 0000207C EB11                      <1>         jmp     short n4
2631                                    <1> n2:
2632 0000207E 20FF                      <1>         and     bh, bh
2633 00002080 740D                      <1>       jz     short n4
2634 00002082 88F8                      <1>       mov    al, bh
2635                                    <1> n3:
2636 00002084 660335[58960100]          <1>         add si, [CRT_LEN]
2637 0000208B FEC8                      <1>         dec al
2638 0000208D 75F5                      <1>       jnz    short n3
2639                                    <1> n4:
2640 0000208F E85D000000                <1>       call   scroll_position ; 16/01/2016
2641 00002094 7420                      <1>         jz      short n6
2642                                    <1>
2643 00002096 01CE                      <1>         add     esi, ecx ; from address for scroll
2644 00002098 88F5                      <1>       mov    ch, dh  ; #rows in block
2645 0000209A 28C5                      <1>       sub    ch, al ; #rows to be moved
2646                                    <1> n5:
2647 0000209C E894000000                <1>       call   n10 ; 16/01/2016
2648                                    <1>
2649 000020A1 51                        <1>         push ecx
2650 000020A2 0FB60D[CC6F0000]          <1>       movzx  ecx, byte [CRT_COLS]
2651 000020A9 00C9                      <1>       add    cl, cl
2652 000020AB 01CE                      <1>         add  esi, ecx  ; next line
2653 000020AD 01CF                      <1>         add  edi, ecx
2654 000020AF 59                        <1>       pop    ecx
2655                                    <1>
2656 000020B0 FECD                      <1>       dec    ch      ; count of lines to move
2657 000020B2 75E8                      <1>       jnz    short n5 ; row loop
2658                                    <1>       ; ch = 0
2659 000020B4 88C6                      <1>       mov    dh, al  ; #rows
2660                                    <1> n6:
2661                                    <1>       ; attribute in ah
2662 000020B6 B020                      <1>       mov    al, ' '       ; fill with blanks
2663                                    <1> n7:
2664 000020B8 E885000000                <1>       call   n11 ; 16/01/2016
2665                                    <1>
2666 000020BD 8A0D[CC6F0000]            <1>       mov    cl, [CRT_COLS]
2667 000020C3 00C9                      <1>       add    cl, cl
2668 000020C5 01CF                      <1>         add  edi, ecx
2669                                    <1>
2670 000020C7 FECE                      <1>       dec    dh
2671 000020C9 75ED                      <1>       jnz    short n7
2672                                    <1> n16:
2673 000020CB 3A3D[EE890100]            <1>       cmp    bh, [ACTIVE_PAGE]
2674 000020D1 750A                      <1>       jne    short n8
2675                                    <1>
2676                                    <1>       ;cmp   byte [CRT_MODE], 7 ; is this the black and white card
2677                                    <1>       ;je    short n8        ; if so, skip the mode reset
2678                                    <1>
2679 000020D3 A0[CB6F0000]              <1>       mov    al, [CRT_MODE_SET] ; get the value of mode set
2680 000020D8 66BAD803                  <1>       mov    dx, 03D8h ; always set color card port
2681 000020DC EE                        <1>       out    dx, al
2682                                    <1> n8:
2683 000020DD C3                        <1>       retn
2684                                    <1>
2685                                    <1> test_line_count:
2686                                    <1>       ; 12/05/2016
```

```
                        <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                        <1>       ; 07/09/2014 (scroll_up)
000020DE 08C0           <1>       or     al, al
000020E0 740E           <1>       jz     short al_set2
000020E2 6652           <1>       push   dx
000020E4 28EE           <1>       sub    dh, ch  ; subtract upper row from lower row number
000020E6 FEC6           <1>       inc    dh     ; adjust difference by 1
000020E8 38C6           <1>       cmp    dh, al       ; line count = amount of rows in window?
000020EA 7502           <1>       jne    short al_set1 ; if not the we're all set
000020EC 30C0           <1>       xor    al, al ; otherwise set al to zero
                        <1> al_set1:
000020EE 665A           <1>       pop    dx
                        <1> al_set2:
000020F0 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>
000020F1 6652           <1>       push   dx
000020F3 6689CA         <1>       mov    dx, cx ; now, upper left position in DX
000020F6 E8E8FEFFFF     <1>       call   position
000020FB 01C6           <1>       add    esi, eax
000020FD 89F7           <1>       mov    edi, esi
000020FF 665A           <1>       pop    dx     ; lower right position in DX
00002101 6629CA         <1>       sub    dx, cx
00002104 FEC6           <1>       inc    dh     ; dh = #rows
00002106 FEC2           <1>       inc    dl     ; dl = #cols in block
00002108 59             <1>       pop    ecx    ; return address
00002109 6658           <1>       pop    ax     ; * ; al = line count, ah = attribute
0000210B 51             <1>       push   ecx    ; return address
0000210C 0FB7C8         <1>       movzx  ecx, ax
0000210F 8A25[CC6F0000] <1>       mov    ah, [CRT_COLS]
00002115 F6E4           <1>       mul    ah     ; determine offset to from address
00002117 6601C0         <1>       add    ax, ax  ; *2 for attribute byte
                        <1>       ;
0000211A 6650           <1>       push   ax     ; offset
0000211C 6652           <1>       push   dx
                        <1>       ;
                        <1>       ; 04/04/2014
0000211E 66BADA03       <1>       mov    dx, 3DAh ; guaranteed to be color card here
                        <1> n9:                       ; wait_display_enable
00002122 EC             <1>         in      al, dx   ; get port
00002123 A808           <1>       test   al, RVRT ; wait for vertical retrace
00002125 74FB           <1>       jz     short n9 ; wait_display_enable
00002127 B025           <1>       mov    al, 25h
00002129 B2D8           <1>       mov    dl, 0D8h ; address control port
0000212B EE             <1>       out    dx, al ; turn off video during vertical retrace
0000212C 665A           <1>       pop    dx     ; #rows, #cols
0000212E 6658           <1>              pop    ax     ; offset
00002130 6691           <1>       xchg   ax, cx ;
                        <1>       ; ecx = offset, al = line count, ah = attribute
                        <1>       ;
00002132 08C0           <1>       or     al, al
00002134 C3             <1>       retn
                        <1> n10:
                        <1>       ; Move rows
00002135 88D1           <1>       mov    cl, dl ; get # of cols to move
00002137 56             <1>       push   esi
00002138 57             <1>       push   edi    ; save start address
                        <1> n10r:
00002139 66A5           <1>       movsw         ; move that line on screen
0000213B FEC9           <1>       dec    cl
0000213D 75FA           <1>         jnz     short n10r
0000213F 5F             <1>       pop    edi
00002140 5E             <1>       pop    esi    ; recover addresses
00002141 C3             <1>       retn
                        <1> n11:
                        <1>       ; Clear rows
                        <1>                     ; dh =  #rows
00002142 88D1           <1>         mov  cl, dl ; get # of cols to clear
00002144 57             <1>         push    edi     ; save address
                        <1> n11r:
00002145 66AB           <1>         stosw           ; store fill character
00002147 FEC9           <1>       dec    cl
00002149 75FA           <1>         jnz     short n11r
0000214B 5F             <1>         pop     edi     ; recover address
0000214C 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
0000214D 38F5           <1>       cmp    ch, dh
                        <1>       ;ja    VIDEO_RETURN
0000214F 7709           <1>       ja     short _s_d_retn ; 18/11/2020
00002151 38D1           <1>       cmp    cl, dl
                        <1>       ;ja    VIDEO_RETURN
00002153 7705           <1>       ja     short _s_d_retn ; 18/11/2020
                        <1>       ;
00002155 E805000000     <1>       call   _scroll_down
                        <1> _s_d_retn:
0000215A E9F8F9FFFF     <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>       ; !!!!
0000215F FD             <1>       std           ; DIRECTION FOR SCROLL DOWN
                        <1>       ; !!!!
00002160 E879FFFFFF     <1>       call   test_line_count ; 16/01/2016
                        <1>
00002165 8A25[CA6F0000] <1>       mov    ah, [CRT_MODE] ; current video mode
                        <1>       ;;cmp  byte [CRT_MODE], 4
                        <1>       ;cmp   ah, 4 ; 07/07/2016
                        <1>       ;jnb   GRAPHICS_DOWN ; 26/06/2016
                        <1>       ; 18/11/2020
0000216B 80FC04         <1>       cmp    ah, 4
0000216E 720A           <1>       jb     short _n0
00002170 80FC07         <1>       cmp    ah, 7 ; TEST FOR BW CARD
                        <1>                    ;      (80x25 text, mono)
00002173 7405           <1>       je     short _n0 ; same with mode 3 for TRDOS 386
00002175 E903080000     <1>       jmp    GRAPHICS_DOWN
                        <1> _n0:
                        <1>       ; 07/07/2016
0000217A 80FF07         <1>       cmp    bh, 7 ; video page number
0000217D 7606           <1>       jna    short n12
0000217F 8A3D[EE890100] <1>       mov    bh, [ACTIVE_PAGE]
                        <1>       ;
                        <1> n12:              ; CONTINUE_DOWN
00002185 88DC           <1>       mov    ah, bl
00002187 6650           <1>       push   ax     ; * ; save attribute in ah
00002189 6689D0         <1>       mov    ax, dx ; LOWER RIGHT CORNER
0000218C E860FFFFFF     <1>       call   scroll_position     ; GET REGEN LOCATION
00002191 741F           <1>       jz     short n14
00002193 29CE           <1>       sub    esi, ecx  ; SI IS FROM ADDRESS
00002195 88F5           <1>       mov    ch, dh  ; #rows in block
00002197 28C5           <1>       sub    ch, al ; #rows to be moved
                        <1> n13:
00002199 E897FFFFFF     <1>       call   n10    ; MOVE ONE ROW
                        <1>
0000219E 51             <1>       push   ecx
0000219F 8A0D[CC6F0000] <1>       mov    cl, [CRT_COLS]
000021A5 00C9           <1>       add    cl, cl
000021A7 29CE           <1>         sub  esi, ecx  ; next line
000021A9 29CF           <1>         sub  edi, ecx
000021AB 59             <1>         pop  ecx
                        <1>
000021AC FECD           <1>       dec    ch     ; count of lines to move
000021AE 75E9           <1>       jnz    short n13 ; row loop
                        <1>       ; ch = 0
000021B0 88C6           <1>       mov    dh, al  ; #rows
                        <1> n14:
                        <1>       ; attribute in ah
000021B2 B020           <1>       mov    al, ' '       ; fill with blanks
                        <1> n15:
000021B4 E889FFFFFF     <1>       call   n11 ; 16/01/2016
                        <1>
000021B9 8A0D[CC6F0000] <1>       mov    cl, [CRT_COLS]
000021BF 00C9           <1>       add    cl, cl
000021C1 29CF           <1>         sub  edi, ecx
                        <1>
000021C3 FECE           <1>       dec    dh
000021C5 75ED           <1>       jnz    short n15
                        <1>       ;
                        <1>       ; 18/11/2020
000021C7 FC             <1>       cld    ; clear direction flag
                        <1>       ;
000021C8 E9FEFEFFFF     <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
000021CD 803D[CA6F0000]07       <1>         cmp     byte [CRT_MODE], 7 ; 6!?
000021D4 7607                   <1>       jna     short read_ac_c
000021D6 31C0                   <1>       xor     eax, eax
000021D8 E97FF9FFFF             <1>         jmp     _video_return
                                <1> read_ac_c:
000021DD E805000000             <1>       call    _read_ac_current
                                <1>       ; 12/05/2016
                                <1>         ;jmp     VIDEO_RETURN
000021E2 E975F9FFFF             <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>
000021E7 8A25[CA6F0000]         <1>       mov     ah, [CRT_MODE] ; current video mode
000021ED 80FC04                 <1>       cmp     ah, 4
000021F0 720A                   <1>       jb      short p10
                                <1>       ; 18/11/2020
                                <1>       ;cmp   byte [CRT_MODE], 4
                                <1>       ;jnb   GRAPHICS_READ ; 26/06/2016
                                <1>
000021F2 80FC07                 <1>       cmp     ah, 7 ; TEST FOR BW CARD (80x25 monochrome text)
000021F5 7405                   <1>       je      short p10      ; same with mode 3 in TRDOS 386
000021F7 E9D7080000             <1>       jmp     GRAPHICS_READ
                                <1> p10:
000021FC E8F4FDFFFF             <1>       call    find_position; GET REGEN LOCATION AND PORT ADDRESS
                                <1>       ;
                                <1>       ; esi = regen location
                                <1>       ; dx = status port
                                <1>       ;
                                <1>
                                <1>       ; 18/11/2020
                                <1>       ; convert display mode to a zero value
                                <1>       ; for 80 column color mode
                                <1>       ;mov   ah, [CRT_MODE]
                                <1>       ;sub   ah, 2
                                <1>       ;shr   ah, 1
                                <1>       ;jnz   short p13
                                <1>
                                <1>       ; 05/12/2020
                                <1>       ; 18/11/2020 (TRDOS 386 v2.0.3)
                                <1>       ;xor   bl, bl ; 0
00002201 803D[CA6F0000]03       <1>       cmp     byte [CRT_MODE], 03h ; 80x25 color text
                                <1>       ;je     short p11     ; Note: Only mode 03h and mode 01h are
                                <1>       ;inc   bl     ; 1  ;       in use by TRDOS 386 as text modes
                                <1>       ;jmp   short p14     ;       (07h, 00h and 02h are redirected)
00002208 7516                   <1>       jne     short p13
                                <1>
                                <1>       ; 05/12/2020
0000220A 3A3D[EE890100]         <1>       cmp     bh, [ACTIVE_PAGE]
00002210 750E                   <1>       jne     short p13
                                <1>
                                <1>       ; WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
                                <1> p11:
00002212 FB                     <1>       sti           ; enable interrupts first
                                <1>       ; 05/12/2020
00002213 90                     <1>       nop
                                <1>       ; 18/11/2020
                                <1>       ;or    bl, bl
                                <1>       ;jnz   short p13 ; mode 01h (and mode 00h) - 40x25 color text
00002214 FA                     <1>       cli           ; block interrupts for single loop
00002215 EC                     <1>       in      al, dx ; get status from the adapter
00002216 A801                   <1>       test    al, RHRZ ; is horizontal retrace low
00002218 75F8                   <1>       jnz     short p11 ; wait until it is
                                <1> p12:                ;  wait for either retrace high
0000221A EC                     <1>       in      al, dx ; get status again
0000221B A809                   <1>       test    al, RVRT+RHRZ ; is horizontal or vertical retrace high
0000221D 74FB                   <1>       jz      short p12 ; wait until either retrace active
                                <1> ;p14:
0000221F FB                     <1>       sti
                                <1> p13:
00002220 81C600800B00           <1>       add     esi, 0B8000h
00002226 668B06                 <1>       mov     ax, [esi]
                                <1>
00002229 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
0000222A 803D[CA6F0000]07       <1>       cmp   byte [CRT_MODE], 7 ; 6!?
00002231 760A                   <1>       jna   short write_ac_c
                                <1>
00002233 E8110B0000             <1>       call  vga_write_char_attr
00002238 E91AF9FFFF             <1>       jmp    VIDEO_RETURN
                                <1>
                                <1> write_ac_c:
0000223D E834000000             <1>       call  _write_c_current
                                <1>
00002242 0FB6F7                 <1>       movzx esi, bh ; video page number (0 to 7)
00002245 889E[D36F0000]         <1>       mov   [esi+chr_attrib], bl ; color/attribute
                                <1>
0000224B E907F9FFFF             <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
00002250 803D[CA6F0000]07       <1>       cmp   byte [CRT_MODE], 7 ; 6!?
00002257 760A                   <1>       jna   short write_c_c
                                <1>
00002259 E8EB0A0000             <1>       call  vga_write_char_only
0000225E E9F4F8FFFF             <1>       jmp    VIDEO_RETURN
                                <1>
                                <1> write_c_c:
                                <1>       ;and  bh, 7 ; video page number (<= 7)
00002263 0FB6F7                 <1>       movzx esi, bh
00002266 8A9E[D36F0000]         <1>       mov   bl, [esi+chr_attrib]
                                <1>
0000226C E805000000             <1>       call  _write_c_current
00002271 E9E1F8FFFF             <1>         jmp     VIDEO_RETURN
                                <1>
                                <1> _write_c_current:  ; from 'write_tty'
                                <1>       ; 18/11/2020
                                <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>       ; 18/11/2020
00002276 8A25[CA6F0000]         <1>       mov   ah, [CRT_MODE] ; current video mode
0000227C 80FC04                 <1>       cmp   ah, 4
0000227F 720A                   <1>       jb    short p40
                                <1>
                                <1>       ;cmp  byte [CRT_MODE], 4
                                <1>         ;jnb GRAPHICS_WRITE ; 26/06/2016
                                <1>
00002281 80FC07                 <1>       cmp   ah, 7 ; TEST FOR BW CARD
00002284 7405                   <1>       je    short p40
00002286 E998070000             <1>       jmp   GRAPHICS_WRITE
                                <1> p40:
                                <1>       ; al = character
                                <1>       ; bl = color/attribute
                                <1>       ; bh = video page
                                <1>       ; cx = count of characters to write
0000228B 6652                   <1>       push  dx
0000228D 88DC                   <1>       mov   ah, bl  ; color/attribute (12/05/2016)
0000228F 6650                   <1>       push  ax    ; save character & attribute/color
00002291 6651                   <1>       push  cx
00002293 E85DFDFFFF             <1>       call  find_position  ; get regen location and port address
```

```
3107 00002298 6659                <1>       pop    cx
3108                              <1>       ; esi = regen location
3109                              <1>       ; dx = status port
3110                              <1>       ;
3111 0000229A 81C600800B00        <1>       add    esi, 0B8000h ; 30/08/2014 (crt_base)
3112                              <1>       ;
3113                              <1>       ; 18/11/2020
3114                              <1>       ; convert display mode to a zero value
3115                              <1>       ; for 80 column color mode
3116                              <1>       ;mov   ah, [CRT_MODE]
3117                              <1>       ;sub   ah, 2
3118                              <1>       ;shr   ah, 1
3119                              <1>       ;jnz   short p44    ; 26/06/2016
3120                              <1>
3121                              <1>       ; 05/12/2020
3122                              <1>       ; 18/11/2020 (TRDOS 386 v2.0.3)
3123                              <1>       ;xor   bl, bl ; 0
3124 000022A0 803D[CA6F0000]03    <1>       cmp    byte [CRT_MODE], 03h ; 80x25 color text
3125                              <1>       ;je    short p41    ; Note: Only mode 03h and mode 01h are
3126                              <1>       ;inc   bl     ; 1  ;       in use by TRDOS 386 as text modes
3127                              <1>       ;jmp   short p43    ;       (07h, 00h and 02h are redirected)
3128                              <1>       ; 05/12/2020
3129 000022A7 751A                <1>       jne    short p44
3130                              <1> p46:
3131                              <1>       ; 05/12/2020
3132 000022A9 3A3D[EE890100]      <1>       cmp     bh, [ACTIVE_PAGE]
3133 000022AF 7512                <1>       jne    short p44
3134                              <1>
3135                              <1>       ; WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
3136                              <1> p41:
3137                              <1>       ; 05/12/2020
3138 000022B1 FB                  <1>       sti          ; enable interrupts first
3139 000022B2 90                  <1>       nop
3140                              <1>       ; 18/11/2020
3141                              <1>       ;or    bl, bl
3142                              <1>       ;jnz   short p44 ; mode 01h (and mode 00h) - 40x25 color text
3143 000022B3 FA                  <1>       cli          ; block interrupts for single loop
3144 000022B4 EC                  <1>       in     al, dx ; get status from the adapter
3145 000022B5 A808                <1>       test   al, RVRT ; check for vertical retrace first
3146 000022B7 7509                <1>       jnz    short p43 ; Do fast write now if vertical retrace
3147 000022B9 A801                <1>       test   al, RHRZ ; is horizontal retrace low
3148 000022BB 75F4                <1>       jnz    short p41 ; wait until it is
3149                              <1> p42:                ;  wait for either retrace high
3150 000022BD EC                  <1>       in     al, dx ; get status again
3151 000022BE A809                <1>       test   al, RVRT+RHRZ ; is horizontal or vertical retrace high
3152 000022C0 74FB                <1>       jz     short p42 ; wait until either retrace active
3153                              <1> p43:
3154 000022C2 FB                  <1>       sti
3155                              <1> p44:
3156 000022C3 668B0424            <1>       mov    ax, [esp] ; restore the character (al) & attribute (ah)
3157 000022C7 668906              <1>       mov    [esi], ax
3158                              <1>
3159 000022CA 6649                <1>       dec    cx
3160 000022CC 740D                <1>       jz     short p45
3161                              <1>
3162 000022CE 46                  <1>       inc    esi
3163 000022CF 46                  <1>       inc    esi
3164                              <1>
3165                              <1>       ; 05/12/2020
3166 000022D0 803D[CA6F0000]03    <1>       cmp    byte [CRT_MODE], 03h
3167 000022D7 75EA                <1>       jne    short p44
3168                              <1>       ;jmp   short p41
3169 000022D9 EBCE                <1>       jmp    short p46
3170                              <1> p45:
3171 000022DB 6658                <1>       pop    ax
3172 000022DD 665A                <1>       pop    dx
3173 000022DF C3                  <1>       retn
3174                              <1>
3175                              <1> ; 09/07/2016
3176                              <1> ; 26/06/2016
3177                              <1> ; 24/06/2016
3178                              <1> ; 12/05/2016
3179                              <1> ; 18/01/2016
3180                              <1> ; 16/01/2016 - TRDOS 386 (TRDOS v2.0)
3181                              <1> ; 30/06/2015
3182                              <1> ; 27/06/2015
3183                              <1> ; 11/03/2015
3184                              <1> ; 02/09/2014
3185                              <1> ; 30/08/2014
3186                              <1> ; VIDEO FUNCTIONS
3187                              <1> ; (write_tty - Retro UNIX 8086 v1 - U9.ASM, 01/02/2014)
3188                              <1>
3189                              <1> WRITE_TTY:
3190                              <1>       ; 09/12/2017
3191                              <1>       ; 09/07/2016
3192                              <1>       ; 01/07/2016
3193                              <1>       ; 26/06/2016
3194                              <1>       ; 24/06/2016
3195                              <1>       ; 13/05/2016
3196                              <1>       ; 12/05/2016
3197                              <1>       ; 30/01/2016
3198                              <1>       ; 18/01/2016
3199                              <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
3200                              <1>       ; 13/08/2015
3201                              <1>       ; 02/09/2014
3202                              <1>       ; 30/08/2014 (Retro UNIX 386 v1 - beginning)
3203                              <1>       ; 01/02/2014 (Retro UNIX 8086 v1 - last update)
3204                              <1>       ; 03/12/2013 (Retro UNIX 8086 v1 - beginning)
3205                              <1>       ; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)
3206                              <1>       ;
3207                              <1>       ; INPUT -> AL = Character to be written
3208                              <1>       ;        BL = Color (Forecolor, Backcolor)
3209                              <1>       ;        BH = Video Page (0 to 7)
3210                              <1>
3211                              <1>       ; 09/07/2016
```

```
000022E0 803D[CA6F0000]07    <1>         cmp     byte [CRT_MODE], 7
000022E7 760A                <1>       jna    short write_tty_cga
                             <1>
000022E9 E8460D0000          <1>       call   vga_write_teletype
000022EE E964F8FFFF          <1>       jmp    VIDEO_RETURN
                             <1>
                             <1> write_tty_cga:
                             <1>       ; 13/05/2016
                             <1>       ;call  _write_tty
                             <1>       ; 01/07/2016
000022F3 E818000000          <1>       call   _write_tty_m3
000022F8 E95AF8FFFF          <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> ; 18/11/2020 (TRDOS 386 v2.0.3)
                             <1> ; 09/12/2017
                             <1> ; 08/07/2016
                             <1> ; 26/06/2016
                             <1> ; 24/06/2016
                             <1> _write_tty:
                             <1>       ; 13/05/2016
                             <1>       ; --- 18/11/2020 ---
                             <1>       ; NOTE:
                             <1>       ; Only kernel calls "_write_tty" procedure...
                             <1>       ; TRDOS 386 v2 kernel uses video mode 3 for displaying
                             <1>       ; (some error) messages and also mainprog command interpreter
                             <1>       ; (in kernel) uses "_write_tty".
                             <1>       ; So, here video mode must be set to 3 if it is not 3.
                             <1>
000022FD FA                  <1>       cli    ; disable interrupts
                             <1>       ;
                             <1>       ; 01/09/2014
000022FE 803D[CA6F0000]03    <1>       cmp    byte [CRT_MODE], 3
00002305 7409                <1>       je     short _write_tty_m3
                             <1>       ;
                             <1> set_mode_3:
00002307 53                  <1>       push   ebx
00002308 50                  <1>       push   eax
                             <1>       ;call  _set_mode
                             <1>       ; 18/11/2020
00002309 E858F8FFFF          <1>       call   set_txt_mode  ; set video mode to 03h
0000230E 58                  <1>       pop    eax
0000230F 5B                  <1>       pop    ebx
                             <1>       ;
                             <1> _write_tty_m3: ; 24/06/2016 (m3 -> _write_tty_m3)
00002310 0FB6F7              <1>       movzx  esi, bh ; 12/05/2016
00002313 66D1E6              <1>       shl    si, 1
00002316 81C6[DE890100]      <1>       add    esi, CURSOR_POSN
0000231C 668B16              <1>       mov    dx, [esi]
                             <1>       ;
                             <1>       ; dx now has the current cursor position
                             <1>       ;
0000231F 3C0D                <1>       cmp    al, 0Dh      ; CR   ; is it carriage return or control character
00002321 763E                <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>       ;
00002323 66B90100            <1>       mov    cx, 1  ; 24/06/2016
                             <1>       ; cx = count of characters to write
                             <1>       ;
00002327 E84AFFFFFF          <1>       call   _write_c_current ; 16/01/2015
                             <1>       ;
                             <1>       ; position the cursor for next char
0000232C FEC2                <1>       inc    dl           ; next column
0000232E 3A15[CC6F0000]      <1>       cmp    dl, [CRT_COLS]  ; test for column overflow
00002334 7561                <1>         jne     _set_cpos
00002336 B200                <1>       mov    dl, 0        ; column = 0
                             <1> u10:                      ; (line feed found)
00002338 80FE18              <1>       cmp    dh, 25-1     ; check for last row
0000233B 7220                <1>       jb     short u6
                             <1>       ;
```

```
                                <1>       ; scroll required
                                <1> u1:
                                <1>       ; SET CURSOR POSITION (04/12/2013)
0000233D E855000000             <1>       call  _set_cpos
                                <1>       ;
                                <1>       ; determine value to fill with during scroll
                                <1> u2:
                                <1>       ; bh = video page number
                                <1>       ;
00002342 E8A0FEFFFF             <1>       call  _read_ac_current ; 18/01/2016
                                <1>       ;
                                <1>       ; al = character, ah = attribute
                                <1>       ; bh = video page number
                                <1>       ; 18/11/2020
00002347 88E3                   <1>       mov   bl, ah ; color/attribute
                                <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
00002349 6629C9                 <1>       sub   cx, cx
                                <1>       ;mov   dx, 184Fh ; dl = 79 (column), dh = 24 (row)
                                <1>       ; 18/11/2020
0000234C B618                   <1>       mov   dh, 25-1
0000234E 8A15[CC6F0000]         <1>       mov   dl, [CRT_COLS]
00002354 FECA                   <1>       dec   dl
                                <1>       ;
00002356 B001                   <1>       mov   al, 1        ; scroll 1 line up
                                <1>             ; ah = attribute
                                <1>       ;mov  bl, al ; 12/05/2016
00002358 E9E0FCFFFF             <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
0000235D 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
0000235F EB36                   <1>       jmp     _set_cpos
                                <1>
                                <1>       ; check for control characters
                                <1> u8:
00002361 7432                   <1>       je    short u9
00002363 3C0A                   <1>       cmp   al, 0Ah            ; is it a line feed (0Ah)
00002365 74D1                   <1>       je    short u10
00002367 3C07                   <1>       cmp   al, 07h      ; is it a bell
00002369 747E                   <1>       je    short u11
0000236B 3C08                   <1>       cmp   al, 08h            ; is it a backspace
                                <1>       ;jne  short u0
0000236D 741E                   <1>       je    short bs     ; 12/12/2013
                                <1>       ; 12/12/2013 (tab stop)
0000236F 3C09                   <1>       cmp   al, 09h            ; is it a tab stop
00002371 75B0                   <1>       jne   short u0
00002373 88D0                   <1>       mov   al, dl
                                <1>       ;cbw
00002375 30E4                   <1>       xor   ah, ah ; 09/12/2017
00002377 B108                   <1>       mov   cl, 8
00002379 F6F1                   <1>       div   cl
0000237B 28E1                   <1>       sub   cl, ah
                                <1> ts:
                                <1>       ; 02/09/2014
                                <1>       ; 01/09/2014
                                <1>       ;mov  al, 20h
                                <1> tsloop:
0000237D 6651                   <1>       push  cx
                                <1>       ; 18/11/2020
                                <1>       ;push  ax
                                <1>       ;;mov  bh, [ACTIVE_PAGE]
                                <1>       ; 05/12/2020
                                <1>       ;push  bx
0000237F B020                   <1>       mov   al, 20h ; al = space (blank)
                                <1>                     ; bl = color/attribute
00002381 E88AFFFFFF             <1>       call  _write_tty_m3 ; 24/06/2016 (m3 -> _write_tty_m3)
                                <1>       ; 05/12/2020
                                <1>       ; bx is preserved in '_write_tty_m3'
                                <1>       ; 18/11/2020
                                <1>       ;pop  bx  ; BL = color/attribute, Bh = video page
                                <1>       ;pop  ax  ; AL = character
00002386 6659                   <1>       pop   cx
00002388 FEC9                   <1>       dec   cl
0000238A 75F1                   <1>       jnz   short tsloop
0000238C C3                     <1>       retn
                                <1> bs:
                                <1>       ; back space found
                                <1>
0000238D 08D2                   <1>       or    dl, dl             ; is it already at start of line
                                <1>       ;je   short u7     ; set_cursor
```

```
3422 0000238F 7406              <1>       jz     short _set_cpos
3423 00002391 664A              <1>       dec    dx                  ; no -- just move it back
3424                            <1>       ;jmp   short u7
3425 00002393 EB02              <1>       jmp    short _set_cpos
3426                            <1>
3427                            <1>       ; carriage return found
3428                            <1> u9:
3429 00002395 B200              <1>       mov    dl, 0        ; move to first column
3430                            <1>       ;jmp   short u7
3431                            <1>       ;jmp   short _set_cpos ; 30/01/2016
3432                            <1>
3433                            <1>       ; line feed found
3434                            <1> ;u10:
3435                            <1> ;     cmp    dh, 25-1     ; bottom of screen
3436                            <1> ;     jne    short u6     ; no, just set the cursor
3437                            <1> ;       jmp     u1              ; yes, scroll the screen
3438                            <1>
3439                            <1> _set_cpos:
3440                            <1>       ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
3441                            <1>       ; 27/06/2015
3442                            <1>       ; 01/09/2014
3443                            <1>       ; 30/08/2014 (Retro UNIX 386 v1)
3444                            <1>       ;
3445                            <1>       ; 04/12/2013 - 12/12/2013 (Retro UNIX 8086 v1)
3446                            <1>       ;
3447                            <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
3448                            <1>       ;
3449                            <1> ;--------------------------------------------
3450                            <1> ; SET_CPOS
3451                            <1> ;     THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
3452                            <1> ;     NEW X-Y VALUES PASSED
3453                            <1> ; INPUT
3454                            <1> ;     DX - ROW,COLUMN OF NEW CURSOR
3455                            <1> ;     BH - DISPLAY PAGE OF CURSOR
3456                            <1> ; OUTPUT
3457                            <1> ;     CURSOR ID SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
3458                            <1> ;--------------------------------------------
3459                            <1>       ;
3460 00002397 BE[DE890100]      <1>       mov    esi, CURSOR_POSN
3461 0000239C 0FB6C7            <1>         movzx   eax, bh   ; BH = video page number
3462                            <1> ;     or     al, al
3463                            <1> ;     jz     short _set_cpos_0
3464 0000239F D0E0              <1>         shl     al, 1   ; word offset
3465 000023A1 01C6              <1>         add     esi, eax
3466                            <1> ;_set_cpos_0:
3467 000023A3 668916            <1>       mov    [esi], dx ; save the pointer
3468 000023A6 383D[EE890100]    <1>       cmp    [ACTIVE_PAGE], bh
3469 000023AC 7532              <1>       jne    short m17
3470                            <1>       ;call  m18    ; CURSOR SET
3471                            <1> ;m17:              ; SET_CPOS_RETURN
3472                            <1>       ; 01/09/2014
3473                            <1> ;     retn
3474                            <1>             ; DX  = row/column
3475                            <1> m18:
3476 000023AE E830FCFFFF        <1>       call   position ; determine location in regen buffer
3477 000023B3 668B0D[DC890100]  <1>       mov    cx, [CRT_START]
3478 000023BA 6601C1            <1>       add    cx, ax  ; add char position in regen buffer
3479                            <1>                    ; to the start address (offset) for this page
3480 000023BD 66D1E9            <1>       shr    cx, 1  ; divide by 2 for char only count
3481 000023C0 B40E              <1>       mov    ah, 14 ; register number for cursor
3482                            <1>       ;call  m16    ; output value to the 6845
3483                            <1>       ;retn
3484                            <1>
3485                            <1>       ;-----THIS ROUTINE OUTPUTS THE CX REGISTER
3486                            <1>       ;     TO THE 6845 REGISTERS NAMED IN (AH)
3487                            <1> m16:
3488 000023C2 FA                <1>       cli
3489                            <1>       ;mov   dx, [addr_6845] ; address register
3490 000023C3 66BAD403          <1>       mov    dx, 03D4h ; I/O address of color card
3491 000023C7 88E0              <1>       mov    al, ah ; get value
3492 000023C9 EE                <1>       out    dx, al ; register set
3493 000023CA 6642              <1>       inc    dx     ; data register
3494 000023CC EB00              <1>       jmp    $+2    ; i/o delay
3495 000023CE 88E8              <1>       mov    al, ch ; data
3496 000023D0 EE                <1>       out    dx, al
3497 000023D1 664A              <1>       dec    dx
3498 000023D3 88E0              <1>       mov    al, ah
3499 000023D5 FEC0              <1>       inc    al     ; point to other data register
3500 000023D7 EE                <1>       out    dx, al ; set for second register
3501 000023D8 6642              <1>       inc    dx
3502 000023DA EB00              <1>       jmp    $+2    ; i/o delay
3503 000023DC 88C8              <1>       mov    al, cl ; second data value
3504 000023DE EE                <1>       out    dx, al
3505 000023DF FB                <1>       sti
3506                            <1> m17:
3507 000023E0 C3                <1>       retn
3508                            <1>
3509                            <1> _beep:
3510                            <1>       ; 12/02/2021 (TRDOS v2.0.3)
3511 000023E1 FA                <1>       cli
3512 000023E2 E811000000        <1>       call   beep
3513 000023E7 FB                <1>       sti
3514 000023E8 C3                <1>       retn
3515                            <1>
3516                            <1> beeper:
3517                            <1>       ; 04/08/2016
3518                            <1>       ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
3519                            <1>       ; 30/08/2014 (Retro UNIX 386 v1)
3520                            <1>       ; 18/01/2014
3521                            <1>       ; 03/12/2013
3522                            <1>       ; bell found
3523                            <1> u11:
3524 000023E9 FB                <1>       sti
3525 000023EA 3A3D[EE890100]    <1>       cmp    bh, [ACTIVE_PAGE]
3526 000023F0 7553              <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
000023F2 66B93305               <1>       mov   cx, 1331      ; divisor for 896 hz tone
000023F6 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>       ; 12/02/2021
                                <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>
000023F8 9C                     <1>       pushfd  ; 18/01/2014      ; save interrupt status
000023F9 FA                     <1>       cli                 ; block interrupts during update
000023FA B0B6                   <1>       mov   al, 10110110b; select timer 2, lsb, msb binary
000023FC E643                   <1>       out   TIMER+3, al  ; write timer mode register
000023FE EB00                   <1>       jmp   $+2           ; I/O delay
00002400 88C8                   <1>       mov   al, cl        ; divisor for hz (low)
00002402 E642                   <1>       out   TIMER+2,AL    ; write timer 2 count - lsb
00002404 EB00                   <1>       jmp   $+2           ; I/O delay
00002406 88E8                   <1>       mov   al, ch        ; divisor for hz (high)
00002408 E642                   <1>       out   TIMER+2, al   ; write timer 2 count - msb
0000240A E461                   <1>       in    al, PORT_B    ; get current setting of port
0000240C 88C4                   <1>       mov   ah, al        ; save that setting
0000240E 0C03                   <1>       or    al, GATE2+SPK2      ; gate timer 2 and turn speaker on
00002410 E661                   <1>       out   PORT_B, al    ; and restore interrupt status
                                <1>       ; 12/02/2021
00002412 9D                     <1>       popfd ; 18/01/2014
                                <1>       ;sti
                                <1> g7:                         ; 1/64 second per count (bl)
00002413 B90B040000             <1>       mov   ecx, 1035     ; delay count for 1/64 of a second
00002418 E829000000             <1>       call  waitf         ; go to beep delay 1/64 count
0000241D FECB                   <1>       dec   bl            ; (bl) length count expired?
0000241F 75F2                   <1>       jnz   short g7      ; no - continue beeping speaker
                                <1>       ;
00002421 9C                     <1>       pushfd ; 12/02/2021 ; save interrupt status
00002422 FA                     <1>       cli    ; 18/01/2014 ; block interrupts during update
00002423 E461                   <1>       in    al, PORT_B    ; get current port value
                                <1>         ;or      al, not (GATE2+SPK2) ; isolate current speaker bits in case
00002425 0CFC                   <1>         or      al, ~(GATE2+SPK2)
00002427 20C4                   <1>         and ah, al        ; someone turned them off during beep
00002429 88E0                   <1>       mov   al, ah        ; recover value of port
                                <1>         ;or      al, not (GATE2+SPK2) ; force speaker data off
0000242B 0CFC                   <1>       or    al, ~(GATE2+SPK2) ; isolate current speaker bits in case
0000242D E661                   <1>       out   PORT_B, al    ; and stop speaker timer
0000242F 9D                     <1>       popfd ; 12/02/2021        ; restore interrupt flag state
                                <1>       ;sti
00002430 B90B040000             <1>       mov   ecx, 1035     ; force 1/64 second delay (short)
00002435 E80C000000             <1>       call  waitf         ; minimum delay between all beeps
0000243A 9C                     <1>       pushfd              ; save interrupt status
0000243B FA                     <1>       cli                 ; block interrupts during update
0000243C E461                   <1>       in    al, PORT_B    ; get current port value in case
0000243E 2403                   <1>       and   al, GATE2+SPK2      ; someone turned them on
00002440 08E0                   <1>       or    al, ah        ; recover value of port_b
00002442 E661                   <1>       out   PORT_B, al    ; restore speaker status
00002444 9D                     <1>       popfd               ; restore interrupt flag state
                                <1> u12:
00002445 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
00002446 6650           <1>       PUSH   AX                 ; SAVE WORK REGISTER (AH)
                        <1>       ; 16/12/2014
                        <1>       ;shr   cx, 1              ; convert to count of 30 micro seconds
00002448 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:
0000244A E461           <1>       IN     AL,PORT_B          ; IN AL,SYS1
0000244C A810           <1>       TEST   AL,010H
0000244E 74FA           <1>       JZ     SHORT WR_STATE_0
                        <1> WR_STATE_1:
00002450 E461           <1>       IN     AL,PORT_B          ; IN AL,SYS1
00002452 A810           <1>       TEST   AL,010H
00002454 75FA           <1>       JNZ    SHORT WR_STATE_1
00002456 E2F2           <1>         LOOP    WR_STATE_0
                        <1>       ;
00002458 6658           <1>       POP    AX                 ; RESTORE (AH)
0000245A 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>       ;
0000245B A2[54960100]   <1>       mov    [w_str_cmd], al           ; save (AL) command
00002460 3C04           <1>       CMP    AL, 4              ; TEST FOR INVALID WRITE STRING OPTION
00002462 0F83EFF6FFFF   <1>       JNB    VIDEO_RETURN       ; IF OPTION INVALID THEN RETURN
                        <1>
                        <1>         ;JCXZ   VIDEO_RETURN            ; IF ZERO LENGTH STRING THEN RETURN
                        <1>
00002468 67E362         <1>         jcxz    P55                   ; 01/07/2016
                        <1>
                        <1>       ; 01/07/2016
                        <1>       ;and   ecx, 0FFFFh
                        <1>       ; ECX = byte count
                        <1>       ;push  ecx
0000246B 89EE           <1>       mov    esi, ebp ; user buffer
0000246D BF00000700     <1>       mov    edi, Cluster_Buffer  ; system buffer
00002472 E8C5F60000     <1>       call   transfer_from_user_buffer
                        <1>       ;pop   ecx
00002477 0F82DAF6FFFF   <1>       jc     VIDEO_RETURN
                        <1>       ; ecx = transfer (byte) count = character count
0000247D BD00000700     <1>       mov    ebp, Cluster_Buffer
                        <1>       ; 12/09/2016
00002482 803D[CA6F0000]07 <1>     cmp    byte [CRT_MODE], 7 ; 6?!
00002489 0F87A1000000   <1>       ja     vga_write_string
                        <1>       ;
0000248F 0FB6F7         <1>       movzx  esi, bh                   ; GET CURRENT CURSOR PAGE
00002492 66D1E6         <1>       SAL    SI, 1              ; CONVERT TO PAGE OFFSET  (SI= PAGE)
                        <1>       ; *****
00002495 66FFB6[DE890100] <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:
0000249C 51             <1>     push    ecx ; ****
0000249D 53             <1>     push    ebx ; *** ; 18/11/2020
0000249E 56             <1>     push    esi ; **
0000249F 52             <1>     push    edx ; *
000024A0 E8F2FEFFFF     <1>     call    _set_cpos
                        <1> P50:
000024A5 8A4500         <1>     MOV     AL, [eBP]           ; GET CHARACTER FROM INPUT STRING
000024A8 45             <1>     INC     eBP                 ; BUMP POINTER TO CHARACTER
                        <1>
                        <1> ;-----      TEST FOR SPECIAL CHARACTER'S
                        <1>
000024A9 3C08           <1>     CMP     AL, 08H                 ; IS IT A BACKSPACE
000024AB 7410           <1>     JE      short P51           ; BACK_SPACE
000024AD 3C0D           <1>     CMP     AL, 0Dh ; CR        ; IS IT CARRIAGE RETURN
000024AF 740C           <1>     JE      short P51           ; CAR_RET
000024B1 3C0A           <1>     CMP     AL, 0Ah ; LF        ; IS IT A LINE FEED
000024B3 7408           <1>     JE      short P51           ; LINE_FEED
                        <1>     ; 18/11/2020
000024B5 3C09           <1>     cmp     al, 09h                 ; is it a tab stop
000024B7 7404           <1>     je      short P51
                        <1>     ;
000024B9 3C07           <1>     CMP     AL, 07h                 ; IS IT A BELL
000024BB 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>
000024BD E84EFEFFFF     <1>     call    _write_tty_m3
                        <1>
000024C2 5A             <1>     pop     edx ; *
000024C3 5E             <1>     pop     esi ; **
                        <1>
000024C4 668B96[DE890100]   <1>     MOV     DX, [eSI+CURSOR_POSN]     ; GET CURRENT CURSOR POSITION
000024CB EB44           <1>     JMP     SHORT P54           ; SET CURSOR POSITION AND CONTINUE
                        <1> P55:
000024CD E985F6FFFF     <1>     JMP     VIDEO_RETURN
                        <1> P52:
000024D2 66B90100       <1>     MOV     CX, 1               ; SET CHARACTER WRITE AMOUNT TO ONE
000024D6 803D[54960100]02   <1>     CMP     byte [w_str_cmd], 2 ; IS THE ATTRIBUTE IN THE STRING
000024DD 7204           <1>     JB      short P53           ; IF NOT THEN SKIP
000024DF 8A5D00         <1>     MOV     BL, [eBP]           ; ELSE GET NEW ATTRIBUTE
000024E2 45             <1>     INC     eBP                 ; BUMP STRING POINTER
                        <1> P53:
                        <1>     ;MOV    AH,09H              ; GOT_CHARACTER
                        <1>     ;INT    10H                 ; WRITE CHARACTER TO THE CRT
                        <1>
000024E3 E88EFDFFFF     <1>     call    _write_c_current
                        <1>
000024E8 5A             <1>     pop     edx ; *
                        <1>
                        <1>     ; 05/12/2020
                        <1>     ; bx is preserved in '_write_c_current'
                        <1>     ; 18/11/2020
                        <1>     ;mov    ebx, [esp+4] ; ***
                        <1>
000024E9 0FB6F7         <1>     movzx   esi, bh ; video page number (0 to 7)
000024EC 889E[D36F0000]     <1>     mov     [esi+chr_attrib], bl ; color/attribute
                        <1>
000024F2 FEC2           <1>     INC     DL                  ; INCREMENT COLUMN COUNTER
000024F4 3A15[CC6F0000]     <1>     CMP     DL, [CRT_COLS]          ; IF COLS ARE WITHIN RANGE FOR THIS MODE
                        <1>     ;JB     short P54           ;    THEN GO TO COLUMNS SET
000024FA 7214           <1>     jb      short P56 ; 05/12/2020
000024FC FEC6           <1>     INC     DH                  ; BUMP ROW COUNTER BY ONE
000024FE 28D2           <1>     SUB     DL, DL              ; SET COLUMN COUNTER TO ZERO
00002500 80FE19         <1>     CMP     DH, 25              ; IF ROWS ARE LESS THAN 25 THEN
                        <1>     ;JB     short P54           ; GO TO ROWS_COLUMNS_SET
00002503 720B           <1>     jb      short P56 ; 05/12/2020
                        <1>
                        <1>     ; 18/11/2020
                        <1>     ;MOV    AX,0E0AH            ; ELSE SCROLL SCREEN
                        <1>     ;INT    10H                 ; RESET ROW COUNTER TO 24
                        <1>
                        <1>     ; 18/11/2020
00002505 B00A           <1>     mov     al, 0Ah      ; line feed
                        <1>
00002507 E804FEFFFF     <1>     call    _write_tty_m3
                        <1>
0000250C 66BA0018       <1>     mov     dx, 1800h           ; Column = 0, Row = 24
                        <1> P56:
                        <1>     ; 05/12/2020
                        <1>     ; 18/11/2020
00002510 5E             <1>     pop     esi ; **
                        <1> P54:                            ; ROW_COLUMNS_SET
                        <1>     ;MOV    AX,0200H            ; SET NEW CURSOR POSITION COMMAND
                        <1>     ;INT    10H                 ; ESTABLISH NEW CURSOR POSITION
                        <1>
                        <1>     ; 18/11/2020
00002511 5B             <1>     pop     ebx ; ***
00002512 59             <1>     pop     ecx ; ****
                        <1>
                        <1>     ;LOOP   P50                 ; DO IT ONCE MORE UNTIL (CX) = ZERO
00002513 6649           <1>     dec     cx
00002515 7585           <1>     jnz     short P50next
                        <1>
00002517 665A           <1>     POP     DX  ; *****         ; RESTORE OLD CURSOR COORDINATES
                        <1>
00002519 F605[54960100]01   <1>     test    byte [w_str_cmd], 1 ; IF CURSOR WAS NOT TO BE MOVED
00002520 0F8531F6FFFF       <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
00002526 E86CFEFFFF     <1>     call    _set_cpos
0000252B E927F6FFFF     <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>
 00002530 80FEFF                <1>       cmp    dh, 0FFh
 00002533 7407                  <1>       je     short vga_wstr_1 ; user current cursor position
                                <1> vga_wstr_0:
                                <1>       ; set cursor position
 00002535 668915[DE890100]      <1>       mov    [CURSOR_POSN], dx ; save cursor pos for pg 0
                                <1> vga_wstr_1:
 0000253C 66FF35[DE890100]      <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
 00002543 F605[54960100]02      <1>       test   byte [w_str_cmd], 2
 0000254A 751D                  <1>       jnz    short vga_wstr_3
 0000254C 881D[EF890100]        <1>       mov    [ccolor], bl
                                <1> vga_wstr_2:
 00002552 51                    <1>       push   ecx
 00002553 8A4500                <1>       mov    al, [ebp]
 00002556 E8D90A0000            <1>       call   vga_write_teletype
 0000255B 59                    <1>       pop    ecx
 0000255C 6649                  <1>       dec    cx
 0000255E 741E                  <1>       jz     short vga_wstr_4
 00002560 45                    <1>       inc    ebp
 00002561 8A1D[EF890100]        <1>       mov    bl, [ccolor]
 00002567 EBE9                  <1>       jmp    short vga_wstr_2
                                <1> vga_wstr_3:
 00002569 51                    <1>       push   ecx
 0000256A 8A4500                <1>       mov    al, [ebp]
 0000256D 45                    <1>       inc    ebp
 0000256E 8A5D00                <1>       mov    bl, [ebp]
 00002571 E8BE0A0000            <1>       call   vga_write_teletype
 00002576 59                    <1>       pop    ecx
 00002577 6649                  <1>       dec    cx
 00002579 7403                  <1>       jz     short vga_wstr_4
 0000257B 45                    <1>       inc    ebp
 0000257C 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 ; **
 0000257E 665A                  <1>       pop    dx ; word [CURSOR_POSN] ; *
                                <1>       ;test  al, 1
 00002580 F605[54960100]01      <1>       test   byte [w_str_cmd], 1
 00002587 0F85CAF5FFFF          <1>       jnz    VIDEO_RETURN
 0000258D 668915[DE890100]      <1>       mov    [CURSOR_POSN], dx
 00002594 E9BEF5FFFF            <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]
00002599 80FC07                 <1>       cmp    ah, 7
0000259C 7766                   <1>       ja     short vga_graphics_up
                                <1>       ;je    n0
                                <1>
0000259E 88C7                   <1>       MOV    bh, al              ; save line count in BH
000025A0 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>
000025A3 E8DA050000             <1>       CALL   GRAPH_POSN
000025A8 0FB7F8                 <1>       MOVzx  eDI, AX                  ; SAVE RESULT AS DESTINATION ADDRESS
                                <1>
                                <1> ;-----      DETERMINE SIZE OF WINDOW
                                <1>
000025AB 6629CA                 <1>       SUB    DX, CX
000025AE 6681C20101             <1>         ADD     DX, 101h               ; ADJUST VALUES
000025B3 C0E602                 <1>       SAL    DH, 2               ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
                                <1>                                  ; AND EVEN/ODD ROWS
                                <1> ;-----      DETERMINE CRT MODE
                                <1>
000025B6 803D[CA6F0000]06       <1>       CMP    byte [CRT_MODE], 6  ; TEST FOR MEDIUM RES
000025BD 7305                   <1>         JNC     short _R7_             ; FIND_SOURCE
                                <1>
                                <1> ;-----      MEDIUM RES UP
000025BF D0E2                   <1>       SAL    DL, 1               ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
000025C1 66D1E7                 <1>       SAL    DI, 1               ; OFFSET *2 SINCE 2 BYTES/CHAR
                                <1>
                                <1> ;-----      DETERMINE THE SOURCE ADDRESS IN THE BUFFER
                                <1> _R7_:                                   ; FIND_SOURCE
000025C4 81C700800B00           <1>       add    edi, 0B8000h
000025CA C0E702                 <1>       sal    bh, 2               ; multiply number of lines by 4
000025CD 7431                   <1>         JZ      short _R11             ; IF ZERO, THEN BLANK ENTIRE FIELD
000025CF B050                   <1>       MOV    AL, 80              ; 80 BYTES/ROW
000025D1 F6E7                   <1>       mul    bh                  ; determine offset to source
000025D3 0FB7F0                 <1>       movzx  esi, ax                   ; offset to source
000025D6 01FE                   <1>       add    eSI, eDI            ; SET UP SOURCE
000025D8 88F4                   <1>       MOV    AH, DH              ; NUMBER OF ROWS IN FIELD
000025DA 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
000025DC E813040000             <1>         CALL    _R17                    ; MOVE ONE ROW
000025E1 6681EEB01F             <1>       SUB    SI, 2000h-80        ; MOVE TO NEXT ROW
000025E6 6681EFB01F             <1>       SUB    DI, 2000h-80
000025EB FECC                   <1>       DEC    AH                  ; NUMBER OF ROWS TO MOVE
000025ED 75ED                   <1>         JNZ     short _R8              ; CONTINUE TILL ALL MOVED
                                <1>
                                <1> ;-----      FILL IN THE VACATED LINE(S)
                                <1> _R9:                                    ; CLEAR ENTRY
000025EF 88D8                   <1>       mov    al, bl              ; attribute to fill with
                                <1> _R10_:
000025F1 E81A040000             <1>         CALL    _R18                    ; CLEAR THAT ROW
000025F6 6681EFB01F             <1>       SUB    DI, 2000h-80        ; POINT TO NEXT LINE
000025FB FECF                   <1>       dec    bh                  ; number of lines to fill
000025FD 75F2                   <1>         JNZ     short _R10_            ; CLEAR LOOP
000025FF C3                     <1>       retn                       ; EVERYYHING DONE
                                <1>
                                <1> _R11:                                   ; BLANK_FIELD
00002600 88F7                   <1>       mov    bh, dh              ; set blank count to everything in field
00002602 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)
```

```
4052                                    <1>       ; bl = fill value for blanked lines
4053                                    <1>       ; bh = unused
4054                                    <1>       ;
4055                                    <1>       ; ah = [CRT_MODE], current video mode
4056                                    <1>
4057 00002604 88C7                     <1>       mov    bh, al ; 31/07/2016
4058 00002606 BE[EE6F0000]             <1>       mov    esi, vga_g_modes
4059 0000260B 89F7                     <1>       mov    edi, esi
4060 0000260D 83C708                   <1>       add    edi, vga_g_mode_count
4061                                    <1> vga_g_up_0:
4062 00002610 AC                       <1>       lodsb
4063 00002611 38E0                     <1>       cmp    al, ah ; [CRT_MODE]
4064 00002613 7405                     <1>       je     short vga_g_up_1
4065 00002615 39FE                     <1>       cmp    esi, edi
4066 00002617 72F7                     <1>       jb     short vga_g_up_0
4067                                    <1>       ;xor   bh, bh ; 31/07/2016)
4068 00002619 C3                       <1>       retn   ; nothing to do
4069                                    <1> vga_g_up_1:
4070 0000261A 88F8                     <1>       mov    al, bh ; 31/07/2016
4071 0000261C 83C64F                   <1>       add    esi, vga_g_memmodel - (vga_g_modes + 1)
4072                                    <1>       ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
4073                                    <1>
4074                                    <1>       ; if(rlr>=nbrows)rlr=nbrows-1;
4075                                    <1>       ; if(clr>=nbcols)clr=nbcols-1;
4076                                    <1>       ; if(nblines>nbrows)nblines=0;
4077                                    <1>       ; cols=clr-cul+1;
4078                                    <1>
4079 0000261F 3A35[D26F0000]           <1>       cmp    dh, [VGA_ROWS]
4080 00002625 7208                     <1>       jb     short vga_g_up_2
4081 00002627 8A35[D26F0000]           <1>       mov    dh, [VGA_ROWS]
4082 0000262D FECE                     <1>       dec    dh
4083                                    <1> vga_g_up_2:
4084 0000262F 3A15[CC6F0000]           <1>       cmp    dl, [CRT_COLS]  ; = [VGA_COLS]
4085 00002635 7208                     <1>       jb     short vga_g_up_3
4086 00002637 8A15[CC6F0000]           <1>       mov    dl, [CRT_COLS]
4087 0000263D FECA                     <1>       dec    dl
4088                                    <1> vga_g_up_3:
4089 0000263F 3A05[D26F0000]           <1>       cmp    al, [VGA_ROWS]
4090 00002645 7602                     <1>       jna    short vga_g_up_4
4091 00002647 28C0                     <1>       sub    al, al ; 0
4092                                    <1> vga_g_up_4:
4093 00002649 88D7                     <1>       mov    bh, dl ; clr
4094 0000264B 28CF                     <1>       sub    bh, cl ; cul
4095 0000264D FEC7                     <1>       inc    bh ; cols = clr-cul+1
4096                                    <1>
4097 0000264F 20C0                     <1>       and    al, al ; nblines = 0
4098 00002651 755D                     <1>       jnz    short vga_g_up_6
4099 00002653 20ED                     <1>       and    ch, ch ; rul = 0
4100 00002655 7559                     <1>       jnz    short vga_g_up_6
4101 00002657 20C9                     <1>       and    cl, cl ; cul = 0
4102 00002659 7555                     <1>       jnz    short vga_g_up_6
4103                                    <1>
4104 0000265B 6650                     <1>       push   ax
4105 0000265D A0[D26F0000]             <1>       mov    al, [VGA_ROWS]
4106 00002662 FEC8                     <1>       dec    al
4107 00002664 38C6                     <1>       cmp    dh, al ; rlr = nbrows-1
4108 00002666 7546                     <1>       jne    short vga_g_up_5
4109 00002668 A0[CC6F0000]             <1>         mov     al, [CRT_COLS]  ; = VGA_COLS
4110 0000266D FEC8                     <1>       dec    al
4111 0000266F 38C2                     <1>       cmp    dl, al ; clr = nbcols-1
4112 00002671 753B                     <1>       jne    short vga_g_up_5
4113 00002673 6658                     <1>       pop    ax
4114                                    <1>
4115 00002675 66B80502                 <1>       mov    ax, 0205h
4116 00002679 66BACE03                 <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
4117 0000267D 66EF                     <1>       out    dx, ax
4118 0000267F A0[D26F0000]             <1>       mov    al, [VGA_ROWS]
4119 00002684 8A25[CC6F0000]           <1>       mov    ah, [CRT_COLS] ; = [VGA_COLS]
4120 0000268A F6E4                     <1>       mul    ah
4121 0000268C 0FB7D0                   <1>       movzx  edx, ax
4122                                    <1>       ; 08/08/2016
4123 0000268F 0FB605[CE6F0000]         <1>       movzx  eax, byte [CHAR_HEIGHT]
4124 00002696 F7E2                     <1>       mul    edx
4125                                    <1>       ; eax = byte count
4126 00002698 89C1                     <1>       mov    ecx, eax
4127                                    <1>       ;; 07/08/2016
4128                                    <1>       ;shl   dx, 3 ; * 8 ; * [CHAR_HEIGHT]
4129                                    <1>       ;mov   ecx, edx
4130 0000269A 88D8                     <1>       mov    al, bl ; fill value for blanked lines
4131 0000269C BF00000A00               <1>       mov    edi, 0A0000h
4132 000026A1 F3AA                     <1>       rep    stosb
4133                                    <1>
4134 000026A3 66B80500                 <1>       mov    ax, 5
4135 000026A7 66BACE03                 <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
4136 000026AB 66EF                     <1>       out    dx, ax ; 0005h
4137                                    <1>
4138 000026AD C3                       <1>       retn
4139                                    <1>
4140                                    <1> vga_g_up_5:
4141 000026AE 6658                     <1>       pop    ax
4142                                    <1>
4143                                    <1> vga_g_up_6:
4144                                    <1>       ; [ESI] = VGA memory model number for current video mode
4145                                    <1>         ;
4146                                    <1>         ; LINEAR8 equ 5
4147                                    <1>         ; PLANAR4 equ 4
4148                                    <1>         ; PLANAR1 equ 3
4149                                    <1>
4150 000026B0 803E04                   <1>       cmp    byte [esi], PLANAR4
4151 000026B3 7424                     <1>       je     short vga_g_up_planar
4152 000026B5 803E03                   <1>       cmp    byte [esi], PLANAR1
4153 000026B8 741F                     <1>       je     short vga_g_up_planar
4154                                    <1> vga_g_up_linear8:
4155                                    <1>       ; 07/07/2016 (TEMPORARY)
4156                                    <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))
000026BA 08C0                   <1>       or     al, al
000026BC 7414                   <1>       jz     short vga_g_up_l2
000026BE 88C4                   <1>       mov    ah, al
000026C0 00EC                   <1>       add    ah, ch ; i+nblines
                                <1>       ;jc    short vga_g_up_l2
000026C2 38F4                   <1>       cmp    ah, dh
000026C4 770C                   <1>       ja     short vga_g_up_l2
                                <1>       ; else
                                <1>       ;  vgamem_copy_pl4(cul,i+nblines,i,cols,nbcols,cheight);
000026C6 E8F2000000             <1>       call   vgamem_copy_l8
                                <1> vga_g_up_l1:
000026CB FEC5                   <1>       inc    ch
000026CD 38F5                   <1>       cmp    ch, dh
000026CF 76E9                   <1>       jna    short vga_g_up_l0
000026D1 C3                     <1>       retn
                                <1> vga_g_up_l2:
                                <1>       ; vgamem_fill_pl4(cul,i,cols,nbcols,cheight,attr);
000026D2 E850010000             <1>       call   vgamem_fill_l8
000026D7 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))
000026D9 20C0                   <1>       and    al, al
000026DB 7414                   <1>       jz     short vga_g_up_pl2
000026DD 88C4                   <1>       mov    ah, al
000026DF 00EC                   <1>       add    ah, ch ; i+nblines
                                <1>       ;jc    short vga_g_up_pl2
000026E1 38F4                   <1>       cmp    ah, dh
000026E3 770C                   <1>       ja     short vga_g_up_pl2
                                <1>       ; else
                                <1>       ;  vgamem_copy_pl4(cul,i+nblines,i,cols,nbcols,cheight);
000026E5 E80E000000             <1>       call   vgamem_copy_pl4
                                <1> vga_g_up_pl1:
000026EA FEC5                   <1>       inc    ch
000026EC 38F5                   <1>       cmp    ch, dh
000026EE 76E9                   <1>       jna    short vga_g_up_pl0
000026F0 C3                     <1>       retn
                                <1> vga_g_up_pl2:
                                <1>       ; vgamem_fill_pl4(cul,i,cols,nbcols,cheight,attr);
000026F1 E870000000             <1>       call   vgamem_fill_pl4
000026F6 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>
000026F8 52                     <1>       push   edx
000026F9 50                     <1>       push   eax
                                <1>
                                <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0105)
000026FA 66B80501               <1>       mov    ax, 0105h
000026FE 66BACE03               <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
00002702 66EF                   <1>       out    dx, ax
                                <1>
                                <1>       ; 07/08/2016
                                <1>       ;mov     ah, [esp+1]
                                <1>       ;movzx edx, ah ; ysrc
00002704 0FB6542401             <1>       movzx  edx, byte [esp+1]
                                <1>       ; 08/08/2016
00002709 0FB605[CE6F0000]       <1>       movzx  eax, byte [CHAR_HEIGHT]
00002710 8A25[CC6F0000]         <1>       mov    ah, [CRT_COLS] ; nbcols
00002716 F6E4                   <1>       mul    ah
                                <1>       ;; 07/08/2016
                                <1>       ;movzx eax, byte [CRT_COLS]
                                <1>       ;shl   ax, 3 ; * 8 ; * [CHAR_HEIGHT]
00002718 50                     <1>       push   eax ; cheight * nbcols
00002719 F7E2                   <1>       mul    edx ; * ysrc
                                <1>       ; eax = ysrc * cheight * nbcols
                                <1>       ; edx = 0
0000271B 88CA                   <1>       mov    dl, cl ; edx = xstart
0000271D 01D0                   <1>       add    eax, edx
0000271F 89C6                   <1>       mov    esi, eax ; src
00002721 88EA                   <1>       mov    dl, ch ; ydest
00002723 58                     <1>       pop    eax ; cheight * nbcols
00002724 F7E2                   <1>       mul    edx
                                <1>       ; eax = ydest * cheight * nbcols
00002726 88CA                   <1>       mov    dl, cl ; edx = xstart
00002728 01D0                   <1>       add    eax, edx
```

```
4262 0000272A 89C7                <1>       mov    edi, eax ; dest
4263                              <1>       ; esi = src
4264                              <1>       ; edi = dest
4265                              <1>       ; for(i=0;i<cheight;i++)
4266                              <1>       ; {
4267                              <1>       ;  memcpyb(0xa000,dest+i*nbcols,0xa000,src+i*nbcols,cols);
4268                              <1>       ; }
4269 0000272C 51                  <1>       push   ecx
4270 0000272D B900000A00          <1>       mov    ecx, 0A0000h
4271 00002732 01CE                <1>       add    esi, ecx
4272 00002734 01CF                <1>       add    edi, ecx
4273                              <1>       ; 08/08/2016
4274 00002736 8A35[CE6F0000]      <1>       mov    dh, [CHAR_HEIGHT]
4275                              <1>       ;; 07/08/2016
4276                              <1>       ;mov   dh, 8 ; 07/08/2016
4277 0000273C 28D2                <1>       sub    dl, dl ; i
4278                              <1> vgamem_copy_pl4_0:
4279 0000273E 56                  <1>       push   esi
4280 0000273F 57                  <1>       push   edi
4281 00002740 0FB605[CC6F0000]    <1>       movzx  eax, byte [CRT_COLS]
4282 00002747 F6E2                <1>       mul    dl
4283                              <1>       ; eax = i * nbcols
4284 00002749 01C7                <1>       add    edi, eax ; dest+i*nbcols
4285 0000274B 01C6                <1>       add    esi, eax
4286 0000274D 0FB6CF              <1>       movzx  ecx, bh ; cols
4287 00002750 F3A4                <1>       rep    movsb
4288 00002752 5F                  <1>       pop    edi
4289 00002753 5E                  <1>       pop    esi
4290 00002754 FECE                <1>       dec    dh
4291 00002756 75E6                <1>       jnz    short vgamem_copy_pl4_0
4292                              <1> vgamem_copy_pl4_1:
4293 00002758 59                  <1>       pop    ecx
4294                              <1>
4295                              <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0005);
4296 00002759 66B80500            <1>       mov    ax, 0005h
4297 0000275D 66BACE03            <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
4298 00002761 66EF                <1>       out    dx, ax
4299                              <1>
4300 00002763 58                  <1>       pop    eax
4301 00002764 5A                  <1>       pop    edx
4302                              <1>
4303 00002765 C3                  <1>       retn
4304                              <1>
4305                              <1> vgamem_fill_pl4:
4306                              <1>       ; 08/08/2016
4307                              <1>       ; 07/08/2016
4308                              <1>       ; 04/08/2016
4309                              <1>       ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
4310                              <1>       ;
4311                              <1>       ; derived from 'Plex86/Bochs VGABios' source code
4312                              <1>       ; vgabios-0.7a (2011)
4313                              <1>       ; by the LGPL VGABios developers Team (2001-2008)
4314                              <1>       ; 'vgabios.c', 'vgamem_fill_pl4'
4315                              <1>       ;
4316                              <1>       ; vgamem_fill_pl4(xstart,ystart,cols,nbcols,cheight,attr)
4317                              <1>       ; cl = xstart, edi = ch = ystart, bh = cols,
4318                              <1>       ; [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight, attr = 0
4319                              <1>
4320                              <1>       ; dest=ystart*cheight*nbcols+xstart;
4321 00002766 52                  <1>       push   edx
4322 00002767 50                  <1>       push   eax
4323                              <1>
4324                              <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0205)
4325 00002768 66B80502            <1>       mov    ax, 0205h
4326 0000276C 66BACE03            <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
4327 00002770 66EF                <1>       out    dx, ax
4328                              <1>
4329                              <1>              ; 08/08/2016
4330 00002772 0FB605[CE6F0000]    <1>       movzx   eax, byte [CHAR_HEIGHT]
4331 00002779 F6E5                <1>       mul    ch
4332                              <1>       ;; 07/08/2016
4333                              <1>       ;movzx eax, ch
4334                              <1>       ;shl   ax, 3 ; * 8 ; * [CHAR_HEIGHT]
4335 0000277B 0FB615[CC6F0000]    <1>       movzx  edx, byte [CRT_COLS] ; = [VGA_COLS]
4336 00002782 F7E2                <1>       mul    edx
4337                              <1>       ; edx  = 0
4338 00002784 88CA                <1>       mov    dl, cl
4339 00002786 01D0                <1>       add    eax, edx
4340 00002788 89C7                <1>       mov    edi, eax
4341                              <1>       ; edi = dest
4342                              <1>       ; for(i=0;i<cheight;i++)
4343                              <1>       ; {
4344                              <1>       ;  memsetb(0xa000,dest+i*nbcols,attr,cols);
4345                              <1>       ; }
4346 0000278A 81C700000A00        <1>       add    edi, 0A0000h
4347 00002790 51                  <1>       push   ecx
4348                              <1>       ; 08/08/2016
4349 00002791 8A35[CE6F0000]      <1>       mov    dh, [CHAR_HEIGHT]
4350                              <1>       ;; 07/08/2016
4351                              <1>       ;mov   dh, 8 ; 07/08/2016
4352 00002797 28D2                <1>       sub    dl, dl ; i
4353                              <1> vgamem_fill_pl4_0:
4354 00002799 57                  <1>       push   edi
4355 0000279A 0FB605[CC6F0000]    <1>       movzx  eax, byte [CRT_COLS]
4356 000027A1 F6E2                <1>       mul    dl
4357                              <1>       ; eax = i * nbcols
4358 000027A3 01C7                <1>       add    edi, eax ; dest+i*nbcols
4359 000027A5 88D8                <1>       mov    al, bl ; attr ; 04/08/2016
4360 000027A7 0FB6CF              <1>       movzx  ecx, bh ; cols
4361 000027AA F3AA                <1>       rep    stosb
4362 000027AC 5F                  <1>       pop    edi
4363 000027AD 75EA                <1>       jnz    short vgamem_fill_pl4_0
4364                              <1> vgamem_fill_pl4_1:
4365 000027AF 59                  <1>       pop    ecx
4366                              <1>
```

```
                                <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0005);
000027B0 66B80500               <1>       mov    ax, 0005h
000027B4 66BACE03               <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
000027B8 66EF                   <1>       out    dx, ax
                                <1>
000027BA 58                     <1>       pop    eax
000027BB 5A                     <1>       pop    edx
                                <1>
000027BC 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>
000027BD 52                     <1>       push   edx
000027BE 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>
000027BF 0FB6D4                 <1>       movzx  edx, ah ; ysrc
                                <1>       ; 08/08/2016
000027C2 0FB605[CE6F0000]       <1>       movzx  eax, byte [CHAR_HEIGHT]
000027C9 8A25[CC6F0000]         <1>       mov    ah, [CRT_COLS] ; nbcols
000027CF F6E4                   <1>       mul    ah
                                <1>       ;; 07/08/2016
                                <1>       ;movzx eax, byte [CRT_COLS]
                                <1>       ;shl   ax, 3 ; * 8 ; * [CHAR_HEIGHT]
000027D1 50                     <1>       push   eax ; cheight * nbcols
000027D2 F7E2                   <1>       mul    edx ; * ysrc
                                <1>       ; eax = ysrc * cheight * nbcols
                                <1>       ; edx = 0
000027D4 88CA                   <1>       mov    dl, cl ; edx = xstart
000027D6 01D0                   <1>       add    eax, edx
000027D8 89C6                   <1>       mov    esi, eax ; src
000027DA 66C1E603               <1>       shl    si, 3 ; * 8 ; 06/08/2016
000027DE 88EA                   <1>       mov    dl, ch ; ydest
000027E0 58                     <1>       pop    eax ; cheight * nbcols
000027E1 F7E2                   <1>       mul    edx
                                <1>       ; eax = ydest * cheight * nbcols
000027E3 88CA                   <1>       mov    dl, cl ; edx = xstart
000027E5 01D0                   <1>       add    eax, edx
000027E7 89C7                   <1>       mov    edi, eax ; dest
000027E9 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>       ; }
000027ED 51                     <1>       push   ecx
000027EE B900000A00             <1>       mov    ecx, 0A0000h
000027F3 01CE                   <1>       add    esi, ecx
000027F5 01CF                   <1>       add    edi, ecx
                                <1>       ; 08/08/2016
000027F7 8A35[CE6F0000]         <1>       mov    dh, [CHAR_HEIGHT]
                                <1>       ;; 07/08/2016
                                <1>       ;mov   dh, 8 ; 07/08/2016
000027FD 28D2                   <1>       sub    dl, dl ; i
                                <1> vgamem_copy_l8_0:
000027FF 56                     <1>       push   esi
00002800 57                     <1>       push   edi
00002801 0FB605[CC6F0000]       <1>       movzx  eax, byte [CRT_COLS]
00002808 F6E2                   <1>       mul    dl
                                <1>       ; eax = i * nbcols
0000280A 66C1E003               <1>       shl    ax, 3 ; * 8 ; 06/08/2016
0000280E 01C7                   <1>       add    edi, eax ; dest+i*nbcols
00002810 01C6                   <1>       add    esi, eax
00002812 0FB6CF                 <1>       movzx  ecx, bh ; cols
00002815 66C1E103               <1>       shl    cx, 3 ; * 8 ; 06/08/2016
00002819 F3A4                   <1>       rep    movsb
0000281B 5F                     <1>       pop    edi
0000281C 5E                     <1>       pop    esi
0000281D FEC2                   <1>       inc    dl ; 06/08/2016
0000281F FECE                   <1>       dec    dh
00002821 75DC                   <1>       jnz    short vgamem_copy_l8_0
                                <1> vgamem_copy_l8_1:
00002823 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>
00002824 58                     <1>       pop    eax
```

```
4472 00002825 5A                  <1>       pop    edx
4473                              <1>
4474 00002826 C3                  <1>       retn
4475                              <1>
4476                              <1> vgamem_fill_l8:
4477                              <1>       ; 08/08/2016
4478                              <1>       ; 07/08/2016
4479                              <1>       ; 06/08/2016
4480                              <1>       ; 04/08/2016
4481                              <1>       ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
4482                              <1>       ;
4483                              <1>       ; TEMPORARY
4484                              <1>       ;
4485                              <1>       ; derived from 'Plex86/Bochs VGABios' source code
4486                              <1>       ; vgabios-0.7a (2011)
4487                              <1>       ; by the LGPL VGABios developers Team (2001-2008)
4488                              <1>       ; 'vgabios.c', 'vgamem_fill_pl4'
4489                              <1>       ;
4490                              <1>       ; vgamem_fill_pl4(xstart,ystart,cols,nbcols,cheight,attr)
4491                              <1>       ; cl = xstart, edi = ch = ystart, bh = cols,
4492                              <1>       ; [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight, attr = 0
4493                              <1>
4494                              <1>       ; dest=ystart*cheight*nbcols+xstart;
4495 00002827 52                  <1>       push   edx
4496 00002828 50                  <1>       push   eax
4497                              <1>
4498                              <1>       ;; outw(VGAREG_GRDC_ADDRESS, 0x0205)
4499                              <1>       ;mov   ax, 0205h
4500                              <1>       ;mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
4501                              <1>       ;out   dx, ax
4502                              <1>
4503                              <1>         ; 08/08/2016
4504 00002829 0FB605[CE6F0000]    <1>       movzx   eax, byte [CHAR_HEIGHT]
4505 00002830 F6E5                <1>       mul    ch
4506                              <1>       ;; 07/08/2016
4507                              <1>       ;movzx eax, ch
4508                              <1>       ;shl   ax, 3 ; * 8 ; * [CHAR_HEIGHT]
4509 00002832 0FB615[CC6F0000]    <1>       movzx  edx, byte [CRT_COLS] ; = [VGA_COLS]
4510 00002839 F7E2                <1>       mul    edx
4511                              <1>       ; edx  = 0
4512 0000283B 88CA                <1>       mov    dl, cl
4513 0000283D 01D0                <1>       add    eax, edx
4514 0000283F 89C7                <1>       mov    edi, eax
4515 00002841 66C1E703            <1>       shl    di, 3 ; * 8 ; 06/08/2016
4516                              <1>       ; edi = dest
4517                              <1>       ; for(i=0;i<cheight;i++)
4518                              <1>       ; {
4519                              <1>       ;  memsetb(0xa000,dest+i*nbcols,attr,cols);
4520                              <1>       ; }
4521 00002845 81C700000A00        <1>       add    edi, 0A0000h
4522 0000284B 51                  <1>       push   ecx
4523                              <1>       ; 08/08/2016
4524 0000284C 8A35[CE6F0000]      <1>       mov    dh, [CHAR_HEIGHT]
4525                              <1>       ;; 07/08/2016
4526                              <1>       ;mov   dh, 8 ; 07/08/2016
4527 00002852 28D2                <1>       sub    dl, dl ; i
4528                              <1> vgamem_fill_l8_0:
4529 00002854 57                  <1>       push   edi
4530 00002855 0FB605[CC6F0000]    <1>       movzx  eax, byte [CRT_COLS]
4531 0000285C F6E2                <1>       mul    dl
4532                              <1>       ; eax = i * nbcols
4533 0000285E 66C1E003            <1>       shl    ax, 3 ; * 8 ; 06/08/2016
4534 00002862 01C7                <1>       add    edi, eax ; dest+i*nbcols
4535 00002864 88D8                <1>       mov    al, bl ; attr ; 04/08/2016
4536 00002866 0FB6CF              <1>       movzx  ecx, bh ; cols
4537 00002869 66C1E103            <1>       shl    cx, 3 ; * 8 ; 06/08/2016
4538 0000286D F3AA                <1>       rep    stosb
4539 0000286F 5F                  <1>       pop    edi
4540 00002870 FEC2                <1>       inc    dl ; 06/08/2016
4541 00002872 FECE                <1>       dec    dh
4542 00002874 75DE                <1>       jnz    short vgamem_fill_l8_0
4543                              <1> vgamem_fill_l8_1:
4544 00002876 59                  <1>       pop    ecx
4545                              <1>
4546                              <1>       ;; outw(VGAREG_GRDC_ADDRESS, 0x0005);
4547                              <1>       ;mov   ax, 0005h
4548                              <1>       ;mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
4549                              <1>       ;out   dx, ax
4550                              <1>
4551 00002877 58                  <1>       pop    eax
4552 00002878 5A                  <1>       pop    edx
4553                              <1>
4554 00002879 C3                  <1>       retn
4555                              <1>
4556                              <1> vga_graphics_down:
4557                              <1>       ; 08/08/2016
4558                              <1>       ; 07/08/2016
4559                              <1>       ; 31/07/2016
4560                              <1>       ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
4561                              <1>       ;
4562                              <1>       ; derived from 'Plex86/Bochs VGABios' source code
4563                              <1>       ; vgabios-0.7a (2011)
4564                              <1>       ; by the LGPL VGABios developers Team (2001-2008)
4565                              <1>       ; 'vgabios.c', 'biosfn_scroll'
4566                              <1>       ;
4567                              <1>
4568                              <1>       ; cl = upper left column
4569                              <1>       ; ch = upper left row
4570                              <1>       ; dl = lower rigth column
4571                              <1>       ; dh = lower right row
4572                              <1>       ;
4573                              <1>       ; al = line count (AL=0 means blank entire fields)
4574                              <1>       ; bl = fill value for blanked lines
4575                              <1>       ; bh = unused
4576                              <1>       ;
```

```
4577                                    <1>       ; ah = [CRT_MODE], current video mode
4578                                    <1>
4579 0000287A FC                        <1>         cld     ; !!! Clear direction flag !!!
4580                                    <1>
4581 0000287B 88C7                      <1>       mov   bh, al ; 31/07/2016
4582                                    <1>
4583 0000287D BE[E66F0000]              <1>       mov   esi, vga_modes
4584 00002882 89F7                      <1>       mov   edi, esi
4585 00002884 83C710                    <1>       add   edi, vga_mode_count
4586                                    <1> vga_g_down_0:
4587 00002887 AC                        <1>       lodsb
4588 00002888 38E0                      <1>       cmp   al, ah ; [CRT_MODE]
4589 0000288A 7405                      <1>       je    short vga_g_down_1
4590 0000288C 39FE                      <1>       cmp   esi, edi
4591 0000288E 72F7                      <1>       jb    short vga_g_down_0
4592                                    <1>       ; xor  bh, bh ; 31/07/2016
4593 00002890 C3                        <1>       retn  ; nothing to do
4594                                    <1> vga_g_down_1:
4595 00002891 88F8                      <1>       mov   al, bh ; 31/07/2016
4596 00002893 83C64F                    <1>       add   esi, vga_memmodel - (vga_modes + 1)
4597                                    <1>       ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
4598                                    <1>
4599                                    <1>       ; if(rlr>=nbrows)rlr=nbrows-1;
4600                                    <1>       ; if(clr>=nbcols)clr=nbcols-1;
4601                                    <1>       ; if(nblines>nbrows)nblines=0;
4602                                    <1>       ; cols=clr-cul+1;
4603                                    <1>
4604 00002896 3A35[D26F0000]            <1>       cmp   dh, [VGA_ROWS]
4605 0000289C 7208                      <1>       jb    short vga_g_down_2
4606 0000289E 8A35[D26F0000]            <1>       mov   dh, [VGA_ROWS]
4607 000028A4 FECE                      <1>       dec   dh
4608                                    <1> vga_g_down_2:
4609 000028A6 3A15[CC6F0000]            <1>       cmp   dl, [CRT_COLS]  ; = [VGA_COLS]
4610 000028AC 7208                      <1>       jb    short vga_g_down_3
4611 000028AE 8A15[CC6F0000]            <1>       mov   dl, [CRT_COLS]
4612 000028B4 FECA                      <1>       dec   dl
4613                                    <1> vga_g_down_3:
4614 000028B6 3A05[D26F0000]            <1>       cmp   al, [VGA_ROWS]
4615 000028BC 7602                      <1>       jna   short vga_g_down_4
4616 000028BE 28C0                      <1>       sub   al, al ; 0
4617                                    <1> vga_g_down_4:
4618 000028C0 88F7                      <1>       mov   bh, dh ; clr
4619 000028C2 28CF                      <1>       sub   bh, cl ; cul
4620 000028C4 FEC7                      <1>       inc   bh ; cols = clr-cul+1
4621                                    <1>
4622 000028C6 20C0                      <1>       and   al, al ; nblines = 0
4623 000028C8 755B                      <1>       jnz   short vga_g_down_6
4624 000028CA 20ED                      <1>       and   ch, ch ; rul = 0
4625 000028CC 7557                      <1>       jnz   short vga_g_down_6
4626 000028CE 20C9                      <1>       and   cl, cl ; cul = 0
4627 000028D0 7553                      <1>       jnz   short vga_g_down_6
4628                                    <1>
4629 000028D2 6650                      <1>       push  ax
4630 000028D4 A0[D26F0000]              <1>       mov   al, [VGA_ROWS]
4631 000028D9 FEC8                      <1>       dec   al
4632 000028DB 38C6                      <1>       cmp   dh, al ; rlr = nbrows-1
4633 000028DD 7544                      <1>       jne   short vga_g_down_5
4634 000028DF A0[CC6F0000]              <1>         mov     al, [CRT_COLS]  ; = VGA_COLS
4635 000028E4 FEC8                      <1>       dec   al
4636 000028E6 38C2                      <1>       cmp   dl, al ; clr = nbcols-1
4637 000028E8 7539                      <1>       jne   short vga_g_down_5
4638 000028EA 6658                      <1>       pop   ax
4639                                    <1>
4640 000028EC 66B80502                  <1>       mov   ax, 0205h
4641 000028F0 66BACE03                  <1>       mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
4642 000028F4 66EF                      <1>       out   dx, ax
4643 000028F6 A0[D26F0000]              <1>       mov   al, [VGA_ROWS]
4644 000028FB 8A25[CC6F0000]            <1>       mov   ah, [CRT_COLS] ; = [VGA_COLS]
4645 00002901 F6E4                      <1>       mul   ah
4646 00002903 0FB7D0                    <1>       movzx edx, ax
4647                                    <1>       ; 08/08/2016
4648 00002906 0FB605[CE6F0000]          <1>       movzx eax, byte [CHAR_HEIGHT]
4649 0000290D F7E2                      <1>       mul   edx
4650                                    <1>       ; eax = byte count
4651 0000290F 89C1                      <1>       mov   ecx, eax
4652                                    <1>       ;; 07/08/2016
4653                                    <1>       ;shl   dx, 3 ; * 8 ; * [CHAR_HEIGHT]
4654                                    <1>       ;mov   ecx, edx
4655 00002911 88D8                      <1>       mov   al, bl ; fill value for blanked lines
4656 00002913 BF00000A00                <1>       mov   edi, 0A0000h
4657 00002918 F3AA                      <1>       rep   stosb
4658                                    <1>
4659 0000291A B005                      <1>       mov   al, 5
4660 0000291C 66BACE03                  <1>       mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
4661 00002920 66EF                      <1>       out   dx, ax ; 0005h
4662                                    <1>
4663 00002922 C3                        <1>       retn
4664                                    <1>
4665                                    <1> vga_g_down_5:
4666 00002923 6658                      <1>       pop   ax
4667                                    <1>
4668                                    <1> vga_g_down_6:
4669                                    <1>       ; [ESI] = VGA memory model number for current video mode
4670                                    <1>         ;
4671                                    <1>         ; LINEAR8 equ 5
4672                                    <1>         ; PLANAR4 equ 4
4673                                    <1>         ; PLANAR1 equ 3
4674                                    <1>
4675 00002925 803E04                    <1>       cmp   byte [esi], PLANAR4
4676 00002928 742C                      <1>       je    short vga_g_down_planar
4677 0000292A 803E03                    <1>       cmp   byte [esi], PLANAR1
4678 0000292D 7427                      <1>       je    short vga_g_down_planar
4679                                    <1> vga_g_down_linear8:
4680                                    <1>       ; 07/07/2016 (TEMPORARY)
4681                                    <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))
0000292F 08C0                   <1>       or     al, al
00002931 741C                   <1>       jz     short vga_g_down_l2
00002933 88C4                   <1>       mov    ah, al
00002935 00EC                   <1>       add    ah, ch
                                <1>       ;jc    short vga_g_down_l2
00002937 86EE                   <1>       xchg   ch, dh
00002939 38E5                   <1>       cmp    ch, ah
0000293B 7212                   <1>       jb     short vga_g_down_l2
0000293D 88EC                   <1>       mov    ah, ch
0000293F 28C4                   <1>       sub    ah, al ; ah = i - nblines
                                <1>       ; else
                                <1>       ; vgamem_copy_pl4(cul,i,i-nblines,cols,nbcols,cheight);
00002941 E877FEFFFF             <1>       call   vgamem_copy_l8
                                <1> vga_g_down_l1:
00002946 86F5                   <1>       xchg   dh, ch
00002948 FECE                   <1>       dec    dh
0000294A 38EE                   <1>       cmp    dh, ch
0000294C 73E1                   <1>       jnb    short vga_g_down_l0
0000294E C3                     <1>       retn
                                <1>
                                <1> vga_g_down_l2:
                                <1>       ; vgamem_fill_pl4(cul,i,cols,nbcols,cheight,attr);
0000294F E8D3FEFFFF             <1>       call   vgamem_fill_l8
00002954 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))
00002956 08C0                   <1>       or     al, al
00002958 741C                   <1>       jz     short vga_g_down_pl2
0000295A 88C4                   <1>       mov    ah, al
0000295C 00EC                   <1>       add    ah, ch
                                <1>       ;jc    short vga_g_down_pl2
0000295E 86EE                   <1>       xchg   ch, dh
00002960 38E5                   <1>       cmp    ch, ah
00002962 7212                   <1>       jb     short vga_g_down_pl2
00002964 88EC                   <1>       mov    ah, ch
00002966 28C4                   <1>       sub    ah, al ; ah = i - nblines
                                <1>       ; else
                                <1>       ; vgamem_copy_pl4(cul,i,i-nblines,cols,nbcols,cheight);
00002968 E88BFDFFFF             <1>       call   vgamem_copy_pl4
                                <1> vga_g_down_pl1:
0000296D 86F5                   <1>       xchg   dh, ch
0000296F FECE                   <1>       dec    dh
00002971 38EE                   <1>       cmp    dh, ch
00002973 73E1                   <1>       jnb    short vga_g_down_pl0
00002975 C3                     <1>       retn
                                <1>
                                <1> vga_g_down_pl2:
                                <1>       ; vgamem_fill_pl4(cul,i,cols,nbcols,cheight,attr);
00002976 E8EBFDFFFF             <1>       call   vgamem_fill_pl4
0000297B 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
0000297D 80FC07                 <1>       cmp    ah, 7
00002980 0F87F4FEFFFF           <1>         ja     vga_graphics_down
                                <1>       ;je    _n0
                                <1>
00002986 88C7                   <1>       MOV    bh, al             ; save line count in BH
00002988 6689D0                 <1>       MOV    AX, DX             ; GET LOWER RIGHT POSITION INTO AX REG
                                <1>
                                <1> ;-----      USE CHARACTER SUBROUTINE FOR POSITIONING
```

```
4787                                 <1> ;-----      ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
4788                                 <1>
4789 0000298B E8F2010000             <1>       CALL   GRAPH_POSN
4790 00002990 0FB7F8                 <1>       MOVzx  eDI, AX                  ; SAVE RESULT AS DESTINATION ADDRESS
4791                                 <1>
4792                                 <1> ;-----      DETERMINE SIZE OF WINDOW
4793                                 <1>
4794 00002993 6629CA                 <1>       SUB    DX, CX
4795 00002996 6681C20101             <1>         ADD     DX, 101h               ; ADJUST VALUES
4796 0000299B C0E602                 <1>       SAL    DH, 2               ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
4797                                 <1>                                  ; AND EVEN/ODD ROWS
4798                                 <1>
4799                                 <1> ;-----      DETERMINE CRT MODE
4800                                 <1>
4801 0000299E 803D[CA6F0000]06       <1>       CMP    byte [CRT_MODE], 6  ; TEST FOR MEDIUM RES
4802 000029A5 7307                   <1>         JNC     short _R12             ; FIND_SOURCE_DOWN
4803                                 <1>
4804                                 <1> ;-----      MEDIUM RES DOWN
4805 000029A7 D0E2                   <1>       SAL    DL, 1               ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
4806 000029A9 66D1E7                 <1>       SAL    DI, 1               ; OFFSET *2 SINCE 2 BYTES/CHAR
4807 000029AC 6647                   <1>       INC    DI                  ; POINT TO LAST BYTE
4808                                 <1>
4809                                 <1> ;-----      DETERMINE THE SOURCE ADDRESS IN THE BUFFER
4810                                 <1>
4811                                 <1> _R12:                                   ; FIND_SOURCE_DOWN
4812 000029AE 81C700800B00           <1>       add    edi, 0B8000h
4813 000029B4 6681C7F000             <1>       ADD    DI, 240                  ; POINT TO LAST ROW OF PIXELS
4814 000029B9 C0E702                 <1>       sal    bh, 2               ; multiply number of lines by 4
4815 000029BC 74(06)                 <1>       JZ     short 6                  ; IF ZERO, THEN BLANK ENTIRE FIELD
4816 000029BE B050                   <1>       MOV    AL, 80              ; 80 BYTES/ROW
4817 000029C0 F6E7                   <1>       mul    bh                  ; determine offset to source
4818 000029C2 89FE                   <1>       MOV    eSI, eDI            ; SET UP SOURCE
4819 000029C4 6629C6                 <1>       SUB    SI, AX              ; SUBTRACT THE OFFSET
4820 000029C7 88F4                   <1>       MOV    AH, DH              ; NUMBER OF ROWS IN FIELD
4821 000029C9 28FC                   <1>       sub    ah, bh              ; determine number to move
4822                                 <1>
4823                                 <1> ;-----      LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
4824                                 <1>
4825                                 <1> _R13:                                   ; ROW_LOOP_DOWN
4826 000029CB E824000000             <1>         CALL    _R17                   ; MOVE ONE ROW
4827 000029D0 6681EE5020             <1>       SUB    SI, 2000h+80         ; MOVE TO NEXT ROW
4828 000029D5 6681EF5020             <1>       SUB    DI, 2000h+80
4829 000029DA FECC                   <1>       DEC    AH                  ; NUMBER OF ROWS TO MOVE
4830 000029DC 75ED                   <1>         JNZ     short _R13             ; CONTINUE TILL ALL MOVED
4831                                 <1>
4832                                 <1> ;-----      FILL IN THE VACATED LINE(S)
4833                                 <1> _R14:                                   ; CLEAR_ENTRY_DOWN
4834 000029DE 88D8                   <1>       mov    al, bl              ; attribute to fill with
4835                                 <1> _R15_:                                  ; CLEAR_LOOP_DOWN
4836 000029E0 E82B000000             <1>       CALL   _R18                ; CLEAR A ROW
4837 000029E5 6681EF5020             <1>       SUB    DI, 2000h+80         ; POINT TO NEXT LINE
4838 000029EA FECF                   <1>       dec    bh                  ; number of lines to fill
4839 000029EC 75F2                   <1>         JNZ     short _R15_            ; CLEAR_LOOP_DOWN
4840                                 <1>       ; 18/11/2020
4841 000029EE FC                     <1>       CLD                        ; RESET THE DIRECTION FLAG
4842                                 <1>
4843 000029EF C3                     <1>       retn                       ; EVERYTHING DONE
4844                                 <1>
4845                                 <1> _R16:                                   ; BLANK_FIELD_DOWN
4846 000029F0 88F7                   <1>       mov    bh, dh              ; set blank count to everything in field
4847 000029F2 EBEA                   <1>         JMP     short _R14             ; CLEAR THE FIELD
4848                                 <1>
4849                                 <1> ; 27/06/2016 - TRDOS 386 (TRDOS v2.0)
4850                                 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
4851                                 <1>
4852                                 <1> ;-----      ROUTINE TO MOVE ONE ROW OF INFORMATION
4853                                 <1>
4854                                 <1> _R17:
4855 000029F4 0FB6CA                 <1>       MOVzx  ecx, DL                  ; NUMBER OF BYTES IN THE ROW
4856 000029F7 56                     <1>       PUSH   eSI
4857 000029F8 57                     <1>       PUSH   eDI                 ; SAVE POINTERS
4858 000029F9 F3A4                   <1>       REP    MOVSB               ; MOVE THE EVEN FIELD
4859 000029FB 5F                     <1>       POP    eDI
4860 000029FC 5E                     <1>       POP    eSI
4861 000029FD 6681C60020             <1>       ADD    SI, 2000h
4862 00002A02 6681C70020             <1>       ADD    DI, 2000h           ; POINT TO THE ODD FIELD
4863 00002A07 56                     <1>       PUSH   eSI
4864 00002A08 57                     <1>       PUSH   eDI                 ; SAVE THE POINTERS
4865 00002A09 88D1                   <1>       MOV    CL, DL              ; COUNT BACK
4866 00002A0B F3A4                   <1>       REP    MOVSB               ; MOVE THE ODD FIELD
4867 00002A0D 5F                     <1>       POP    eDI
4868 00002A0E 5E                     <1>       POP    eSI                 ; POINTERS BACK
4869 00002A0F C3                     <1>       RETn                       ; RETURN TO CALLER
4870                                 <1>
4871                                 <1> ;-----      CLEAR A SINGLE ROW
4872                                 <1>
4873                                 <1> _R18:
4874 00002A10 0FB6CA                 <1>       MOVzx  ecx, DL                  ; NUMBER OF BYTES IN FIELD
4875 00002A13 57                     <1>       PUSH   eDI                 ; SAVE POINTER
4876 00002A14 F3AA                   <1>       REP    STOSB               ; STORE THE NEW VALUE
4877 00002A16 5F                     <1>       POP    eDI                 ; POINTER BACK
4878 00002A17 6681C70020             <1>       ADD    DI, 2000h           ; POINT TO ODD FIELD
4879 00002A1C 57                     <1>       PUSH   eDI
4880 00002A1D 88D1                   <1>       MOV    CL, DL
4881 00002A1F F3AA                   <1>       REP    STOSB               ; FILL THE ODD FIELD
4882 00002A21 5F                     <1>       POP    eDI
4883 00002A22 C3                     <1>       RETn                       ; RETURN TO CALLER
4884                                 <1>
4885                                 <1> ; 04/07/2016
4886                                 <1> ; 01/07/2016
4887                                 <1> ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)
4888                                 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
4889                                 <1> ;-------------------------------------------------
4890                                 <1> ; GRAPHICS WRITE
4891                                 <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:
00002A23 25FF000000     <1>       and   eax, 0FFh           ; ZERO TO HIGH OF CODE POINT
00002A28 50             <1>       PUSH  eAX                 ; SAVE CODE POINT VALUE
                        <1>
                        <1> ;-----      DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
                        <1>
00002A29 E84D010000     <1>       CALL  S26                 ; FIND LOCATION IN REGEN BUFFER
00002A2E 89C7           <1>       MOV   eDI, eAX            ; REGEN POINTER IN DI
                        <1>
                        <1> ;-----      DETERMINE REGION TO GET CODE POINTS FROM
                        <1>
00002A30 58             <1>       POP   eAX                 ; RECOVER CODE POINT
                        <1>
00002A31 BE[54600100]   <1>       MOV   eSI, CRT_CHAR_GEN   ; OFFSET OF IMAGES
                        <1>
                        <1> ;-----      DETERMINE GRAPHICS MODE IN OPERATION
                        <1>                                 ; DETERMINE_MODE
00002A36 66C1E003       <1>       SAL   AX, 3               ; MULTIPLY CODE POINT VALUE BY 8
00002A3A 01C6           <1>       ADD   eSI, eAX            ; SI HAS OFFSET OF DESIRED CODES
                        <1>
00002A3C 803D[CA6F0000]06 <1>     CMP   byte [CRT_MODE], 6
00002A43 7231           <1>       JC    short S6            ; TEST FOR MEDIUM RESOLUTION MODE
                        <1>
                        <1> ;-----      HIGH RESOLUTION MODE
                        <1>
00002A45 81C700800B00   <1>       add   edi, 0B8000h
                        <1> S1:                             ; HIGH_CHAR
00002A4B 57             <1>       PUSH  eDI                 ; SAVE REGEN POINTER
00002A4C 56             <1>       PUSH  eSI                 ; SAVE CODE POINTER
00002A4D B604           <1>       MOV   DH, 4               ; NUMBER OF TIMES THROUGH LOOP
                        <1> S2:
00002A4F AC             <1>       LODSB                     ; GET BYTE FROM CODE POINTS
00002A50 F6C380         <1>       TEST  BL, 80H                   ; SHOULD WE USE THE FUNCTION
00002A53 7515           <1>       JNZ   short S5            ; TO PUT CHAR IN
00002A55 AA             <1>       STOSB                     ; STORE IN REGEN BUFFER
00002A56 AC             <1>       LODSB
                        <1> S4:
00002A57 8887FF1F0000   <1>       MOV   [eDI+2000H-1], AL ; STORE IN SECOND HALF
00002A5D 83C74F         <1>       ADD   eDI, 79                   ; MOVE TO NEXT ROW IN REGEN
00002A60 FECE           <1>       DEC   DH                  ; DONE WITH LOOP
00002A62 75EB           <1>       JNZ   short S2
00002A64 5E             <1>       POP   eSI
00002A65 5F             <1>       POP   eDI                 ; RECOVER REGEN POINTER
00002A66 47             <1>       INC   eDI                 ; POINT TO NEXT CHAR POSITION
00002A67 E2E2           <1>       LOOP  S1                  ; MORE CHARS TO WRITE
00002A69 C3             <1>       retn
                        <1>
                        <1> S5:
00002A6A 3207           <1>       XOR   AL, [eDI]           ; EXCLUSIVE OR WITH CURRENT
00002A6C AA             <1>       STOSB                     ; STORE THE CODE POINT
00002A6D AC             <1>       LODSB                     ; AGAIN FOR ODD FIELD
00002A6E 3287FF1F0000   <1>       XOR   AL, [eDI+2000H-1]
00002A74 EBE1           <1>       JMP   short S4            ; BACK TO MAINSTREAM
                        <1>
                        <1> ;-----      MEDIUM RESOLUTION WRITE
                        <1> S6:                             ; MED_RES_WRITE
00002A76 88DA           <1>       MOV   DL, BL              ; SAVE HIGH COLOR BIT
00002A78 66D1E7         <1>       SAL   DI, 1               ; OFFSET*2 SINCE 2 BYTES/CHAR
                        <1>                                 ; EXPAND BL TO FULL WORD OF COLOR
00002A7B 80E303         <1>       AND   BL, 3               ; ISOLATE THE COLOR BITS ( LOW 2 BITS )
00002A7E B055           <1>       MOV   AL, 055H            ; GET BIT CONVERSION MULTIPLIER
00002A80 F6E3           <1>       MUL   BL                  ; EXPAND 2 COLOR BITS TO 4 REPLICATIONS
00002A82 88C3           <1>       MOV   BL, AL              ; PLACE BACK IN WORK REGISTER
00002A84 88C7           <1>       MOV   BH, AL              ; EXPAND TO 8 REPLICATIONS OF COLOR BITS
00002A86 81C700800B00   <1>       add   edi, 0B8000h
                        <1> S7:                                   ; MED_CHAR
00002A8C 57             <1>       PUSH  eDI                 ; SAVE REGEN POINTER
00002A8D 56             <1>       PUSH  eSI                 ; SAVE THE CODE POINTER
00002A8E B604           <1>       MOV   DH, 4               ; NUMBER OF LOOPS
                        <1> S8:
00002A90 AC             <1>       LODSB                     ; GET CODE POINT
00002A91 E8B3000000     <1>       CALL  S21                 ; DOUBLE UP ALL THE BITS
00002A96 6621D8         <1>       AND   AX, BX              ; CONVERT TO FOREGROUND COLOR ( 0 BACK )
00002A99 86E0           <1>       XCHG  AH, AL              ; SWAP HIGH/LOW BYTES FOR WORD MOVE
00002A9B F6C280         <1>       TEST  DL, 80H                   ; IS THIS XOR FUNCTION
00002A9E 7403           <1>       JZ    short S9            ; NO, STORE IT IN AS IS
00002AA0 663307         <1>       XOR   AX, [eDI]           ; DO FUNCTION WITH LOW/HIGH
```

```
                                <1> S9:
00002AA3 668907                 <1>         MOV     [eDI], AX           ; STORE FIRST BYTE HIGH, SECOND LOW
00002AA6 AC                     <1>         LODSB                       ; GET CODE POINT
00002AA7 E89D000000             <1>         CALL    S21
00002AAC 6621D8                 <1>         AND     AX, BX              ; CONVERT TO COLOR
00002AAF 86E0                   <1>         XCHG    AH, AL              ; SWAP HIGH/LOW BYTES FOR WORD MOVE
00002AB1 F6C280                 <1>         TEST    DL, 80H                     ; AGAIN, IS THIS XOR FUNCTION
00002AB4 7407                   <1>         JZ      short _S10          ; NO, JUST STORE THE VALUES
00002AB6 66338700200000         <1>         XOR     AX, [eDI+2000H]             ; FUNCTION WITH FIRST HALF LOW
                                <1> _S10:
00002ABD 66898700200000         <1>         MOV     [eDI+2000H], AX             ; STORE SECOND PORTION HIGH
00002AC4 6683C750               <1>         ADD     DI, 80              ; POINT TO NEXT LOCATION
00002AC8 FECE                   <1>         DEC     DH
00002ACA 75C4                   <1>         JNZ     short S8            ; KEEP GOING
00002ACC 5E                     <1>         POP     eSI                 ; RECOVER CODE POINTER
00002ACD 5F                     <1>         POP     eDI                 ; RECOVER REGEN POINTER
00002ACE 47                     <1>         INC     eDI                 ; POINT TO NEXT CHAR POSITION
00002ACF 47                     <1>         INC     eDI
00002AD0 E2BA                   <1>         LOOP    S7                  ; MORE TO WRITE
00002AD2 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:
00002AD3 E8A3000000             <1>         CALL    S26                 ; CONVERTED TO OFFSET IN REGEN
00002AD8 89C6                   <1>         MOV     eSI, eAX            ; SAVE IN SI
00002ADA 81C600800B00           <1>         add     esi, 0B8000h        ; 01/07/2016
00002AE0 83EC08                 <1>         SUB     eSP, 8              ; ALLOCATE SPACE FOR THE READ CODE POINT
00002AE3 89E5                   <1>         MOV     eBP, eSP            ; POINTER TO SAVE AREA
                                <1>
                                <1> ;-----      DETERMINE GRAPHICS MODES
00002AE5 B604                   <1>         mov     dh, 4               ; number of passes ; 01/07/2016
00002AE7 803D[CA6F0000]06       <1>         CMP     byte [CRT_MODE], 6
00002AEE 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:
00002AF0 8A06                   <1>         MOV     AL, [eSI]           ; GET FIRST BYTE
00002AF2 884500                 <1>         MOV     [eBP], AL           ; SAVE IN STORAGE AREA
00002AF5 45                     <1>         INC     eBP                 ; NEXT LOCATION
00002AF6 8A8600200000           <1>         MOV     AL, [eSI+2000H]             ; GET LOWER REGION BYTE
00002AFC 884500                 <1>         MOV     [eBP], AL           ; ADJUST AND STORE
00002AFF 45                     <1>         INC     eBP
00002B00 83C650                 <1>         ADD     eSI, 80                     ; POINTER INTO REGEN
00002B03 FECE                   <1>         DEC     DH                  ; LOOP CONTROL
00002B05 75E9                   <1>         JNZ     short S11           ; DO IT SOME MORE
00002B07 EB1D                   <1>         JMP     SHORT S14           ; GO MATCH THE SAVED CODE POINTS
                                <1>
                                <1> ;-----      MEDIUM RESOLUTION READ
                                <1> S12:
00002B09 66D1E6                 <1>         SAL     SI, 1               ; OFFSET*2 SINCE 2 BYTES/CHAR
                                <1>         ;MOV    DH, 4               ; NUMBER OF PASSES
                                <1> S13:
00002B0C E84D000000             <1>         CALL    S23                 ; GET BYTES FROM REGEN INTO SINGLE SAVE
00002B11 81C6FE1F0000           <1>         ADD     eSI, 2000H-2        ; GO TO LOWER REGION
00002B17 E842000000             <1>         CALL    S23                 ; GET THIS PAIR INTO SAVE
00002B1C 81EEB21F0000           <1>         SUB     eSI, 2000H-80+2             ; ADJUST POINTER BACK INTO UPPER
00002B22 FECE                   <1>         DEC     DH
00002B24 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
00002B26 BF[54600100]           <1>         MOV     eDI, CRT_CHAR_GEN   ; ESTABLISH ADDRESSING
00002B2B 83ED08                 <1>         SUB     eBP, 8              ; ADJUST POINTER TO START OF SAVE AREA
00002B2E 89EE                   <1>         MOV     eSI, eBP
                                <1> S15:
00002B30 66B80001               <1>         mov     ax, 256                     ; NUMBER TO TEST AGAINST
                                <1> S16:
00002B34 56                     <1>         PUSH    eSI                 ; SAVE SAVE AREA POINTER
00002B35 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
00002B36 A7                     <1>         cmpsd                       ; compare first 4 bytes
00002B37 7501                   <1>         jne     short S17           ;
00002B39 A7                     <1>         cmpsd                       ; compare last 4 bytes
                                <1> S17:
00002B3A 5F                     <1>         POP     eDI                 ; RECOVER THE POINTERS
00002B3B 5E                     <1>         POP     eSI
                                <1>         ;JZ     short S18           ; IF ZERO FLAG SET, THEN MATCH OCCURRED
00002B3C 7407                   <1>         je      short S18
                                <1>         ;                           ; NO MATCH, MOVE ON TO NEXT
00002B3E 83C708                 <1>         ADD     eDI, 8              ; NEXT CODE POINT
00002B41 6648                   <1>         dec     ax                  ; LOOP CONTROL
00002B43 75EF                   <1>         JNZ     short S16           ; DO ALL OF THEM
                                <1>
                                <1> ;-----      CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
                                <1> S18:
00002B45 83C408                 <1>         ADD     eSP, 8              ; READJUST THE STACK, THROW AWAY SAVE
00002B48 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:
00002B49 6651           <1>       PUSH   CX                ; SAVE REGISTER
                        <1>       ;MOV   CX, 8             ; SHIFT COUNT REGISTER FOR ONE BYTE
00002B4B B108           <1>       mov    cl, 8
                        <1> S22:
00002B4D D0C8           <1>       ROR    AL,1              ; SHIFT BITS, LOW BIT INTO CARRY FLAG
00002B4F 66D1DD         <1>       RCR    BP,1              ; MOVE CARRY FLAG (LOW BIT INTO RESULTS
00002B52 66D1FD         <1>       SAR    BP,1              ; SIGN EXTEND HIGH BIT (DOUBLE IT)
                        <1>       ;LOOP  S22               ; REPEAT FOR ALL 8 BITS
00002B55 FEC9           <1>       dec    cl
00002B57 75F4           <1>       jnz    short S22
00002B59 6695           <1>       XCHG   AX, BP            ; MOVE RESULTS TO PARAMETER REGISTER
00002B5B 6659           <1>       POP    CX                ; RECOVER REGISTER
00002B5D 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:
00002B5E 66AD           <1>       LODSW                    ; GET FIRST BYTE AND SECOND BYTES
00002B60 86C4           <1>       XCHG   AL, AH            ; SWAP FOR COMPARE
00002B62 66B900C0       <1>       MOV    CX, 0C000H        ; 2 BIT MASK TO TEST THE ENTRIES
00002B66 B200           <1>       MOV    DL, 0             ; RESULT REGISTER
                        <1> S24:
00002B68 6685C8         <1>       TEST   AX, CX            ; IS THIS SECTION BACKCROUND?
00002B6B 7401           <1>         JZ     short S25             ; IF ZERO, IT IS BACKGROUND (CARRY=0)
00002B6D F9             <1>       STC                      ; WASN'T, SO SET CARRY
                        <1> S25:
00002B6E D0D2           <1>       RCL    DL, 1             ; MOVE THAT BIT INTO THE RESULT
00002B70 66C1E902       <1>       SHR    CX, 2             ; MOVE THE MASK TO THE RIGHT BY 2 BITS
00002B74 73F2           <1>       JNC    short S24         ; DO IT AGAIN IF MASK DIDN'T FALL OUT
00002B76 885500         <1>       MOV    [eBP], DL         ; STORE RESULT IN SAVE AREA
00002B79 45             <1>       INC    eBP               ; ADJUST POINTER
00002B7A 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:
00002B7B 0FB705[DE890100]   <1>       movzx  eax, word [CURSOR_POSN]   ; GET CURRENT CURSOR
                        <1> GRAPH_POSN:
00002B82 53             <1>       PUSH   eBX               ; SAVE REGISTER
00002B83 0FB6D8         <1>       movzx  ebx, al                 ; SAVE A COPY OF CURRENT CURSOR
00002B86 A0[CC6F0000]       <1>       MOV    AL, [CRT_COLS]          ; GET BYTES PER COLUMN
00002B8B F6E4           <1>       MUL    AH                ; MULTIPLY BY ROWS
00002B8D 66C1E002       <1>       SHL    AX, 2             ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
00002B91 01D8           <1>       ADD    eAX, eBX          ; DETERMINE OFFSET
00002B93 5B             <1>       POP    eBX               ; RECOVER POINTER
00002B94 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:
00002B95 803D[CA6F0000]07   <1>         cmp     byte [CRT_MODE], 7      ; 09/07/2016
00002B9C 0F87B5EFFFFF       <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
00002BA2 66BAD903           <1>       mov    dx, 3D9h
00002BA6 A0[CD6F0000]       <1>       MOV    AL, [CRT_PALETTE]   ; GET THE CURRENT PALETTE VALUE
00002BAB 08FF           <1>       OR     BH, BH            ; IS THIS COLOR 0?
00002BAD 7512           <1>       JNZ    short M20         ; OUTPUT COLOR 1
                        <1>
                        <1> ;-----      HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
                        <1>
```

```
5207 00002BAF 24E0                <1>       AND    AL, 0E0H            ; TURN OFF LOW 5 BITS OF CURRENT
5208 00002BB1 80E31F              <1>       AND    BL, 01FH            ; TURN OFF HIGH 3 BITS OF INPUT VALUE
5209 00002BB4 08D8                <1>       OR     AL, BL              ; PUT VALUE INTO REGISTER
5210                              <1> M19:                             ; OUTPUT THE PALETTE
5211 00002BB6 EE                  <1>       OUT    DX, AL              ; OUTPUT COLOR SELECTION TO 3D9 PORT
5212 00002BB7 A2[CD6F0000]        <1>       MOV    [CRT_PALETTE], AL   ; SAVE THE COLOR VALUE
5213 00002BBC E996EFFFFF          <1>       JMP    VIDEO_RETURN
5214                              <1>
5215                              <1> ;-----      HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED
5216                              <1>
5217                              <1> M20:
5218 00002BC1 24DF                <1>       AND    AL, 0DFH            ; TURN OFF PALETTE SELECT BIT
5219 00002BC3 D0EB                <1>       SHR    BL, 1               ; TEST THE LOW ORDER BIT OF BL
5220 00002BC5 73EF                <1>       JNC    short M19           ; ALREADY DONE
5221 00002BC7 0C20                <1>       OR     AL, 20H                   ; TURN ON PALETTE SELECT BIT
5222 00002BC9 EBEB                <1>       JMP    short M19           ; GO DO IT
5223                              <1>
5224                              <1> ; 09/07/2016
5225                              <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
5226                              <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
5227                              <1> ;--------------------------------------------
5228                              <1> ; READ DOT -- WRITE DOT
5229                              <1> ; THESE ROUTINES WILL WRITE A DOT, OR READ THE
5230                              <1> ;  DOT AT THE INDICATED LOCATION
5231                              <1> ; ENTRY --
5232                              <1> ;   DX = ROW (0-199)      (THE ACTUAL VALUE DEPENDS ON THE MODE)
5233                              <1> ;   CX = COLUMN ( 0-639) ( THE VALUES ARE NOT RANGE CHECKED )
5234                              <1> ;   AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
5235                              <1> ;     REQUIRED FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
5236                              <1> ;     BIT 7 OF AL = 1 INDICATES XOR THE VALUE INTO THE LOCATION
5237                              <1> ;   DS = DATA SEGMENT
5238                              <1> ;   ES = REGEN SEGMENT
5239                              <1> ;
5240                              <1> ; EXIT
5241                              <1> ;     AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
5242                              <1> ;--------------------------------------------
5243                              <1>
5244                              <1> READ_DOT:
5245                              <1>       ; 09/07/2016
5246 00002BCB 8A25[CA6F0000]      <1>       mov    ah,  [CRT_MODE]
5247 00002BD1 80FC07              <1>       cmp    ah, 7 ; 6!?
5248 00002BD4 760A                <1>       jna    short read_dot_cga
5249                              <1>
5250 00002BD6 E8D7030000          <1>       call   vga_read_pixel
5251                              <1>       ; al = pixel value
5252 00002BDB E97CEFFFFF          <1>       jmp    _video_return
5253                              <1>
5254                              <1> read_dot_cga:
5255                              <1>       ;je    VIDEO_RETURN ; 7
5256 00002BE0 80FC04              <1>       cmp    ah, 4 ; graphics ?
5257 00002BE3 0F826EEFFFFF        <1>       jb     VIDEO_RETURN ; no, text mode, nothing to do
5258                              <1>
5259 00002BE9 E855000000          <1>       CALL   R3                  ; DETERMINE BYTE POSITION OF DOT
5260 00002BEE 8A06                <1>       MOV    AL, [eSI]           ; GET THE BYTE
5261 00002BF0 20E0                <1>       AND    AL, AH              ; MASK OFF THE OTHER BITS IN THE BYTE
5262 00002BF2 D2E0                <1>       SHL    AL, CL              ; LEFT JUSTIFY THE VALUE
5263 00002BF4 88F1                <1>       MOV    CL, DH              ; GET NUMBER OF BITS IN RESULT
5264 00002BF6 D2C0                <1>       ROL    AL, CL              ; RIGHT JUSTIFY THE RESULT
5265                              <1>       ;JMP   VIDEO_RETURN        ; RETURN FROM VIDEO I/O
5266 00002BF8 0FB6C0              <1>       movzx  eax, al
5267 00002BFB E95CEFFFFF          <1>       jmp    _video_return
5268                              <1>
5269                              <1> ; 09/07/2016
5270                              <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
5271                              <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
5272                              <1>
5273                              <1> WRITE_DOT:
5274                              <1>       ; 09/07/2016
5275 00002C00 8A25[CA6F0000]      <1>       mov    ah, [CRT_MODE]
5276 00002C06 80FC07              <1>       cmp    ah, 7 ; 6!?
5277 00002C09 760A                <1>       jna    short write_dot_cga
5278                              <1>
5279 00002C0B E811030000          <1>       call   vga_write_pixel
5280 00002C10 E942EFFFFF          <1>       jmp    VIDEO_RETURN
5281                              <1>
5282                              <1> write_dot_cga:
5283                              <1>       ;je    VIDEO_RETURN ; 7
5284 00002C15 80FC04              <1>       cmp    ah, 4 ; graphics ?
5285 00002C18 0F8239EFFFFF        <1>       jb     VIDEO_RETURN ; no, text mode, nothing to do
5286                              <1>
5287                              <1>       ;PUSH  AX                  ; SAVE DOT VALUE
5288 00002C1E 6650                <1>       PUSH   AX                  ; TWICE
5289 00002C20 E81E000000          <1>       CALL   R3                  ; DETERMINE BYTE POSITION OF THE DOT
5290 00002C25 D2E8                <1>       SHR    AL, CL              ; SHIFT TO SET UP THE BITS FOR OUTPUT
5291 00002C27 20E0                <1>       AND    AL, AH              ; STRIP OFF THE OTHER BITS
5292 00002C29 8A0E                <1>       MOV    CL, [eSI]           ; GET THE CURRENT BYTE
5293 00002C2B 665B                <1>       POP    BX                  ; RECOVER XOR FLAG
5294 00002C2D F6C380              <1>       TEST   BL, 80H                   ; IS IT ON
5295 00002C30 750D                <1>       JNZ    short R2            ; YES, XOR THE DOT
5296 00002C32 F6D4                <1>       NOT    AH                  ; SET MASK TO REMOVE THE INDICATED BITS
5297 00002C34 20E1                <1>       AND    CL, AH
5298 00002C36 08C8                <1>       OR     AL, CL              ; OR IN THE NEW VALUE OF THOSE BITS
5299                              <1> R1:                              ; FINISH_DOT
5300 00002C38 8806                <1>       MOV    [eSI], AL           ; RESTORE THE BYTE IN MEMORY
5301                              <1>       ;POP   AX
5302 00002C3A E918EFFFFF          <1>       JMP    VIDEO_RETURN        ; RETURN FROM VIDEO I/O
5303                              <1> R2:                              ; XOR_DOT
5304 00002C3F 30C8                <1>       XOR    AL, CL              ; EXCLUSIVE OR THE DOTS
5305 00002C41 EBF5                <1>       JMP    short R1            ; FINISH UP THE WRITING
5306                              <1>
5307                              <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
5308                              <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
5309                              <1>
5310                              <1> ;--------------------------------------------
5311                              <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>
00002C43 0FB7F0         <1>       movzx  esi, ax                  ; WILL SAVE AL AND AH DURING OPERATION
00002C46 B028           <1>       MOV    AL, 40
00002C48 F6E2           <1>       MUL    DL                  ; AX= ADDRESS OF START OF INDICATED ROW
00002C4A A808           <1>       TEST   AL, 08H             ; TEST FOR EVEN/ODD ROW CALCULATED
00002C4C 7404           <1>       JZ     short R4            ; JUMP IF EVEN ROW
00002C4E 6605D81F       <1>       ADD    AX, 2000H-40        ; OFFSET TO LOCATION OF ODD ROWS ADJUST
                        <1> R4:                               ; EVEN_ROW
00002C52 6696           <1>       XCHG   SI, AX              ; MOVE POINTER TO (SI) AND RECOVER (AX)
00002C54 81C600800B00   <1>       add    esi, 0B8000h
00002C5A 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>
00002C5D 66BBC002       <1>       MOV    BX, 2C0H
00002C61 66B90203       <1>       MOV    CX, 302H            ; SET PARMS FOR MED RES
00002C65 803D[CA6F0000]06 <1>     CMP    byte [CRT_MODE], 6
00002C6C 7208           <1>       JC     short R5            ; HANDLE IF MED RES
00002C6E 66BB8001       <1>       MOV    BX, 180H
00002C72 66B90307       <1>       MOV    CX, 703H            ; SET PARMS FOR HIGH RES
                        <1>
                        <1> ;-----      DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
                        <1> R5:
00002C76 20D5           <1>       AND    CH, DL              ; ADDRESS OF PEL WITHIN BYTE TO CH
                        <1>
                        <1> ;-----      DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
                        <1>
00002C78 66D3EA         <1>       SHR    DX, CL              ; SHIFT BY CORRECT AMOUNT
00002C7B 6601D6         <1>       ADD    SI, DX              ; INCREMENT THE POINTER
00002C7E 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>
00002C80 28C9           <1>       SUB    CL, CL              ; ZERO INTO STORAGE LOCATION
                        <1> R6:
00002C82 D0C8           <1>       ROR    AL, 1               ; LEFT JUSTIFY VALUE IN AL (FOR WRITE)
00002C84 00E9           <1>       ADD    CL, CH              ; ADD IN THE BIT OFFSET VALUE
00002C86 FECF           <1>       DEC    BH                  ; LOOP CONTROL
00002C88 75F8           <1>       JNZ    short R6            ; ON EXIT, CL HAS COUNT TO RESTORE BITS
00002C8A 88DC           <1>       MOV    AH, BL              ;  GET MASK TO AH
00002C8C D2EC           <1>       SHR    AH, CL              ;  MOVE THE MASK TO CORRECT LOCATION
00002C8E 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>       ;
00002C8F 66BAC803       <1>       mov    dx, 3C8h  ; VGAREG_DAC_WRITE_ADDRESS
00002C93 28C0           <1>       sub    al, al ; 0
00002C95 EE             <1>       out    dx, al ; 0 ; color index, always 0 at the beginning
00002C96 6642           <1>       inc    dx   ; 3C9h ; VGAREG_DAC_DATA
00002C98 B900010000     <1>       mov    ecx, 256   ; always 256*3 values
                        <1>       ;push  esi
00002C9D 88E0           <1>       mov    al, ah
00002C9F B43F           <1>       mov    ah, 3Fh      ; 3Fh except DAC selection number 3
00002CA1 3C02           <1>       cmp    al, 2
00002CA3 7414           <1>       je     short l_dac_p_2
00002CA5 7719           <1>       ja     short l_dac_p_3
00002CA7 20C0           <1>       and    al, al
00002CA9 7507           <1>       jnz    short l_dac_p_1
                        <1> l_dac_p_0:
00002CAB BE[145B0100]   <1>       mov    esi, palette0
00002CB0 EB15           <1>       jmp    short l_dac_p_4
                        <1> l_dac_p_1:
00002CB2 BE[D45B0100]   <1>       mov    esi, palette1
00002CB7 EB0E           <1>       jmp    short l_dac_p_4
                        <1> l_dac_p_2:
00002CB9 BE[945C0100]   <1>       mov    esi, palette2
00002CBE EB07           <1>       jmp    short l_dac_p_4
                        <1> l_dac_p_3:
00002CC0 B4FF           <1>       mov    ah, 0FFh ; dac registers
```

```
5417 00002CC2 BE[545D0100]        <1>       mov   esi, palette3
5418                              <1> l_dac_p_4:
5419 00002CC7 AC                  <1>       lodsb
5420 00002CC8 EE                  <1>       out   dx, al  ; Red
5421 00002CC9 AC                  <1>       lodsb
5422 00002CCA EE                  <1>       out   dx, al ; Green
5423 00002CCB AC                  <1>       lodsb
5424 00002CCC EE                  <1>       out   dx, al ; Blue
5425 00002CCD 20E4                <1>       and   ah, ah
5426 00002CCF 7405                <1>       jz    short l_dac_p_5
5427 00002CD1 FECC                <1>       dec   ah
5428 00002CD3 E2F2                <1>       loop  l_dac_p_4
5429                              <1>       ;pop  esi
5430 00002CD5 C3                  <1>       retn
5431                              <1> l_dac_p_5:
5432                              <1>       ; 29/07/2016
5433 00002CD6 FEC9                <1>       dec   cl
5434 00002CD8 7407                <1>       jz    short l_dac_p_7
5435                              <1>       ;
5436 00002CDA 28C0                <1>       sub   al, al ; 0
5437                              <1> l_dac_p_6:
5438 00002CDC EE                  <1>       out   dx, al ; outb(VGAREG_DAC_DATA,0);
5439 00002CDD EE                  <1>       out   dx, al
5440 00002CDE EE                  <1>       out   dx, al
5441 00002CDF E2FB                <1>       loop  l_dac_p_6
5442                              <1> l_dac_p_7:
5443                              <1>       ;pop  esi
5444 00002CE1 C3                  <1>       retn
5445                              <1>
5446                              <1> gray_scale_summing:
5447                              <1>       ; 03/07/2016 (TRDOS 386 = TRDOS v2.0)
5448                              <1>       ; (set_mode_vga)
5449                              <1>       ; derived from 'Plex86/Bochs VGABios' source code
5450                              <1>       ; vgabios-0.7a (2011)
5451                              <1>       ; by the LGPL VGABios developers Team (2001-2008)
5452                              <1>       ; 'vgabios.c', 'biosfn_perform_gray_scale_summing'
5453                              <1>       ;
5454                              <1>       ; Oracle VirtualBox 5.0.24 VGABios Source Code
5455                              <1>       ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
5456                              <1>       ;
5457                              <1>
5458                              <1>       ; INPUT -> EBX = Start address (color index <= 255)
5459                              <1>       ;         ECX = Count (<= 256)
5460                              <1>       ; OUTPUT -> (E)CX = 0
5461                              <1>       ; (Modifed registers: EAX, ECX, EDX, EBX)
5462                              <1>
5463 00002CE2 66BADA03            <1>       mov   dx, 3DAh ; VGAREG_ACTL_RESET
5464 00002CE6 EC                  <1>       in    al, dx
5465 00002CE7 30C0                <1>       xor   al, al ; 0
5466 00002CE9 66BAC003            <1>       mov   dx, 3C0h ; VGAREG_ACTL_ADDRESS
5467 00002CED EE                  <1>       out   dx, al ; clear bit 5
5468                              <1>                    ; (while loading palette registers)
5469                              <1>       ; set read address and switch to read mode
5470                              <1> g_s_s_1:
5471 00002CEE 66BAC703            <1>       mov   dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
5472 00002CF2 88D8                <1>       mov   al, bl
5473 00002CF4 EE                  <1>       out   dx, al
5474                              <1>       ; get 6-bit wide RGB data values
5475                              <1>       ; intensity = (0.3*Red)+(0.59*Green)+(0.11*Blue)
5476                              <1>       ; i = ( ( 77*r + 151*g + 28*b ) + 0x80 ) >> 8;
5477 00002CF5 66BAC903            <1>       mov   dx, 3C9h ; VGAREG_DAC_DATA
5478 00002CF9 EC                  <1>       in    al, dx ; red
5479 00002CFA B44D                <1>       mov   ah, 77 ; 0.3* Red
5480 00002CFC F6E4                <1>       mul   ah
5481 00002CFE 6650                <1>       push  ax
5482 00002D00 EC                  <1>       in    al, dx ; green
5483 00002D01 B497                <1>       mov   ah, 151  ; 0.59 * Green
5484 00002D03 F6E4                <1>       mul   ah
5485 00002D05 6650                <1>       push  ax
5486 00002D07 EC                  <1>       in    al, dx ; blue
5487 00002D08 B41C                <1>       mov   ah, 28 ; 0.11 * Blue
5488 00002D0A F6E4                <1>       mul   ah
5489 00002D0C 665A                <1>       pop   dx
5490 00002D0E 6601D0              <1>       add   ax, dx
5491 00002D11 665A                <1>       pop   dx
5492 00002D13 6601D0              <1>       add   ax, dx
5493 00002D16 66058000            <1>       add   ax, 80h
5494 00002D1A B03F                <1>       mov   al, 3Fh
5495 00002D1C 38C4                <1>       cmp   ah, al ; if(i>0x3f)i=0x3f
5496 00002D1E 7602                <1>       jna   short g_s_s_2
5497 00002D20 88C4                <1>       mov   ah, al
5498                              <1> g_s_s_2:
5499 00002D22 66BAC803            <1>       mov   dx, 3C8h  ; VGAREG_DAC_WRITE_ADDRESS
5500 00002D26 88D8                <1>       mov   al, bl ; color index
5501 00002D28 EE                  <1>       out   dx, al
5502 00002D29 88E0                <1>       mov   al, ah ; intensity
5503 00002D2B 6642                <1>       inc   dx ; 3C9h ; VGAREG_DAC_DATA
5504 00002D2D EE                  <1>       out   dx, al ; R (R=G=B)
5505 00002D2E 88E0                <1>       mov   al, ah ; intensity
5506 00002D30 EE                  <1>       out   dx, al ; G (R=G=B)
5507 00002D31 88E0                <1>       mov   al, ah ; intensity
5508 00002D33 EE                  <1>       out   dx, al ; B (R=G=B)
5509 00002D34 6649                <1>       dec   cx
5510 00002D36 7404                <1>       jz    short g_s_s_3
5511 00002D38 FEC3                <1>       inc   bl    ; next color index value
5512 00002D3A EBB2                <1>       jmp   short g_s_s_1
5513                              <1> g_s_s_3:
5514 00002D3C 66BADA03            <1>       mov   dx, 3DAh ; VGAREG_ACTL_RESET
5515 00002D40 EC                  <1>       in    al, dx
5516 00002D41 B020                <1>       mov   al, 20h
5517 00002D43 66BAC003            <1>       mov   dx, 3C0h ; VGAREG_ACTL_ADDRESS
5518 00002D47 EE                  <1>       out   dx, al ; 20h -> set bit 5
5519                              <1>                      ; (after loading palette regs)
5520 00002D48 C3                  <1>       retn
5521                              <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>
00002D49 8A25[CA6F0000]         <1>       mov   ah, [CRT_MODE]
00002D4F 668B15[DE890100]       <1>       mov   dx, [CURSOR_POSN] ; cursor pos for page 0
                                <1>
00002D56 BE[E66F0000]           <1>       mov   esi, vga_modes
00002D5B 89F7                   <1>       mov   edi, esi
00002D5D 83C710                 <1>       add   edi, vga_mode_count
                                <1> vga_wca_0:
00002D60 AC                     <1>       lodsb
00002D61 38E0                   <1>       cmp   al, ah ; [CRT_MODE]
00002D63 7405                   <1>       je    short vga_wca_2
00002D65 39FE                   <1>       cmp   esi, edi
00002D67 72F7                   <1>       jb    short vga_wca_0
                                <1> vga_wca_1:
00002D69 C3                     <1>       retn  ; nothing to do
                                <1> vga_wca_2:
00002D6A 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
00002D6D 803E04                 <1>       cmp   byte [esi], PLANAR4
00002D70 741D                   <1>       je    short vga_wca_planar
00002D72 803E03                 <1>       cmp   byte [esi], PLANAR1
00002D75 7418                   <1>       je    short vga_wca_planar
                                <1> vga_wca_linear8:
                                <1>       ; while((count-->0) && (xcurs<nbcols))
                                <1>       ; CX = count
00002D77 6621C9                 <1>       and   cx, cx
00002D7A 74ED                   <1>       jz    short vga_wca_1
00002D7C 3A15[CC6F0000]         <1>       cmp   dl, [CRT_COLS]
00002D82 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
00002D84 E81E000000             <1>       call  write_gfx_char_lin
00002D89 6649                   <1>       dec   cx ; count
00002D8B FEC2                   <1>       inc   dl ; xcurs
00002D8D EBE8                   <1>       jmp   short vga_wca_linear8
                                <1> vga_wca_planar:
                                <1>       ; while((count-->0) && (xcurs<nbcols))
                                <1>       ; CX = count
00002D8F 6621C9                 <1>       and   cx, cx
00002D92 74D5                   <1>       jz    short vga_wca_1
00002D94 3A15[CC6F0000]         <1>       cmp   dl, [CRT_COLS]
00002D9A 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
00002D9C E8A9000000             <1>       call  write_gfx_char_pl4
00002DA1 6649                   <1>       dec   cx ; count
00002DA3 FEC2                   <1>       inc   dl ; xcurs
00002DA5 EBE8                   <1>       jmp   short vga_wca_planar
                                <1>
                                <1> write_gfx_char_lin:
                                <1>       ; 08/01/2021
                                <1>       ; 05/01/2021 (TRDOS 386 v2.0.3)
                                <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>
00002DA7 51                     <1>       push  ecx
00002DA8 53                     <1>       push  ebx
00002DA9 52                     <1>       push  edx
00002DAA 50                     <1>       push  eax
                                <1>       ; addr=xcurs*8+ycurs*nbcols*64;
                                <1>       ; 08/08/2016
00002DAB 0FB6F0                 <1>       movzx esi, al ; car
                                <1>       ; 08/01/2021
                                <1>       ;movzx eax, dh ; ycurs
                                <1>       ;mov   ah, [CRT_COLS] ; nbcols
                                <1>       ;mul   ah
00002DAE A0[CC6F0000]           <1>       mov   al, [CRT_COLS]
00002DB3 F6E6                   <1>       mul   dh
                                <1>       ;shl   ax, 6 ; * 64
00002DB5 66C1E003               <1>       shl   ax, 3 ; 8 * ycurs * [CRT_COLS]
```

```
                                <1>       ;sub   dh, dh
                                <1>       ;shl   dx, 3 ; xcurs * 8
                                <1>       ;movzx edi, dx
00002DB9 BF00000A00             <1>       mov    edi, 0A0000h
00002DBE 30F6                   <1>       xor    dh, dh
00002DC0 6689D7                 <1>       mov    di, dx
                                <1>       ;movzx edi, dl
00002DC3 66C1E703               <1>       shl    di, 3 ; xcurs * 8
                                <1>       ;xor   dh, dh
00002DC7 8A15[CE6F0000]         <1>       mov    dl, [CHAR_HEIGHT]
00002DCD 66F7E2                 <1>       mul    dx
                                <1>       ; eax = ycurs*nbcols*8*[CHAR_HEIGHT]
                                <1>       ;add   edi, eax ; addr
                                <1>       ;add   edi, 0A0000h
00002DD0 6601C7                 <1>       add    di, ax
                                <1>       ;shl   si, 3 ; car * 8
00002DD3 30E4                   <1>       xor    ah, ah
00002DD5 A0[CE6F0000]           <1>       mov    al, [CHAR_HEIGHT]
00002DDA 66F7E6                 <1>       mul    si
00002DDD 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
00002DE0 A1[6A960100]           <1>       mov    eax, [VGA_INT43H]
00002DE5 09C0                   <1>       or     eax, eax ; 0 ?
00002DE7 743F                   <1>       jz     short wfxl_7 ; yes, default font
                                <1>       ;cmp   eax, vgafont16
                                <1>         ;je  short wgfxl_0
                                <1>       ;cmp   eax, vgafont14
                                <1>       ;je    short wgfxl_0
                                <1>       ;cmp   eax, vgafont8
                                <1>       ;je    short wgfxl_0
                                <1>       ;; 05/01/2021 (TRDOS 386 v2.0.3)
                                <1>       ;; user font
                                <1>       ;mov   eax, VGAFONTUSR ; 8x16 or 8x8 or 8x14 font
                                <1>       ;                      ; (256 characters)
                                <1> wgfxl_0:
00002DE9 01C6                   <1>       add    esi, eax
                                <1> wgfxl_1:
00002DEB 28FF                   <1>       sub    bh, bh ; i = 0
                                <1> wgfxl_2:
                                <1>       ; for(i=0;i<8;i++)
00002DED 57                     <1>       push   edi ; addr
00002DEE 0FB605[CC6F0000]       <1>       movzx  eax, byte [CRT_COLS] ; nbcols
00002DF5 F6E7                   <1>       mul    bh ; nbcols*i
00002DF7 66C1E003               <1>       shl    ax, 3 ; i*nbcols*8
                                <1>       ; dest=addr+i*nbcols*8;
00002DFB 01C7                   <1>       add    edi, eax ; dest + j ; j = 0
00002DFD B180                   <1>       mov    cl, 80h ; mask = 0x80;
                                <1>       ; esi = fdata + src + i
                                <1>       ; for(j=0;j<8;j++)
00002DFF 29D2                   <1>       sub    edx, edx ; j = 0
                                <1> wgfxl_3:
00002E01 8A06                   <1>       mov    al, [esi] ; al = fdata[src+i]
00002E03 20C8                   <1>       and    al, cl ; if (fdata[src+i] & mask)
00002E05 7402                   <1>       jz     short wgfxl_4  ; data = 0, zf = 1
00002E07 88D8                   <1>       mov    al, bl ; data = attr;
                                <1> wgfxl_4:
                                <1>       ; write_byte(0xa000,dest+j,data);
00002E09 AA                     <1>       stosb  ; dest + j (+ 0A0000h)
                                <1>       ;inc   dl ; j++
                                <1>       ;cmp   dl, 8
00002E0A 80FA07                 <1>       cmp    dl, 7
00002E0D 720E                   <1>       jb     short wgfxl_5
00002E0F 5F                     <1>       pop    edi
                                <1>       ; 08/08/2016
                                <1>       ;cmp   bh, 7
                                <1>       ;jnb   short wgfxl_6
00002E10 FEC7                   <1>       inc    bh ; i++
00002E12 3A3D[CE6F0000]         <1>       cmp    bh, [CHAR_HEIGHT]
00002E18 7309                   <1>       jnb    short wgfxl_6
00002E1A 46                     <1>       inc    esi
00002E1B EBD0                   <1>       jmp    short wgfxl_2
                                <1> wgfxl_5:
00002E1D D0E9                   <1>       shr    cl, 1 ; mask >>= 1;
00002E1F FEC2                   <1>       inc    dl ; j++
00002E21 EBDE                   <1>         jmp     short wgfxl_3
                                <1> wgfxl_6:
00002E23 58                     <1>       pop    eax
00002E24 5A                     <1>       pop    edx
00002E25 5B                     <1>       pop    ebx
00002E26 59                     <1>       pop    ecx
00002E27 C3                     <1>       retn
                                <1> wfxl_7:
                                <1>       ; 08/01/2021
                                <1>       ; 05/01/2021
                                <1>       ; Default font (8x8 or 8x14 or 8x16)
00002E28 A0[CE6F0000]           <1>       mov    al, [CHAR_HEIGHT]
00002E2D 3C08                   <1>       cmp    al, 8
00002E2F 7507                   <1>       jne    short wfxl_8
00002E31 B8[54600100]           <1>       mov    eax, vgafont8
00002E36 EBB1                   <1>       jmp    short wgfxl_0
                                <1> wfxl_8:
00002E38 3C0E                   <1>       cmp    al, 14
00002E3A 7507                   <1>       jne    short wfxl_9
00002E3C B8[54680100]           <1>       mov    eax, vgafont14
00002E41 EBA6                   <1>       jmp    short wgfxl_0
                                <1> wfxl_9:
00002E43 B8[54760100]           <1>       mov    eax, vgafont16
00002E48 EB9F                   <1>       jmp    short wgfxl_0
                                <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>
00002E4A 51             <1>       push   ecx
00002E4B 53             <1>       push   ebx
00002E4C 52             <1>       push   edx
00002E4D 50             <1>       push   eax
                        <1> wgfxpl_f0:
                        <1>       ; switch(cheight)
00002E4E 8A25[CE6F0000] <1>       mov    ah, [CHAR_HEIGHT]
00002E54 80FC10         <1>       cmp    ah, 16 ; case 16:
00002E57 7507           <1>       jne    short wgfxpl_f1
                        <1>       ; fdata = &vgafont16;
00002E59 BE[54760100]   <1>       mov    esi, vgafont16
00002E5E EB13           <1>       jmp    short wgfxpl_f3
                        <1> wgfxpl_f1:
00002E60 80FC0E         <1>       cmp    ah, 14 ; case 14:
00002E63 7507           <1>       jne    short wgfxpl_f2
00002E65 BE[54680100]   <1>       mov    esi, vgafont14
00002E6A EB07           <1>       jmp    short wgfxpl_f3
                        <1> wgfxpl_f2:
                        <1>       ; default:
                        <1>       ;  fdata = &vgafont8;
00002E6C BE[54600100]   <1>       mov    esi, vgafont8
00002E71 B408           <1>       mov    ah, 8
                        <1> wgfxpl_f3:
                        <1>       ; al = car
00002E73 F6E4           <1>       mul    ah ; ah = cheight
00002E75 25FFFF0000     <1>       and    eax, 0FFFFh ; car * cheight
                        <1>       ; src = car * cheight;
00002E7A 01C6           <1>       add    esi, eax ; esi = fdata[src+i]
                        <1>       ; addr=xcurs*8+ycurs*nbcols*64;
00002E7C 88F0           <1>       mov    al, dh ; ycurs
00002E7E 8A25[CC6F0000] <1>       mov    ah, [CRT_COLS] ; nbcols
00002E84 F6E4           <1>       mul    ah
                        <1>       ; 08/08/2016
                        <1>       ;shl   ax, 6 ; * 64
00002E86 66C1E003       <1>       shl    ax, 3 ; * 8
                        <1>       ;sub   dh, dh ; 0
                        <1>       ;shl   dx, 3 ; xcurs * 8
                        <1>       ;movzx edi, dx
00002E8A 0FB6FA         <1>       movzx  edi, dl
00002E8D 66C1E703       <1>       shl    di, 3 ; xcurs * 8
00002E91 30F6           <1>       xor    dh, dh
00002E93 8A15[CE6F0000] <1>       mov    dl, [CHAR_HEIGHT]
00002E99 66F7E2         <1>       mul    dx
                        <1>       ; eax = ycurs*nbcols*8*[CHAR_HEIGHT]
00002E9C 01C7           <1>       add    edi, eax ; addr
00002E9E 81C700000A00   <1>       add    edi, 0A0000h
                        <1>       ;
                        <1>       ; outw(VGAREG_SEQU_ADDRESS, 0x0f02);
                        <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0205);
00002EA4 66BAC403       <1>       mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
00002EA8 66B8020F       <1>       mov    ax, 0F02h
00002EAC 66EF           <1>       out    dx, ax
00002EAE 66BACE03       <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
00002EB2 66B80502       <1>       mov    ax, 0205h
00002EB6 66EF           <1>       out    dx, ax
                        <1>       ;
00002EB8 66BACE03       <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
00002EBC F6C380         <1>       test   bl, 80h ; if(attr&0x80)
00002EBF 7406           <1>       jz     short wgfxpl_f4 ; else
                        <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x1803);
00002EC1 66B80318       <1>       mov    ax, 1803h
00002EC5 EB04           <1>       jmp    short wgfxpl_f5
                        <1> wgfxpl_f4:
                        <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0003);
00002EC7 66B80300       <1>       mov    ax, 0003h
                        <1> wgfxpl_f5:
00002ECB 66EF           <1>       out    dx, ax
                        <1>       ;
00002ECD 28FF           <1>       sub    bh, bh ; i = 0
                        <1> wgfxpl_0:
                        <1>       ; for(i=0;i<cheight;i++)
00002ECF 57             <1>       push   edi ; addr
00002ED0 0FB605[CC6F0000] <1>     movzx  eax, byte [CRT_COLS] ; nbcols
00002ED7 F6E7           <1>       mul    bh ; nbcols*i
                        <1>       ; dest=addr+i*nbcols
00002ED9 01C7           <1>       add    edi, eax ; dest
00002EDB B580           <1>       mov    ch, 80h ; mask = 0x80;
                        <1>       ; for(j=0;j<8;j++)
00002EDD 28C9           <1>       sub    cl, cl ; j = 0
                        <1> wgfxpl_1:
00002EDF 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
00002EE1 88EC           <1>       mov    ah, ch
00002EE3 B008           <1>       mov    al, 8
00002EE5 66EF           <1>       out    dx, ax
00002EE7 8A07           <1>       mov    al, [edi] ; ? (io delay?)
                        <1>       ;
00002EE9 28C0           <1>       sub    al, al ; attr = 0
```

```
                                <1>       ; if (fdata[src+i] & mask)
00002EEB 842E                   <1>       test  byte [esi], ch
00002EED 7404                   <1>       jz    short wgfxpl_2  ; zf = 1
                                <1>       ; write_byte(0xa000,dest,attr&0x0f);
00002EEF 88D8                   <1>       mov   al, bl ; attr;
00002EF1 240F                   <1>       and   al, 0Fh      ; attr&0x0f
                                <1> wgfxpl_2:
                                <1>       ; write_byte(0xa000,dest,0x00);
00002EF3 8807                   <1>       mov   [edi], al ; dest (+ 0A0000h)
00002EF5 FEC1                   <1>       inc   cl ; j++
00002EF7 80F908                 <1>       cmp   cl, 8
00002EFA 72E3                   <1>       jb    short wgfxpl_1
00002EFC 5F                     <1>       pop   edi
                                <1>       ; 08/08/2016
                                <1>       ;cmp  bh, 7
                                <1>       ;jnb  short wgfxpl_3
00002EFD FEC7                   <1>       inc   bh ; i++
00002EFF 3A3D[CE6F0000]         <1>       cmp   bh, [CHAR_HEIGHT]
00002F05 7303                   <1>       jnb   short wgfxpl_3
00002F07 46                     <1>       inc   esi
00002F08 EBC5                   <1>       jmp   short wgfxpl_0
                                <1> wgfxpl_3:
                                <1>       ;mov  dx, 3CEh ; VGAREG_GRDC_ADDRESS
00002F0A 66B808FF               <1>       mov   ax, 0FF08h
00002F0E 66EF                   <1>       out   dx, ax
00002F10 66B80500               <1>       mov   ax, 0005h
00002F14 66EF                   <1>       out   dx, ax
00002F16 66B80300               <1>       mov   ax, 0003h
00002F1A 66EF                   <1>       out   dx, ax
                                <1>       ;
00002F1C 58                     <1>       pop   eax
00002F1D 5A                     <1>       pop   edx
00002F1E 5B                     <1>       pop   ebx
00002F1F 59                     <1>       pop   ecx
00002F20 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>
00002F21 88C3                   <1>       mov   bl, al ; pixel value
                                <1>       ;mov  ah, [CRT_MODE]
00002F23 BE[E66F0000]           <1>       mov   esi, vga_modes
00002F28 89F7                   <1>       mov   edi, esi
00002F2A 83C710                 <1>       add   edi, vga_mode_count
                                <1> vga_wp_0:
00002F2D AC                     <1>       lodsb
00002F2E 38E0                   <1>       cmp   al, ah ; [CRT_MODE]
00002F30 7405                   <1>       je    short vga_wp_1
00002F32 39FE                   <1>       cmp   esi, edi
00002F34 72F7                   <1>       jb    short vga_wp_0
00002F36 C3                     <1>       retn  ; nothing to do
                                <1> vga_wp_1:
00002F37 83C64F                 <1>       add   esi, vga_memmodel - (vga_modes + 1)
                                <1>       ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
00002F3A BF00000A00             <1>       mov   edi, 0A0000h
                                <1>       ;
00002F3F 803E04                 <1>       cmp   byte [esi], PLANAR4
00002F42 741D                   <1>       je    short vga_wp_planar
00002F44 803E03                 <1>       cmp   byte [esi], PLANAR1
00002F47 7418                   <1>       je    short vga_wp_planar
                                <1> vga_wp_linear8:
                                <1>       ; addr=CX+DX*(read_word(BIOSMEM_SEG,BIOSMEM_NB_COLS)*8);
00002F49 0FB605[CC6F0000]       <1>       movzx  eax, byte [CRT_COLS] ; = [VGA_COLS] ; nbcols
00002F50 66C1E003               <1>       shl   ax, 3 ; * 8
00002F54 66F7E2                 <1>       mul   dx
00002F57 50                     <1>       push  eax
                                <1>       ;mov  edi, 0A0000h
00002F58 6601CF                 <1>       add   di, cx
00002F5B 58                     <1>       pop   eax
00002F5C 01C7                   <1>       add   edi, eax ; addr
                                <1>       ; write_byte(0xa000,addr,AL);
00002F5E 881F                   <1>       mov   [edi], bl
00002F60 C3                     <1>       retn
                                <1> vga_wp_planar:
                                <1>       ; addr = CX/8+DX*read_word(BIOSMEM_SEG,BIOSMEM_NB_COLS);
00002F61 0FB7C1                 <1>       movzx  eax, cx
00002F64 66C1E803               <1>       shr   ax, 3 ; CX/8
00002F68 50                     <1>       push  eax
00002F69 28E4                   <1>       sub   ah, ah ; 0
00002F6B A0[CC6F0000]           <1>       mov   al, [CRT_COLS] ; = [VGA_COLS] ; nbcols
00002F70 66F7E2                 <1>       mul   dx
                                <1>       ;mov  edi, 0A0000h
00002F73 6601C7                 <1>         add     di, ax
00002F76 58                     <1>       pop   eax
00002F77 01C7                   <1>       add   edi, eax ; addr
00002F79 80E107                 <1>       and   cl, 7
00002F7C B580                   <1>       mov   ch, 80h ; mask
00002F7E D2ED                   <1>       shr   ch, cl       ; mask = 0x80 >> (CX & 0x07);
                                <1>
                                <1>       ; outw(VGAREG_GRDC_ADDRESS, (mask << 8) | 0x08);
00002F80 66BACE03               <1>       mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
00002F84 88EC                   <1>       mov   ah, ch
```

```
5942 00002F86 B008                <1>       mov    al, 8
5943 00002F88 66EF                <1>       out    dx, ax
5944                              <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0205);
5945 00002F8A 66B80502            <1>       mov    ax, 0205h
5946 00002F8E 66EF                <1>       out    dx, ax
5947                              <1>       ; data = read_byte(0xa000,addr);
5948 00002F90 8A07                <1>       mov    al, [edi] ; (delay?)
5949                              <1>       ; if (AL & 0x80)
5950                              <1>       ; {
5951                              <1>       ;  outw(VGAREG_GRDC_ADDRESS, 0x1803);
5952                              <1>       ; }
5953 00002F92 F6C380              <1>       test   bl, 80h
5954 00002F95 7406                <1>       jz     short vga_wp_2
5955 00002F97 66B80318            <1>       mov    ax, 1803h
5956 00002F9B 66EF                <1>       out    dx, ax
5957                              <1> vga_wp_2:
5958                              <1>       ; write_byte(0xa000,addr,AL);
5959 00002F9D 881F                <1>       mov    [edi], bl
5960                              <1>       ;
5961                              <1>       ;mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
5962 00002F9F 66B808FF            <1>       mov    ax, 0FF08h
5963 00002FA3 66EF                <1>       out    dx, ax
5964 00002FA5 66B80500            <1>       mov    ax, 0005h
5965 00002FA9 66EF                <1>       out    dx, ax
5966 00002FAB 66B80300            <1>       mov    ax, 0003h
5967 00002FAF 66EF                <1>       out    dx, ax
5968                              <1>       ;
5969 00002FB1 C3                  <1>       retn
5970                              <1>
5971                              <1> vga_read_pixel:
5972                              <1>       ; 09/07/2016 (TRDOS 386 = TRDOS v2.0)
5973                              <1>       ;
5974                              <1>       ; derived from 'Plex86/Bochs VGABios' source code
5975                              <1>       ; vgabios-0.7a (2011)
5976                              <1>       ; by the LGPL VGABios developers Team (2001-2008)
5977                              <1>       ; 'vgabios.c', 'biosfn_read_pixel'
5978                              <1>
5979                              <1>       ; INPUT ->
5980                              <1>       ;      DX = row (0-239)
5981                              <1>       ;      CX = column (0-799)
5982                              <1>       ;      (AH = [CRT_MODE])
5983                              <1>       ; OUTPUT ->
5984                              <1>       ;      AL = pixel value
5985                              <1>
5986                              <1>       ;mov   ah, [CRT_MODE]
5987 00002FB2 BE[E66F0000]        <1>       mov    esi, vga_modes
5988 00002FB7 89F7                <1>       mov    edi, esi
5989 00002FB9 83C710              <1>       add    edi, vga_mode_count
5990                              <1> vga_rp_0:
5991 00002FBC AC                  <1>       lodsb
5992 00002FBD 38E0                <1>       cmp    al, ah ; [CRT_MODE]
5993 00002FBF 7405                <1>       je     short vga_rp_1
5994 00002FC1 39FE                <1>       cmp    esi, edi
5995 00002FC3 72F7                <1>       jb     short vga_rp_0
5996 00002FC5 C3                  <1>       retn   ; nothing to do
5997                              <1> vga_rp_1:
5998 00002FC6 83C64F              <1>       add    esi, vga_memmodel - (vga_modes + 1)
5999                              <1>       ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
6000 00002FC9 BF00000A00          <1>       mov    edi, 0A0000h
6001                              <1>       ;
6002 00002FCE 803E04              <1>       cmp    byte [esi], PLANAR4
6003 00002FD1 741D                <1>       je     short vga_rp_planar
6004 00002FD3 803E03              <1>       cmp    byte [esi], PLANAR1
6005 00002FD6 7418                <1>       je     short vga_rp_planar
6006                              <1> vga_rp_linear8:
6007                              <1>       ; addr=CX+DX*(read_word(BIOSMEM_SEG,BIOSMEM_NB_COLS)*8);
6008 00002FD8 0FB605[CC6F0000]    <1>       movzx  eax, byte [CRT_COLS] ; = [VGA_COLS] ; nbcols
6009 00002FDF 66C1E003            <1>       shl    ax, 3 ; * 8
6010 00002FE3 66F7E2              <1>       mul    dx
6011 00002FE6 50                  <1>       push   eax
6012                              <1>       ;mov   edi, 0A0000h
6013 00002FE7 6601CF              <1>       add    di, cx
6014 00002FEA 58                  <1>       pop    eax
6015 00002FEB 01C7                <1>       add    edi, eax ; addr
6016                              <1>       ; attr=read_byte(0xa000,addr);
6017 00002FED 8A07                <1>       mov    al, [edi] ; pixel value
6018 00002FEF C3                  <1>       retn
6019                              <1> vga_rp_planar:
6020                              <1>       ; addr = CX/8+DX*read_word(BIOSMEM_SEG,BIOSMEM_NB_COLS);
6021 00002FF0 0FB7C1              <1>       movzx  eax, cx
6022 00002FF3 66C1E803            <1>       shr    ax, 3 ; CX/8
6023 00002FF7 50                  <1>       push   eax
6024 00002FF8 28E4                <1>       sub    ah, ah ; 0
6025 00002FFA A0[CC6F0000]        <1>       mov    al, [CRT_COLS] ; = [VGA_COLS] ; nbcols
6026 00002FFF 66F7E2              <1>       mul    dx
6027                              <1>       ;mov   edi, 0A0000h
6028 00003002 6601C7              <1>         add     di, ax
6029 00003005 58                  <1>       pop    eax
6030 00003006 01C7                <1>       add    edi, eax ; addr
6031 00003008 80E107              <1>       and    cl, 7
6032 0000300B B580                <1>       mov    ch, 80h ; mask
6033 0000300D D2ED                <1>       shr    ch, cl       ; mask = 0x80 >> (CX & 0x07);
6034                              <1>       ; attr = 0x00;
6035 0000300F 30DB                <1>       xor    bl, bl ; attr = bl = 0,
6036 00003011 30C9                <1>       xor    cl, cl ; i = cl = 0
6037                              <1>       ; for(i=0;i<4;i++)
6038                              <1>             ; {
6039                              <1>             ;  outw(VGAREG_GRDC_ADDRESS, (i << 8) | 0x04);
6040                              <1>             ;  data = read_byte(0xa000,addr) & mask;
6041                              <1>             ;  if (data > 0) attr |= (0x01 << i);
6042                              <1>             ; }
6043                              <1> vga_rp_2:
6044 00003013 88CC                <1>       mov    ah, cl ; i << 8
6045 00003015 B004                <1>       mov    al, 4  ; | 0x04
6046 00003017 66BACE03            <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
```

```
6047 0000301B 66EF              <1>       out    dx, ax
6048                            <1>       ; data = read_byte(0xa000,addr) & mask;
6049 0000301D 8A07              <1>       mov    al, [edi]
6050 0000301F 20E8              <1>       and    al, ch ; & mask
6051                            <1>       ; if (data > 0) attr |= (0x01 << i);
6052 00003021 08C0              <1>       or     al, al
6053 00003023 7408              <1>       jz     short vga_rp_3 ; al = 0
6054 00003025 B701              <1>       mov    bh, 1
6055 00003027 D2E7              <1>       shl    bh, cl ; (0x01 << i)
6056 00003029 08FB              <1>       or     bl, bh ; attr |= (0x01 << i)
6057 0000302B 88D8              <1>       mov    al, bl ; pixel value
6058                            <1> vga_rp_3:
6059 0000302D C3                <1>       retn
6060                            <1>
6061                            <1> vga_beeper:
6062                            <1>       ; 04/08/2016  (TRDOS 386 = TRDOS v2.0)
6063 0000302E FB                <1>       sti
6064                            <1>       ;mov   bh, [ACTIVE_PAGE]
6065 0000302F E9BEF3FFFF        <1>         jmp     beeper_gfx
6066                            <1>
6067                            <1> vga_write_teletype:
6068                            <1>       ; 09/12/2017
6069                            <1>       ; 06/08/2016
6070                            <1>       ; 04/08/2016
6071                            <1>       ; 01/08/2016
6072                            <1>       ; 31/07/2016
6073                            <1>       ; 09/07/2016 (TRDOS 386 = TRDOS v2.0)
6074                            <1>       ;
6075                            <1>       ; derived from 'Plex86/Bochs VGABios' source code
6076                            <1>       ; vgabios-0.7a (2011)
6077                            <1>       ; by the LGPL VGABios developers Team (2001-2008)
6078                            <1>       ; 'vgabios.c', 'biosfn_write_teletype'
6079                            <1>       ; 'biosfn_write_char_only'
6080                            <1>
6081                            <1>       ; INPUT ->
6082                            <1>       ; [CRT_MODE] = current video mode (>7)
6083                            <1>       ; AL = Character to write
6084                            <1>       ; BL = Color of character
6085                            <1>       ; OUTPUT ->
6086                            <1>       ; Regen buffer updated
6087                            <1>
6088                            <1>       ; biosfn_write_teletype (car, page, attr, flag)
6089                            <1>       ; car = character (AL)
6090                            <1>       ; page = 0
6091                            <1>       ; attr = color (BL)
6092                            <1>       ; 'flag' not used
6093                            <1>
6094 00003034 8A25[CA6F0000]    <1>       mov    ah, [CRT_MODE]
6095 0000303A 88C7              <1>       mov    bh, al ; character
6096 0000303C 668B15[DE890100]  <1>       mov    dx, [CURSOR_POSN] ; cursor pos for page 0
6097                            <1>
6098 00003043 BE[EE6F0000]      <1>       mov    esi, vga_g_modes
6099 00003048 89F7              <1>       mov    edi, esi
6100 0000304A 83C708            <1>       add    edi, vga_g_mode_count
6101                            <1> vga_wtty_0:
6102 0000304D AC                <1>       lodsb
6103 0000304E 38E0              <1>       cmp    al, ah ; [CRT_MODE]
6104 00003050 7405              <1>       je     short vga_wtty_2
6105 00003052 39FE              <1>       cmp    esi, edi
6106 00003054 72F7              <1>       jb     short vga_wtty_0
6107                            <1> vga_wtty_1:
6108 00003056 C3                <1>       retn   ; nothing to do
6109                            <1> vga_wtty_2:
6110 00003057 80FF07            <1>       cmp    bh, 07h ; bell (beep)
6111 0000305A 74D2              <1>       je     short vga_beeper  ; u11
6112 0000305C 80FF08            <1>       cmp    bh, 08h ; backspace
6113 0000305F 7508              <1>       jne    short vga_wtty_3
6114                            <1>       ; if(xcurs>0)xcurs--;
6115 00003061 08D2              <1>       or     dl, dl ; xcurs (column)
6116 00003063 74F1              <1>       jz     short vga_wtty_1
6117 00003065 FECA              <1>       dec    dl ; xcurs--;
6118 00003067 EB59              <1>         jmp     short vga_wtty_12
6119                            <1> vga_wtty_3:
6120 00003069 80FF0D            <1>       cmp    bh, 0Dh ; carriage return (\r)
6121 0000306C 7504              <1>       jne    short vga_wtty_4
6122                            <1>       ; xcurs=0;
6123 0000306E 28D2              <1>       sub    dl, dl ; 0
6124 00003070 EB50              <1>         jmp     short vga_wtty_12
6125                            <1> vga_wtty_4:
6126 00003072 80FF0A            <1>       cmp    bh, 0Ah ; new line (\n)
6127 00003075 7504              <1>       jne    short vga_wtty_5
6128                            <1>       ; ycurs++;
6129 00003077 FEC6              <1>       inc    dh ; next row
6130 00003079 EB62              <1>         jmp     short vga_wtty_11
6131                            <1> vga_wtty_5:
6132 0000307B 80FF09            <1>       cmp    bh, 09h ; tab stop
6133 0000307E 7527              <1>       jne    short vga_wtty_8
6134 00003080 88D0              <1>       mov    al, dl
6135                            <1>       ;cbw
6136 00003082 30E4              <1>       xor    ah, ah ; 09/12/2017
6137 00003084 B108              <1>       mov    cl, 8
6138 00003086 F6F1              <1>       div    cl
6139 00003088 28E1              <1>       sub    cl, ah
6140                            <1>       ;
6141 0000308A B720              <1>       mov    bh, 20h ; space
6142                            <1> vga_wtty_6: ; tab stop loop
6143 0000308C 6651              <1>       push   cx
6144 0000308E 6653              <1>       push   bx
6145 00003090 E812000000        <1>       call   vga_wtty_8
6146 00003095 665B              <1>       pop    bx  ; bh = character, bl = color
6147 00003097 6659              <1>       pop    cx
6148 00003099 FEC9              <1>       dec    cl
6149 0000309B 7409              <1>       jz     short vga_wtty_7
6150 0000309D 668B15[DE890100]  <1>       mov    dx, [CURSOR_POSN] ; new cursor position (pg 0)
6151 000030A4 EBE6              <1>       jmp    short vga_wtty_6
```

```
6152                                <1> vga_wtty_7:
6153 000030A6 C3                    <1>         retn
6154                                <1>         ;
6155                                <1> vga_wtty_8:
6156 000030A7 83C64F                <1>         add    esi, vga_g_memmodel - (vga_g_modes + 1)
6157                                <1>         ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
6158 000030AA BF00000A00            <1>         mov    edi, 0A0000h
6159                                <1>         ;
6160 000030AF 88F8                  <1>         mov    al, bh ; character
6161                                <1>         ;
6162 000030B1 803E04                <1>         cmp    byte [esi], PLANAR4
6163 000030B4 7414                  <1>         je     short vga_wtty_planar
6164 000030B6 803E03                <1>         cmp    byte [esi], PLANAR1
6165 000030B9 740F                  <1>         je     short vga_wtty_planar
6166                                <1> vga_wtty_linear8:
6167                                <1>         ; write_gfx_char_lin(car,attr,xcurs,ycurs,nbcols);
6168                                <1>         ; AL = car, BL = attr (color), DL = xcurs, DH = ycurs,
6169                                <1>         ; [CRT_COLS] = nbcols
6170 000030BB E8E7FCFFFF            <1>         call   write_gfx_char_lin
6171 000030C0 EB0D                  <1>         jmp    short vga_wtty_9
6172                                <1>
6173                                <1> vga_wtty_12:
6174                                <1>         ; 09/07/2016
6175                                <1>         ; set cursor position
6176                                <1>         ; NOTE: Hardware cursor position will not be set
6177                                <1>         ;   in any VGA modes (>7)
6178                                <1>         ;   But, cursor position will be saved into
6179                                <1>         ;   [CURSOR_POSN].
6180                                <1>         ;   TRDOS 386 (TRDOS v2.0) uses only one page
6181                                <1>         ;   (page 0) for all graphics modes.
6182                                <1>
6183 000030C2 668915[DE890100]      <1>         mov    [CURSOR_POSN], dx ; save cursor pos for pg 0
6184                                <1>         ; 04/08/2016
6185                                <1>         ;mov   bh, [ACTIVE_PAGE] ; = 0
6186                                <1>         ;call  _set_cpos
6187 000030C9 C3                    <1>         retn
6188                                <1>
6189                                <1> vga_wtty_planar:
6190                                <1>         ; write_gfx_char_pl4(car,attr,xcurs,ycurs,nbcols,cheight);
6191                                <1>         ; AL = car, BL = attr (color), DL = xcurs, DH = ycurs,
6192                                <1>         ; [CRT_COLS]= nbcols, [CHAR_HEIGHT] = cheight
6193 000030CA E87BFDFFFF            <1>         call   write_gfx_char_pl4
6194                                <1> vga_wtty_9:
6195 000030CF FEC2                  <1>         inc    dl ; xcurs++;
6196                                <1> vga_wtty_10:
6197                                <1>         ; Do we need to wrap ?
6198                                <1>         ; if(xcurs==nbcols)
6199 000030D1 3A15[CC6F0000]        <1>         cmp    dl, [CRT_COLS] ; [VGA_COLS]
6200 000030D7 7204                  <1>         jb     short vga_wtty_11 ; no
6201 000030D9 28D2                  <1>         sub    dl, dl ; xcurs=0;
6202 000030DB FEC6                  <1>         inc    dh ;  ycurs++;
6203                                <1> vga_wtty_11:
6204                                <1>         ; Do we need to scroll ?
6205                                <1>         ; if(ycurs==nbrows)
6206 000030DD 3A35[D26F0000]        <1>         cmp    dh, [VGA_ROWS]
6207 000030E3 72DD                  <1>         jb     short vga_wtty_12 ; no
6208                                <1>         ;
6209                                <1>         ; biosfn_scroll (nblines,attr,rul,cul,rlr,clr,page,dir)
6210                                <1>         ; al = nblines = 1, bl = attr (color) = 0
6211                                <1>         ; ch = rul, cl = cul, dh = rlr, dl = clr, page = 0
6212                                <1>         ; dir = SCROLL_UP
6213                                <1>
6214 000030E5 B001                  <1>         mov    al, 1
6215 000030E7 28DB                  <1>         sub    bl, bl ; 0 ; blank/black line (attr=0) will be used
6216 000030E9 6629C9                <1>         sub    cx, cx ; 0,0
6217                                <1>
6218                                <1>         ; 06/08/2016
6219 000030EC 8A35[D26F0000]        <1>         mov    dh, [VGA_ROWS]
6220 000030F2 FECE                  <1>         dec    dh ; nbrows -1
6221                                <1>
6222 000030F4 6652                  <1>         push   dx     ; 04/08/2016
6223 000030F6 8A15[CC6F0000]        <1>         mov    dl, [CRT_COLS]
6224 000030FC FECA                  <1>         dec    dl ; nbcols -1
6225                                <1>
6226 000030FE 8A25[CA6F0000]        <1>         mov    ah, [CRT_MODE]
6227                                <1>
6228                                <1>         ; biosfn_scroll(0x01,0x00,0,0,nbrows-1,nbcols-1,page,SCROLL_UP);
6229 00003104 E8FBF4FFFF            <1>         call   vga_graphics_up
6230                                <1>         ; 04/08/2016
6231 00003109 665A                  <1>         pop    dx
6232                                <1>         ;dec   dh ; ycurs-=1
6233 0000310B EBB5                  <1>         jmp    short vga_wtty_12
6234                                <1>
6235                                <1> font_setup:
6236                                <1>         ; 09/01/2021 (TRDOS 386 v2.0.3)
6237                                <1>         ; 09/07/2016
6238                                <1>         ; character generator (font loading) functions
6239                                <1>         ;
6240                                <1>         ; derived from 'Plex86/Bochs VGABios' source code
6241                                <1>         ; vgabios-0.7a (2011)
6242                                <1>         ; by the LGPL VGABios developers Team (2001-2008)
6243                                <1>         ; 'vgabios.c', 'int10_func'
6244                                <1>
6245                                <1>         ; AX = 1100H ; Load User-Defined Font (EGA/VGA)
6246                                <1>         ;
6247                                <1>           ; BH    height of each character (bytes per character definition)
6248                                <1>           ; (BL   font block to load (EGA: 0-3; VGA: 0-7))
6249                                <1>         ; CX    number of characters to redefine (<=256)
6250                                <1>           ; DX    ASCII code of the first character defined at ES:BP
6251                                <1>           ; EBP      address of font-definition information
6252                                <1>         ;      (in user's memory space)
6253                                <1>
6254                                <1>         ; case 0x11:
6255                                <1>         ; switch(GET_AL())
6256                                <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)
0000310D 08C0           <1>       or    al, al ; 0
0000310F 7404           <1>       jz    short font_setup_0
00003111 3C10           <1>       cmp   al, 10h
00003113 7511           <1>       jne   short font_setup_1
                        <1> font_setup_0:
00003115 E8CE000000     <1>       call  transfer_user_fonts
0000311A 721C           <1>       jc    short font_setup_error
0000311C E8B2010000     <1>       call  load_text_user_pat
00003121 E931EAFFFF     <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;
00003126 3C01           <1>       cmp   al, 1
00003128 7404           <1>       je    short font_setup_2
0000312A 3C11           <1>       cmp   al, 11h
0000312C 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))
0000312E E8DC020000     <1>       call  load_text_8_14_pat
00003133 E91FEAFFFF     <1>         jmp     VIDEO_RETURN
                        <1> font_setup_error:
00003138 29C0           <1>       sub   eax, eax ; 0 -> fonts could not be loaded
0000313A E91DEAFFFF     <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;
0000313F 3C02           <1>       cmp   al, 2
00003141 7404           <1>       je    short font_setup_4
00003143 3C12           <1>       cmp   al, 12h
00003145 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))
00003147 E8F3020000     <1>       call  load_text_8_8_pat
0000314C E906EAFFFF     <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;
00003151 3C04           <1>       cmp   al, 4
00003153 7404           <1>       je    short font_setup_6
00003155 3C14           <1>       cmp   al, 14h
00003157 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))
00003159 E82A030000     <1>       call  load_text_8_16_pat
0000315E E9F4E9FFFF     <1>         jmp     VIDEO_RETURN
                        <1> font_setup_7:
                        <1>       ; Note: AX=1120h (Setup INT 1Fh, EXT_PTR) is not needed
                        <1>       ; for TRDOS 386 (TRDOS 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>
00003163 3C21           <1>       cmp   al, 21h
00003165 7531           <1>       jne   short font_setup_9
                        <1>
                        <1>       ; TRDOS 386 modification !
                        <1>       ; dh = 0 -> 256 characters
                        <1>       ; dh = 80h -> second 128 characters
                        <1>       ; dh = 0FFh -> first 128 characters
                        <1>
                        <1>       ; 09/01/2021 (TRDOS 386 v2.0.3)
                        <1>       ;push  ebx
00003167 51             <1>       push  ecx
00003168 52             <1>       push  edx
00003169 30D2           <1>       xor   dl, dl
0000316B 88CF           <1>       mov   bh, cl ; character height
0000316D 66B90001       <1>       mov   cx, 100h ; 256
00003171 08F6           <1>       or    dh, dh ; 0
00003173 7410           <1>       jz    short  font_setup_8
00003175 FECD           <1>       dec   ch ; cx = 0
00003177 80FEFF         <1>       cmp   dh, 0FFh
0000317A 7409           <1>       je    short font_setup_8 ; 1st 128 chars
```

```
                                <1>       ; 2nd 128 chars
0000317C 80FE80                 <1>       cmp    dh, 80h
0000317F 75B7                   <1>       jne    short font_setup_error ; invalid !
00003181 88F1                   <1>       mov    cl, dh
00003183 86D6                   <1>       xchg   dl, dh
                                <1>       ; number of chars, cx = 80h
                                <1>       ; start char, dl = 80h
                                <1> font_setup_8:
00003185 E85E000000             <1>       call   transfer_user_fonts
0000318A 5A                     <1>       pop    edx
0000318B 59                     <1>       pop    ecx
                                <1>       ;pop   ebx
0000318C 72AA                   <1>       jc     short font_setup_error
                                <1>       ; ebp = user's font data address in system's memory space
0000318E E83F030000             <1>       call   load_gfx_user_chars
00003193 E9BFE9FFFF             <1>         jmp     VIDEO_RETURN
                                <1> font_setup_9:
                                <1>       ; case 0x22:
                                <1>         ; biosfn_load_gfx_8_14_chars(GET_BL());
                                <1>         ; break;
00003198 3C22                   <1>       cmp    al, 22h
0000319A 750A                   <1>       jne    short font_setup_10
0000319C E86D030000             <1>       call   load_gfx_8_14_chars
000031A1 E9B1E9FFFF             <1>         jmp     VIDEO_RETURN
                                <1> font_setup_10:
                                <1>       ; case 0x23:
                                <1>         ; biosfn_load_gfx_8_8_dd_chars(GET_BL());
                                <1>         ; break;
000031A6 3C23                   <1>       cmp    al, 23h
000031A8 750A                   <1>       jne    short font_setup_11
000031AA E8A0030000             <1>       call   load_gfx_8_8_chars
000031AF E9A3E9FFFF             <1>         jmp     VIDEO_RETURN
                                <1> font_setup_11:
                                <1>       ; case 0x24:
                                <1>         ; biosfn_load_gfx_8_16_chars(GET_BL());
                                <1>         ; break;
000031B4 3C24                   <1>       cmp    al, 24h
000031B6 750A                   <1>       jne    short font_setup_12
000031B8 E8D3030000             <1>       call   load_gfx_8_16_chars
000031BD E995E9FFFF             <1>         jmp     VIDEO_RETURN
                                <1> font_setup_12:
                                <1>       ; case 0x30:
                                <1>         ; biosfn_get_font_info(GET_BH(),&ES,&BP,&CX,&DX);
                                <1>         ; break;
000031C2 3C30                   <1>       cmp    al, 30h
000031C4 750A                   <1>       jne    short font_setup_13
000031C6 E806040000             <1>       call   get_font_info
                                <1>       ; eax = return value (info: 4 bytes for 4 parms)
                                <1>       ; eax = 0 -> invalid function (input)
000031CB E98CE9FFFF             <1>         jmp     _video_return
                                <1> font_setup_13:
000031D0 3C03                   <1>       cmp    al, 03h ; AX = 1103h
000031D2 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)
000031D4 66BAC403               <1>       mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
                                <1>       ;mov   ah, bl
000031D8 28E4                   <1>       sub    ah, ah ; 0
                                <1>       ;mov   al, 03h
000031DA 66EF                   <1>       out    dx, ax
000031DC E976E9FFFF             <1>       jmp     VIDEO_RETURN
                                <1>
                                <1> font_setup_14:
000031E1 29C0                   <1>       sub    eax, eax ; 0 = invalid function
000031E3 E974E9FFFF             <1>         jmp     _video_return
                                <1>
                                <1> transfer_user_fonts:
                                <1>       ; 19/01/2021
                                <1>       ; 09/01/2021
                                <1>       ; 05/01/2021 (TRDOS 386 v2.0.3)
                                <1>
                                <1>       ; BH    height of each character (bytes per character)
                                <1>       ; CX    number of characters to redefine (<=256)
                                <1>         ; DX    ASCII code of the first character defined at EBP
                                <1>         ; EBP      address of font-definition information
                                <1>       ;      (in user's memory space)
                                <1>
                                <1>       ; Modified registers: eax, edx, ecx, esi, edi, ebp
                                <1>       ;
                                <1>       ; output:
                                <1>       ;      ebp = user font data address in system memory
                                <1>
000031E8 81E2FFFF0000           <1>       and    edx, 0FFFFh
000031EE 81E1FFFF0000           <1>       and    ecx, 0FFFFh
000031F4 7537                   <1>       jnz    short transfer_user_fonts_5
                                <1>
000031F6 09D2                   <1>       or     edx, edx
000031F8 7531                   <1>       jnz    short transfer_user_fonts_4
000031FA 09ED                   <1>       or     ebp, ebp
000031FC 752D                   <1>       jnz    short transfer_user_fonts_4
                                <1>
                                <1>       ; cx = 0, dx = 0, ebp = 0
                                <1>       ; copy system font to user font
                                <1>
000031FE B140                   <1>       mov    cl, 64 ; 64 dwords
                                <1>
00003200 80FF10                 <1>       cmp    bh, 16
00003203 7417                   <1>       je     short transfer_user_fonts_2
00003205 80FF08                 <1>       cmp    bh, 8
00003208 7406                   <1>       je     short transfer_user_fonts_1
                                <1>
0000320A BD[54680100]           <1>       mov    ebp, vgafont14
0000320F C3                     <1>       retn
```

```
6467                                <1>
6468                                <1> transfer_user_fonts_1:
6469 00003210 BF00500900           <1>         mov     edi, VGAFONT8USER
6470 00003215 BE[54600100]         <1>         mov     esi, vgafont8
6471 0000321A EB0A                 <1>         jmp     short transfer_user_fonts_3
6472                                <1>
6473                                <1> transfer_user_fonts_2:
6474 0000321C BF00400900           <1>         mov     edi, VGAFONT16USER
6475 00003221 BE[54760100]         <1>         mov     esi, vgafont16
6476                                <1> transfer_user_fonts_3:
6477 00003226 89FD                 <1>         mov     ebp, edi
6478 00003228 F3A5                 <1>         rep     movsd
6479 0000322A C3                   <1>         retn
6480                                <1>
6481                                <1> transfer_user_fonts_4:
6482 0000322B F9                   <1>         stc
6483 0000322C C3                   <1>         retn
6484                                <1>
6485                                <1> transfer_user_fonts_5:
6486 0000322D 09ED                 <1>         or      ebp, ebp
6487 0000322F 74FA                 <1>         jz      short transfer_user_fonts_4 ; invalid address !
6488                                <1>
6489 00003231 6681F90001           <1>         cmp     cx, 256
6490 00003236 77F3                 <1>         ja      short transfer_user_fonts_4
6491 00003238 29D1                 <1>         sub     ecx, edx
6492 0000323A 76EF                 <1>         jna     short transfer_user_fonts_4
6493                                <1>
6494 0000323C 80FF0E               <1>         cmp     bh, 14 ; 8x14 font
6495                                <1>                        ; (there is not an alternative buffer)
6496 0000323F 7525                 <1>         jne     short transfer_user_fonts_6
6497                                <1>
6498                                <1>         ; use system's 8x14 font space if permission flag is 1
6499 00003241 F605[7E120300]80     <1>         test    byte [ufont], 80h
6500 00003248 74E1                 <1>         jz      short transfer_user_fonts_4 ; not allowed
6501                                <1>
6502                                <1>         ; permission is given (for vgafont14 location etc.)
6503                                <1>         ; (for permanent font modification)
6504                                <1>         ;
6505                                <1>         ; 19/01/2021
6506                                <1>         ; Note: Permission flag can be set by 'root' while
6507                                <1>         ;       system is not in multi tasking/user mode
6508                                <1>         ;       while [multi_tasking] = 0 and [u.uid] = 0
6509                                <1>
6510 0000324A 52                   <1>         push    edx
6511 0000324B 30E4                 <1>         xor     ah, ah
6512 0000324D 88F8                 <1>         mov     al, bh ; mov al, 14
6513 0000324F 66F7E1               <1>         mul     cx
6514                                <1>                 ; ecx = byte count
6515 00003252 89C1                 <1>         mov     ecx, eax
6516 00003254 5A                   <1>         pop     edx
6517 00003255 30E4                 <1>         xor     ah, ah
6518 00003257 88F8                 <1>         mov     al, bh ; mov ax, 14 ; bytes per character
6519 00003259 66F7E2               <1>         mul     dx
6520 0000325C 6689C2               <1>         mov     dx, ax ; char offset
6521 0000325F BF[54680100]         <1>         mov     edi, vgafont14
6522 00003264 EB4C                 <1>         jmp     short transfer_user_fonts_8
6523                                <1> transfer_user_fonts_6:
6524 00003266 80FF08               <1>         cmp     bh, 8 ; 8x8 font
6525 00003269 7522                 <1>         jne     short transfer_user_fonts_7 ; 8x16 font
6526 0000326B 66C1E203             <1>         shl     dx, 3 ; * 8
6527 0000326F 66C1E103             <1>         shl     cx, 3 ; * 8
6528                                <1>         ; 09/01/2021
6529 00003273 BF00500900           <1>         mov     edi, VGAFONT8USER
6530 00003278 F605[7E120300]08     <1>         test    byte [ufont], 8  ; already loaded ?
6531 0000327F 7531                 <1>         jnz     short transfer_user_fonts_8 ; yes
6532 00003281 BE[54600100]         <1>         mov     esi, vgafont8
6533 00003286 E83B000000           <1>         call    transfer_user_fonts_10
6534 0000328B EB25                 <1>         jmp     short transfer_user_fonts_8
6535                                <1> transfer_user_fonts_7:
6536 0000328D 80FF10               <1>         cmp     bh, 16 ; 8x16 font
6537 00003290 7599                 <1>         jne     short transfer_user_fonts_4  ; invalid !
6538 00003292 66C1E204             <1>         shl     dx, 4 ; * 16
6539 00003296 66C1E104             <1>         shl     cx, 4 ; * 16
6540 0000329A BF00400900           <1>         mov     edi, VGAFONT16USER
6541 0000329F F605[7E120300]10     <1>         test    byte [ufont], 16  ; already loaded ?
6542 000032A6 750A                 <1>         jnz     short transfer_user_fonts_8 ; yes
6543 000032A8 BE[54760100]         <1>         mov     esi, vgafont16
6544 000032AD E814000000           <1>         call    transfer_user_fonts_10
6545                                <1> transfer_user_fonts_8:
6546 000032B2 01D7                 <1>         add     edi, edx ; char offset
6547                                <1>         ; 09/07/2016
6548                                <1>         ;and    ecx, 0FFFFh
6549                                <1>         ; ECX = byte count
6550                                <1>         ;push   ecx
6551 000032B4 89EE                 <1>         mov     esi, ebp ; user's font buffer
6552                                <1>         ; 09/01/2021
6553 000032B6 89FD                 <1>         mov     ebp, edi ; system addr for user's font
6554                                <1>         ; 05/01/2021
6555                                <1>         ;mov    edi, Cluster_Buffer  ; system buffer
6556 000032B8 E87FE80000           <1>         call    transfer_from_user_buffer
6557                                <1>         ;pop    ecx
6558                                <1>         ; ecx = transfer (byte) count = character count
6559 000032BD 7206                 <1>         jc      short transfer_user_fonts_9
6560                                <1>         ; 05/01/2021
6561                                <1>         ;mov    ebp, Cluster_Buffer
6562                                <1>
6563 000032BF 083D[7E120300]       <1>         or      byte [ufont], bh
6564                                <1>                         ; 8x8 or 8x16 user font ready
6565                                <1> transfer_user_fonts_9:
6566 000032C5 C3                   <1>         retn
6567                                <1>
6568                                <1> transfer_user_fonts_10:
6569                                <1>         ; 09/01/2021
6570 000032C6 56                   <1>         push    esi
6571 000032C7 57                   <1>         push    edi
```

```
6572 000032C8 51                <1>       push   ecx
6573 000032C9 66B94000          <1>       mov    cx, 64
6574 000032CD F3A5              <1>       rep    movsd
6575 000032CF 59                <1>       pop    ecx
6576 000032D0 5F                <1>       pop    edi
6577 000032D1 5E                <1>       pop    esi
6578 000032D2 C3                <1>       retn
6579                            <1>
6580                            <1> load_text_user_pat:
6581                            <1>       ; 26/07/2016
6582                            <1>       ; 09/07/2016
6583                            <1>       ; load user defined (EGA/VGA) text fonts
6584                            <1>       ;
6585                            <1>       ; derived from 'Plex86/Bochs VGABios' source code
6586                            <1>       ; vgabios-0.7a (2011)
6587                            <1>       ; by the LGPL VGABios developers Team (2001-2008)
6588                            <1>       ; 'vgabios.c', 'biosfn_load_text_user_pat'
6589                            <1>
6590                            <1>       ; biosfn_load_text_user_pat (AL,ES,BP,CX,DX,BL,BH)
6591                            <1>
6592                            <1>       ; get_font_access();
6593                            <1>       ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
6594                            <1>       ; for(i=0;i<CX;i++)
6595                            <1>       ; {
6596                            <1>       ;  src = BP + i * BH;
6597                            <1>       ;  dest = blockaddr + (DX + i) * 32;
6598                            <1>       ;  memcpyb(0xA000, dest, ES, src, BH);
6599                            <1>       ; }
6600                            <1>       ; release_font_access();
6601                            <1>       ; if(AL>=0x10)
6602                            <1>       ; {
6603                            <1>       ; set_scan_lines(BH);
6604                            <1>       ; }
6605                            <1>
6606 000032D3 50                <1>       push   eax
6607 000032D4 E83C000000        <1>       call   get_font_access
6608 000032D9 28DB              <1>       sub    bl, bl ; i = 0
6609                            <1> ltup_1:
6610 000032DB 88D8              <1>       mov    al, bl
6611 000032DD F6E7              <1>       mul    bh
6612 000032DF 0FB7F0            <1>       movzx  esi, ax
6613 000032E2 01EE              <1>       add    esi, ebp
6614 000032E4 88D8              <1>       mov    al, bl
6615 000032E6 28E4              <1>       sub    ah, ah
6616 000032E8 6601D0            <1>       add    ax, dx ; (DX + i)
6617 000032EB 66C1E005          <1>       shl    ax, 5  ; * 32
6618 000032EF 0FB7F8            <1>       movzx  edi, ax
6619 000032F2 81C700000A00      <1>       add    edi, 0A0000h
6620 000032F8 51                <1>       push   ecx
6621 000032F9 0FB6CF            <1>       movzx  ecx, bh
6622 000032FC F3A4              <1>       rep    movsb
6623 000032FE 59                <1>       pop    ecx
6624 000032FF FEC3              <1>       inc    bl
6625 00003301 38CB              <1>       cmp    bl, cl
6626 00003303 75D6              <1>       jne    short ltup_1
6627                            <1>       ;
6628 00003305 E840000000        <1>       call   release_font_access
6629                            <1>       ;
6630 0000330A 58                <1>       pop    eax
6631                            <1>       ; if(AL>=0x10)
6632 0000330B 3C10              <1>       cmp    al, 10h
6633 0000330D 7205              <1>       jb     short ltup_2
6634                            <1>       ; set_scan_lines(BH);
6635 0000330F E875000000        <1>       call   set_scan_lines
6636                            <1> ltup_2:
6637 00003314 C3                <1>       retn
6638                            <1>
6639                            <1> get_font_access:
6640                            <1>       ; 09/07/2016
6641                            <1>       ;
6642                            <1>       ; derived from 'Plex86/Bochs VGABios' source code
6643                            <1>       ; vgabios-0.7a (2011)
6644                            <1>       ; by the LGPL VGABios developers Team (2001-2008)
6645                            <1>       ; 'vgabios.c', 'get_font_access'
6646                            <1>
6647                            <1>       ; get_font_access()
6648 00003315 52                <1>       push   edx
6649 00003316 66BAC403          <1>       mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
6650 0000331A 66B80001          <1>       mov    ax, 0100h
6651 0000331E 66EF              <1>       out    dx, ax
6652 00003320 66B80204          <1>       mov    ax, 0402h
6653 00003324 66EF              <1>       out    dx, ax
6654 00003326 66B80407          <1>       mov    ax, 0704h
6655 0000332A 66EF              <1>       out    dx, ax
6656 0000332C 66B80003          <1>       mov    ax, 0300h
6657 00003330 66EF              <1>       out    dx, ax
6658 00003332 66BACE03          <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
6659 00003336 66B80402          <1>       mov    ax, 0204h
6660 0000333A 66EF              <1>       out    dx, ax
6661 0000333C 66B80500          <1>       mov    ax, 0005h
6662 00003340 66EF              <1>       out    dx, ax
6663 00003342 66B80604          <1>       mov    ax, 0406h
6664 00003346 66EF              <1>       out    dx, ax
6665 00003348 5A                <1>       pop    edx
6666 00003349 C3                <1>       retn
6667                            <1>
6668                            <1> release_font_access:
6669                            <1>       ; 29/07/2016
6670                            <1>       ; 09/07/2016
6671                            <1>       ;
6672                            <1>       ; derived from 'Plex86/Bochs VGABios' source code
6673                            <1>       ; vgabios-0.7a (2011)
6674                            <1>       ; by the LGPL VGABios developers Team (2001-2008)
6675                            <1>       ; 'vgabios.c', 'release_font_access'
6676                            <1>
```

```
6677 0000334A 66BAC403          <1>       mov   dx, 3C4h ; VGAREG_SEQU_ADDRESS
6678 0000334E 66B80001          <1>       mov   ax, 0100h
6679 00003352 66EF              <1>       out   dx, ax
6680 00003354 66B80203          <1>       mov   ax, 0302h
6681 00003358 66EF              <1>       out   dx, ax
6682 0000335A 66B80403          <1>       mov   ax, 0304h
6683 0000335E 66EF              <1>       out   dx, ax
6684 00003360 66B80003          <1>       mov   ax, 0300h
6685 00003364 66EF              <1>       out   dx, ax
6686 00003366 66BACC03          <1>       mov   dx, 3CCh ; VGAREG_READ_MISC_OUTPUT
6687 0000336A EC                <1>       in    al, dx
6688 0000336B 2401              <1>       and   al, 01h
6689 0000336D C0E002            <1>       shl   al, 2
6690 00003370 0C0A              <1>       or    al, 0Ah
6691 00003372 88C4              <1>       mov   ah, al
6692 00003374 B006              <1>       mov   al, 06h
6693 00003376 66BACE03          <1>       mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
6694 0000337A 66EF              <1>       out   dx, ax
6695 0000337C 66B80400          <1>       mov   ax, 0004h
6696 00003380 66EF              <1>       out   dx, ax
6697 00003382 66B80510          <1>       mov   ax, 1005h
6698 00003386 66EF              <1>       out   dx, ax
6699 00003388 C3                <1>       retn
6700                            <1>
6701                            <1> set_scan_lines:
6702                            <1>       ; 09/07/2016
6703                            <1>       ;
6704                            <1>       ; derived from 'Plex86/Bochs VGABios' source code
6705                            <1>       ; vgabios-0.7a (2011)
6706                            <1>       ; by the LGPL VGABios developers Team (2001-2008)
6707                            <1>       ; 'vgabios.c', 'set_scan_lines'
6708                            <1>
6709                            <1>       ; set scan lines(lines)
6710                            <1>       ; BH = lines
6711                            <1>
6712                            <1>       ; outb(crtc_addr, 0x09);
6713 00003389 66BAD403            <1>       mov   dx, 3D4h ; CRTC_ADDRESS = 3D4h (always)
6714 0000338D B009              <1>       mov   al, 09h
6715 0000338F EE                <1>       out   dx, al
6716                            <1>       ; crtc_r9 = inb(crtc_addr+1);
6717 00003390 6642              <1>       inc   dx ; 3D5h
6718 00003392 EC                <1>       in    al, dx
6719                            <1>       ; crtc_r9 = (crtc_r9 & 0xe0) | (lines - 1);
6720 00003393 24E0              <1>       and   al, 0E0h
6721 00003395 FECF              <1>       dec   bh ; lines - 1
6722 00003397 08F8              <1>       or    al, bh
6723                            <1>       ; outb(crtc_addr+1, crtc_r9);
6724 00003399 EE                <1>       out   dx, al
6725                            <1>       ;inc  bh
6726                            <1>       ; if(lines==8)
6727                            <1>       ;cmp  bh, 8
6728 0000339A 80FF07            <1>       cmp   bh, 7
6729 0000339D 7506              <1>       jne   short ssl_1
6730                            <1>       ; biosfn_set_cursor_shape(0x06,0x07);
6731 0000339F 66B90706          <1>       mov   cx, 0607h
6732 000033A3 EB06              <1>       jmp   short ssl_2
6733                            <1> ssl_1:
6734                            <1>       ; biosfn_set_cursor_shape(lines-4,lines-3);
6735 000033A5 88F9              <1>       mov   cl, bh ; lines - 1
6736 000033A7 88CD              <1>       mov   ch, cl ; lines - 1 (16 -> 15)
6737 000033A9 FECD              <1>       dec   ch  ; lines - 2 (16 -> 14)
6738                            <1> ssl_2:
6739                            <1>       ; CH = start line, CL = stop line
6740 000033AB B40A              <1>       mov   ah, 10 ; 6845 register for cursor set
6741 000033AD 66890D[E36F0000]  <1>       mov   [CURSOR_MODE], cx ; save in data area
6742 000033B4 E809F0FFFF        <1>       call  m16    ; output cx register
6743                            <1>       ; write_word(BIOSMEM_SEG,BIOSMEM_CHAR_HEIGHT, lines);
6744 000033B9 FEC7              <1>       inc   bh ; lines
6745 000033BB 883D[CE6F0000]    <1>       mov   [CHAR_HEIGHT], bh
6746                            <1>       ;  outb(crtc_addr, 0x12);
6747 000033C1 66BAD403          <1>       mov   dx, 3D4h ; CRTC_ADDRESS
6748 000033C5 B012              <1>       mov   al, 12h
6749 000033C7 EE                <1>       out   dx, al
6750                            <1>       ; vde = inb(crtc_addr+1);
6751 000033C8 6642              <1>       inc   dx
6752 000033CA EC                <1>       in    al, dx
6753 000033CB 88C4              <1>       mov   ah, al
6754                            <1>       ; outb(crtc_addr, 0x07);
6755 000033CD 664A              <1>       dec   dx
6756 000033CF B007              <1>       mov   al, 07h
6757 000033D1 EE                <1>       out   dx, al
6758                            <1>       ; ovl = inb(crtc_addr+1);
6759 000033D2 6642              <1>       inc   dx
6760 000033D4 EC                <1>       in    al, dx
6761                            <1>       ; vde += (((ovl & 0x02) << 7) + ((ovl & 0x40) << 3) + 1);
6762 000033D5 88E2              <1>       mov   dl, ah ; vde
6763 000033D7 88C6              <1>       mov   dh, al ; ovl
6764 000033D9 6683E002          <1>       and   ax, 02h
6765 000033DD 66C1E007          <1>       shl   ax, 7
6766 000033E1 6689C1            <1>       mov   cx, ax ; (ovl & 0x02) << 7)
6767 000033E4 88F0              <1>       mov   al, dh ; ovl
6768 000033E6 6683E040          <1>       and   ax, 40h
6769 000033EA 66C1E003          <1>       shl   ax, 3  ; (ovl & 0x40) << 3)
6770 000033EE 6640              <1>       inc   ax ; + 1
6771 000033F0 6601C8            <1>       add   ax, cx
6772 000033F3 30F6              <1>       xor   dh, dh
6773 000033F5 6601D0            <1>       add   ax, dx ; + vde
6774                            <1>       ; rows = vde / lines;
6775 000033F8 F6F7              <1>       div   bh
6776                            <1>       ;dec  al ; rows -1
6777                            <1>       ; write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, rows-1);
6778 000033FA A2[D26F0000]      <1>       mov   [VGA_ROWS], al ; rows (not 'rows-1' !)
6779                            <1>       ; write_word(BIOSMEM_SEG,BIOSMEM_PAGE_SIZE, rows * cols * 2);
6780                            <1>       ;mov  ah, [CRT_COLS]
6781                            <1>       ;mul  ah
```

```
                                <1>       ; 17/11/2020
000033FF F625[CC6F0000]         <1>       mul    byte [CRT_COLS]
00003405 66D1E0                 <1>       shl    ax, 1
00003408 66A3[58960100]         <1>       mov    [CRT_LEN], ax
0000340E 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>
0000340F 50                     <1>       push   eax
00003410 E800FFFFFF             <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>
00003415 28DB                   <1>       sub    bl, bl ; i = 0
00003417 B70E                   <1>       mov    bh, 14
00003419 BE[54680100]           <1>       mov    esi, vgafont14
0000341E 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
00003423 0FB6CF                 <1>       movzx  ecx, bh
00003426 F3A4                   <1>       rep    movsb
00003428 83C712                 <1>       add    edi, 18 ; 32 - 14
0000342B FEC3                   <1>       inc    bl
0000342D 75F4                   <1>       jnz    short lt8_14_1
                                <1>       ;
0000342F E816FFFFFF             <1>       call   release_font_access
                                <1>       ;
00003434 58                     <1>       pop    eax
                                <1>       ; if(AL>=0x10)
00003435 3C10                   <1>       cmp    al, 10h
00003437 7205                   <1>       jb     short lt8_14_4
                                <1>       ; BH = 14
                                <1>       ; set_scan_lines(14);
00003439 E84BFFFFFF             <1>       call   set_scan_lines
                                <1> lt8_14_4:
0000343E C3                     <1>       retn
                                <1>
                                <1> load_text_8_8_pat:
                                <1>       ; 05/01/2021 (TRDOS 386 v2.0.3)
                                <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>       ; }
```

```
6887                                    <1>     ; release_font_access();
6888                                    <1>     ; if(AL>=0x10)
6889                                    <1>     ; {
6890                                    <1>     ; set_scan_lines(8);
6891                                    <1>     ; }
6892                                    <1>
6893 0000343F 50                        <1>     push   eax
6894 00003440 E8D0FEFFFF                <1>     call   get_font_access
6895                                    <1>
6896                                    <1>     ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
6897                                    <1>     ;mov   dl, bl
6898                                    <1>     ;and   dl, 3
6899                                    <1>     ;shl   dx, 14
6900                                    <1>     ;xchg  dx, bx
6901                                    <1>     ;and   dl, 4
6902                                    <1>     ;shl   dx, 11
6903                                    <1>     ;add   dx, bx
6904                                    <1>
6905                                    <1>     ;xor   dx, dx  ; blockaddr = 0
6906                                    <1>     ; Always block 0 for TRDOS 386 ! (blockaddr=0)
6907                                    <1>
6908 00003445 28DB                     <1>     sub    bl, bl ; i = 0
6909 00003447 B708                     <1>     mov    bh, 8
6910                                    <1>     ;mov   esi, vgafont8
6911 00003449 BF00000A00               <1>     mov    edi, 0A0000h
6912                                    <1>
6913                                    <1>     ; 05/01/2021
6914 0000344E F605[7E120300]80         <1>     test   byte [ufont], 80h
6915 00003455 7410                     <1>     jz     short lt8_8_0
6916                                    <1>            ; user font permission (after set mode)
6917 00003457 F605[7E120300]08         <1>     test   byte [ufont], 8
6918 0000345E 7407                     <1>     jz     short lt8_8_0
6919 00003460 BE00500900               <1>     mov    esi, VGAFONT8USER
6920 00003465 EB05                     <1>     jmp    short lt8_8_1
6921                                    <1> lt8_8_0:
6922 00003467 BE[54600100]             <1>     mov    esi, vgafont8
6923                                    <1> lt8_8_1:
6924                                    <1>     ;mov   al, bl
6925                                    <1>     ;mul   bh
6926                                    <1>     ;movzx esi, ax
6927                                    <1>     ;add   esi, vgafont8
6928                                    <1>     ;mov   al, bl
6929                                    <1>     ;sub   ah, ah
6930                                    <1>     ;shl   ax, 5 ; * 32
6931                                    <1>     ;;add  ax, dx ; blockaddr + i * 32;
6932                                    <1>     ;movzx edi, ax ; dest
6933                                    <1>     ;add   edi, 0A0000h
6934 0000346C 0FB6CF                   <1>     movzx  ecx, bh
6935 0000346F F3A4                     <1>     rep    movsb
6936 00003471 83C718                   <1>     add    edi, 24 ; 32 - 8
6937 00003474 FEC3                     <1>     inc    bl
6938 00003476 75F4                     <1>     jnz    short lt8_8_1
6939                                    <1>     ;
6940 00003478 E8CDFEFFFF               <1>     call   release_font_access
6941                                    <1>     ;
6942 0000347D 58                       <1>     pop    eax
6943                                    <1>     ; if(AL>=0x10)
6944 0000347E 3C10                     <1>     cmp    al, 10h
6945 00003480 7205                     <1>     jb     short lt8_8_2
6946                                    <1>     ; BH = 8
6947                                    <1>     ; set_scan_lines(8);
6948 00003482 E802FFFFFF               <1>     call   set_scan_lines
6949                                    <1> lt8_8_2:
6950 00003487 C3                       <1>     retn
6951                                    <1>
6952                                    <1> load_text_8_16_pat:
6953                                    <1>     ; 05/01/2021 (TRDOS 386 v2.0.3)
6954                                    <1>     ; 26/07/2016
6955                                    <1>     ; 25/07/2016
6956                                    <1>     ; 23/07/2016
6957                                    <1>     ; 09/07/2016
6958                                    <1>     ; load user defined (EGA/VGA) text fonts
6959                                    <1>     ;
6960                                    <1>     ; derived from 'Plex86/Bochs VGABios' source code
6961                                    <1>     ; vgabios-0.7a (2011)
6962                                    <1>     ; by the LGPL VGABios developers Team (2001-2008)
6963                                    <1>     ; 'vgabios.c', 'biosfn_load_text_8_16_pat'
6964                                    <1>
6965                                    <1>     ; biosfn_load_text_8_16_pat (AL,BL)
6966                                    <1>
6967                                    <1>     ; get_font_access();
6968                                    <1>     ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
6969                                    <1>     ; for(i=0;i<0x100;i++)
6970                                    <1>     ; {
6971                                    <1>     ;  src = i * 16;
6972                                    <1>     ;  dest = blockaddr + i * 32;
6973                                    <1>     ;  memcpyb(0xA000, dest, 0xC000, vgafont16+src, 16);
6974                                    <1>     ; }
6975                                    <1>     ; release_font_access();
6976                                    <1>     ; if(AL>=0x10)
6977                                    <1>     ; {
6978                                    <1>     ; set_scan_lines(16);
6979                                    <1>     ; }
6980                                    <1>
6981 00003488 50                       <1>     push   eax
6982 00003489 E887FEFFFF               <1>     call   get_font_access
6983                                    <1>
6984                                    <1>     ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
6985                                    <1>     ;mov   dl, bl
6986                                    <1>     ;and   dl, 3
6987                                    <1>     ;shl   dx, 14
6988                                    <1>     ;xchg  dx, bx
6989                                    <1>     ;and   dl, 4
6990                                    <1>     ;shl   dx, 11
6991                                    <1>     ;add   dx, bx
```

```
                                <1>
                                <1>       ;xor   dx, dx  ; blockaddr = 0
                                <1>       ; Always block 0 for TRDOS 386 ! (blockaddr=0)
                                <1>
0000348E 28DB                   <1>       sub    bl, bl ; i = 0
00003490 B710                   <1>       mov    bh, 16
                                <1>       ;mov   esi, vgafont16
00003492 BF00000A00             <1>       mov    edi, 0A0000h
00003497 0FB6C7                 <1>       movzx  eax, bh
                                <1>
                                <1>       ; 05/01/2021
0000349A F605[7E120300]80       <1>       test   byte [ufont], 80h
000034A1 7410                   <1>       jz     short lt8_16_0
                                <1>              ; user font permission (after set mode)
000034A3 F605[7E120300]10       <1>       test   byte [ufont], 16
000034AA 7407                   <1>       jz     short lt8_16_0
000034AC BE00400900             <1>       mov    esi, VGAFONT16USER
000034B1 EB05                   <1>       jmp    short lt8_16_1
                                <1> lt8_16_0:
000034B3 BE[54760100]           <1>       mov    esi, vgafont16
                                <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
000034B8 89C1                   <1>       mov    ecx, eax ; 16
000034BA F3A4                   <1>       rep    movsb
000034BC 01C7                   <1>       add    edi, eax ; add edi, 16
000034BE FEC3                   <1>       inc    bl
000034C0 75F6                   <1>       jnz    short lt8_16_1
                                <1>       ;
000034C2 E883FEFFFF             <1>       call   release_font_access
                                <1>       ;
000034C7 58                     <1>       pop    eax
                                <1>       ; if(AL>=0x10)
000034C8 3C10                   <1>       cmp    al, 10h
000034CA 7205                   <1>       jb     short lt8_16_2
                                <1>       ; BH = 16
                                <1>       ; set_scan_lines(16);
000034CC E8B8FEFFFF             <1>       call   set_scan_lines
                                <1> lt8_16_2:
000034D1 C3                     <1>       retn
                                <1>
                                <1> load_gfx_user_chars:
                                <1>       ; 08/01/2021
                                <1>       ; 05/01/2021 (TRDOS 386 v2.0.3)
                                <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);
                                <1>
                                <1>       ; 08/01/2021
                                <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>         ; EBP  address of font-definition information (user's mem space)
                                <1>
                                <1>       ; 05/01/2021
                                <1>       ;xor   eax, eax
                                <1>       ;dec   eax ; 0FFFFFFFFh (user defined fonts)
                                <1>       ;mov   [VGA_INT43H], eax
                                <1>
                                <1>       ; 08/01/2021
                                <1>       ; ebp = video font data (buffer) address
000034D2 892D[6A960100]         <1>       mov    [VGA_INT43H], ebp
                                <1>
                                <1>       ; switch (BL) {
                                <1>       ; case 0:
                                <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, DL-1);
                                <1>       ;    break;
000034D8 20DB                   <1>       and    bl, bl
000034DA 7508                   <1>       jnz    short l_gfx_uc_1
000034DC 8815[D26F0000]         <1>       mov    [VGA_ROWS], dl  ; not DL-1 !
000034E2 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;
000034E4 FECB                   <1>       dec    bl
000034E6 7509                   <1>       jnz    short l_gfx_uc_2
                                <1>       ; bl = 1
000034E8 C605[D26F0000]0E       <1>       mov    byte [VGA_ROWS], 14  ; not 13 !
000034EF EB16                   <1>       jmp    short l_gfx_uc_4
                                <1> l_gfx_uc_2:
000034F1 FECB                   <1>       dec    bl
```

```
7097 000034F3 740B              <1>       jz     short l_gfx_uc_3 ; bl = 2
7098 000034F5 FECB              <1>       dec    bl
7099 000034F7 750E              <1>       jnz    short l_gfx_uc_4 ; bl > 3
7100                            <1>       ; bl = 3
7101                            <1>       ; case 3:
7102                            <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 42);
7103                            <1>       ;    break;
7104 000034F9 C605[D26F0000]2B  <1>       mov    byte [VGA_ROWS], 43  ; not 42 !
7105                            <1> l_gfx_uc_3:
7106                            <1>       ; case 2:
7107                            <1>       ; default:
7108                            <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 24);
7109                            <1>       ;    break;
7110                            <1>       ; bl = 2 or bl > 3
7111 00003500 C605[D26F0000]19  <1>       mov    byte [VGA_ROWS], 25  ; not 24 !
7112                            <1>       ; }
7113                            <1> l_gfx_uc_4:
7114                            <1>       ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, CX);
7115 00003507 880D[CE6F0000]      <1>       mov    [CHAR_HEIGHT], cl
7116                            <1>       ; }
7117 0000350D C3                <1>       retn
7118                            <1>
7119                            <1> load_gfx_8_14_chars:
7120                            <1>       ; 08/08/2016
7121                            <1>       ; 10/07/2016
7122                            <1>       ; Setup ROM 8x14 Font for Graphics Mode (VGA)
7123                            <1>       ;
7124                            <1>       ; derived from 'Plex86/Bochs VGABios' source code
7125                            <1>       ; vgabios-0.7a (2011)
7126                            <1>       ; by the LGPL VGABios developers Team (2001-2008)
7127                            <1>       ; 'vgabios.c', 'biosfn_load_gfx_8_14_chars'
7128                            <1>
7129                            <1>       ; biosfn_load_gfx_8_14_chars (BL)
7130                            <1>       ; /* set 0x43 INT pointer */
7131                            <1>       ; write_word(0x0, 0x43*4, &vgafont14);
7132                            <1>       ; write_word(0x0, 0x43*4+2, 0xC000);
7133 0000350E C705[6A960100]-    <1>       mov    dword [VGA_INT43H], vgafont14
7133 00003514 [54680100]        <1>
7134                            <1>
7135                            <1>       ; BL    screen rows code: 00H = user-specified (in DL)
7136                            <1>         ;                       01H = 14 rows
7137                            <1>         ;                       02H = 25 rows
7138                            <1>         ;                       03H = 43 rows
7139                            <1>         ; DL    (when BL=0) custom number of char rows on screen
7140                            <1>
7141                            <1>       ; switch (BL) {
7142                            <1>       ; case 0:
7143                            <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, DL-1);
7144                            <1>       ;    break;
7145 00003518 20DB              <1>       and    bl, bl
7146 0000351A 7508              <1>       jnz    short l_gfx_8_14c_1
7147 0000351C 8815[D26F0000]    <1>       mov    [VGA_ROWS], dl  ; not DL-1 !
7148 00003522 EB23              <1>       jmp    short l_gfx_8_14c_4
7149                            <1> l_gfx_8_14c_1:
7150                            <1>       ; case 1:
7151                            <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 13);
7152                            <1>       ;    break;
7153 00003524 FECB              <1>       dec    bl
7154 00003526 7509              <1>       jnz    short l_gfx_8_14c_2
7155                            <1>       ; bl = 1
7156 00003528 C605[D26F0000]0E  <1>       mov    byte [VGA_ROWS], 14  ; not 13 !
7157 0000352F EB16              <1>       jmp    short l_gfx_8_14c_4
7158                            <1> l_gfx_8_14c_2:
7159 00003531 FECB              <1>       dec    bl
7160 00003533 740B              <1>       jz     short l_gfx_8_14c_3 ; bl = 2
7161 00003535 FECB              <1>       dec    bl
7162 00003537 750E              <1>       jnz    short l_gfx_8_14c_4 ; bl > 3
7163                            <1>       ; bl = 3
7164                            <1>       ; case 3:
7165                            <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 42);
7166                            <1>       ;    break;
7167 00003539 C605[D26F0000]2B  <1>       mov    byte [VGA_ROWS], 43  ; not 42 !
7168                            <1> l_gfx_8_14c_3:
7169                            <1>       ; case 2:
7170                            <1>       ; default:
7171                            <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 24);
7172                            <1>       ;    break;
7173                            <1>       ; bl = 2 or bl > 3
7174 00003540 C605[D26F0000]19  <1>       mov    byte [VGA_ROWS], 25  ; not 24 !
7175                            <1>       ; }
7176                            <1> l_gfx_8_14c_4:
7177                            <1>       ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, 14);
7178 00003547 C605[CE6F0000]0E  <1>         mov     byte [CHAR_HEIGHT], 14
7179                            <1>       ; }
7180 0000354E C3                <1>       retn
7181                            <1>
7182                            <1> load_gfx_8_8_chars:
7183                            <1>       ; 08/08/2016
7184                            <1>       ; 10/07/2016
7185                            <1>       ; Setup ROM 8x14 Font for Graphics Mode (VGA)
7186                            <1>       ;
7187                            <1>       ; derived from 'Plex86/Bochs VGABios' source code
7188                            <1>       ; vgabios-0.7a (2011)
7189                            <1>       ; by the LGPL VGABios developers Team (2001-2008)
7190                            <1>       ; 'vgabios.c', 'biosfn_load_gfx_8_8_dd_chars'
7191                            <1>
7192                            <1>       ; biosfn_load_gfx_8_8_dd_chars (BL)
7193                            <1>       ; /* set 0x43 INT pointer */
7194                            <1>       ; write_word(0x0, 0x43*4, &vgafont8);
7195                            <1>       ; write_word(0x0, 0x43*4+2, 0xC000);
7196 0000354F C705[6A960100]-    <1>       mov    dword [VGA_INT43H], vgafont8
7196 00003555 [54600100]        <1>
7197                            <1>
7198                            <1>       ; BL    screen rows code: 00H = user-specified (in DL)
7199                            <1>         ;                       01H = 14 rows
```

```
7200                                    <1>         ;                           02H = 25 rows
7201                                    <1>         ;                           03H = 43 rows
7202                                    <1>         ; DL    (when BL=0) custom number of char rows on screen
7203                                    <1>
7204                                    <1>       ; switch (BL) {
7205                                    <1>       ; case 0:
7206                                    <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, DL-1);
7207                                    <1>       ;    break;
7208 00003559 20DB                      <1>       and    bl, bl
7209 0000355B 7508                      <1>       jnz    short l_gfx_8_8c_1
7210 0000355D 8815[D26F0000]            <1>       mov    [VGA_ROWS], dl  ; not DL-1 !
7211 00003563 EB23                      <1>       jmp    short l_gfx_8_8c_4
7212                                    <1> l_gfx_8_8c_1:
7213                                    <1>       ; case 1:
7214                                    <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 13);
7215                                    <1>       ;    break;
7216 00003565 FECB                      <1>       dec    bl
7217 00003567 7509                      <1>       jnz    short l_gfx_8_8c_2
7218                                    <1>       ; bl = 1
7219 00003569 C605[D26F0000]0E          <1>       mov    byte [VGA_ROWS], 14  ; not 13 !
7220 00003570 EB16                      <1>       jmp    short l_gfx_8_8c_4
7221                                    <1> l_gfx_8_8c_2:
7222 00003572 FECB                      <1>       dec    bl
7223 00003574 740B                      <1>       jz     short l_gfx_8_8c_3 ; bl = 2
7224 00003576 FECB                      <1>       dec    bl
7225 00003578 750E                      <1>       jnz    short l_gfx_8_8c_4 ; bl > 3
7226                                    <1>       ; bl = 3
7227                                    <1>       ; case 3:
7228                                    <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 42);
7229                                    <1>       ;    break;
7230 0000357A C605[D26F0000]2B          <1>       mov    byte [VGA_ROWS], 43  ; not 42 !
7231                                    <1> l_gfx_8_8c_3:
7232                                    <1>       ; case 2:
7233                                    <1>       ; default:
7234                                    <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 24);
7235                                    <1>       ;    break;
7236                                    <1>       ; bl = 2 or bl > 3
7237 00003581 C605[D26F0000]19          <1>       mov    byte [VGA_ROWS], 25  ; not 24 !
7238                                    <1>       ; }
7239                                    <1> l_gfx_8_8c_4:
7240                                    <1>       ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, 8);
7241 00003588 C605[CE6F0000]08          <1>         mov     byte [CHAR_HEIGHT], 8
7242                                    <1>       ; }
7243 0000358F C3                        <1>       retn
7244                                    <1>
7245                                    <1> load_gfx_8_16_chars:
7246                                    <1>       ; 08/08/2016
7247                                    <1>       ; 10/07/2016
7248                                    <1>       ; Setup ROM 8x14 Font for Graphics Mode (VGA)
7249                                    <1>       ;
7250                                    <1>       ; derived from 'Plex86/Bochs VGABios' source code
7251                                    <1>       ; vgabios-0.7a (2011)
7252                                    <1>       ; by the LGPL VGABios developers Team (2001-2008)
7253                                    <1>       ; 'vgabios.c', 'biosfn_load_gfx_8_16_chars'
7254                                    <1>
7255                                    <1>       ; biosfn_load_gfx_8_16_chars (BL)
7256                                    <1>       ; /* set 0x43 INT pointer */
7257                                    <1>       ; write_word(0x0, 0x43*4, &vgafont16);
7258                                    <1>       ; write_word(0x0, 0x43*4+2, 0xC000);
7259 00003590 C705[6A960100]-           <1>       mov    dword [VGA_INT43H], vgafont16
7259 00003596 [54760100]                <1>
7260                                    <1>
7261                                    <1>       ; BL    screen rows code: 00H = user-specified (in DL)
7262                                    <1>         ;                           01H = 14 rows
7263                                    <1>         ;                           02H = 25 rows
7264                                    <1>         ;                           03H = 43 rows
7265                                    <1>         ; DL    (when BL=0) custom number of char rows on screen
7266                                    <1>
7267                                    <1>       ; switch (BL) {
7268                                    <1>       ; case 0:
7269                                    <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, DL-1);
7270                                    <1>       ;    break;
7271 0000359A 20DB                      <1>       and    bl, bl
7272 0000359C 7508                      <1>       jnz    short l_gfx_8_16c_1
7273 0000359E 8815[D26F0000]            <1>       mov    [VGA_ROWS], dl  ; not DL-1 !
7274 000035A4 EB23                      <1>       jmp    short l_gfx_8_16c_4
7275                                    <1> l_gfx_8_16c_1:
7276                                    <1>       ; case 1:
7277                                    <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 13);
7278                                    <1>       ;    break;
7279 000035A6 FECB                      <1>       dec    bl
7280 000035A8 7509                      <1>       jnz    short l_gfx_8_16c_2
7281                                    <1>       ; bl = 1
7282 000035AA C605[D26F0000]0E          <1>       mov    byte [VGA_ROWS], 14  ; not 13 !
7283 000035B1 EB16                      <1>       jmp    short l_gfx_8_16c_4
7284                                    <1> l_gfx_8_16c_2:
7285 000035B3 FECB                      <1>       dec    bl
7286 000035B5 740B                      <1>       jz     short l_gfx_8_16c_3 ; bl = 2
7287 000035B7 FECB                      <1>       dec    bl
7288 000035B9 750E                      <1>       jnz    short l_gfx_8_16c_4 ; bl > 3
7289                                    <1>       ; bl = 3
7290                                    <1>       ; case 3:
7291                                    <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 42);
7292                                    <1>       ;    break;
7293 000035BB C605[D26F0000]2B          <1>       mov    byte [VGA_ROWS], 43  ; not 42 !
7294                                    <1> l_gfx_8_16c_3:
7295                                    <1>       ; case 2:
7296                                    <1>       ; default:
7297                                    <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 24);
7298                                    <1>       ;    break;
7299                                    <1>       ; bl = 2 or bl > 3
7300 000035C2 C605[D26F0000]19          <1>       mov    byte [VGA_ROWS], 25  ; not 24 !
7301                                    <1>       ; }
7302                                    <1> l_gfx_8_16c_4:
7303                                    <1>       ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, 16);
```

```
7304 000035C9 C605[CE6F0000]10    <1>         mov     byte [CHAR_HEIGHT], 16
7305                              <1>       ; }
7306 000035D0 C3                  <1>       retn
7307                              <1>
7308                              <1> get_font_info:
7309                              <1>       ; 08/01/2021 (TRDOS 386 v2.0.3)
7310                              <1>       ; 19/09/2016
7311                              <1>       ; 08/08/2016
7312                              <1>       ; 10/07/2016
7313                              <1>       ; Get Current Character Generator Info (VGA)
7314                              <1>       ;
7315                              <1>       ; derived from 'Plex86/Bochs VGABios' source code
7316                              <1>       ; vgabios-0.7a (2011)
7317                              <1>       ; by the LGPL VGABios developers Team (2001-2008)
7318                              <1>       ; 'vgabios.c', 'biosfn_get_font_info'
7319                              <1>
7320                              <1>       ; Modified for TRDOS 386 !
7321                              <1>       ;
7322                              <1>       ; INPUT ->
7323                              <1>       ;    AX = 1130h
7324                              <1>       ;    BL = 0 -> Get info for current VGA font
7325                              <1>       ;              (BH = unused)
7326                              <1>       ;    19/09/2016
7327                              <1>       ;    BL > 0 -> Get requested character font data
7328                              <1>       ;       BL = 1 -> vgafont8
7329                              <1>       ;         BL = 2 -> vgafont14
7330                              <1>       ;       BL = 3 -> vgafont16
7331                              <1>       ;      ;08/01/2021
7332                              <1>       ;       BL = 4 -> user defined 8x8 font
7333                              <1>       ;       BL = 5 -> user defined 8x14 font
7334                              <1>       ;       BL = 6 -> user defined 8x16 font
7335                              <1>       ;       BL > 6 -> Invalid function (for now!)
7336                              <1>       ;       BH = ASCII code of the first character
7337                              <1>       ;       ECX = Number of characters from the 1st char
7338                              <1>       ;       ECX >= 256 -> All (256-BH) characters
7339                              <1>       ;       ECX = 0 -> All characters (BH = unused)
7340                              <1>       ;         EDX = User's Buffer Address
7341                              <1>       ; OUTPUT ->
7342                              <1>       ;    AL = height (scanlines), bytes per character
7343                              <1>       ;    AH = screen rows
7344                              <1>       ;    Byte 16-23 of EAX = number of columns
7345                              <1>       ;    Byte 24-31 of EAX =
7346                              <1>       ;        0 -> default font (not configured yet)
7347                              <1>       ;        0FFh -> user defined font
7348                              <1>       ;        14 = vgafont14
7349                              <1>       ;         8 = vgafont8
7350                              <1>       ;        16 = vgafont16
7351                              <1>       ;   If BL input > 0 ->
7352                              <1>       ;       EAX = Actual transfer count
7353                              <1>       ;
7354 000035D1 20DB                <1>       and   bl, bl
7355 000035D3 740F                <1>       jz    short gfi_1
7356                              <1>       ; invalid function (input)
7357                              <1>       ; 08/01/2021
7358 000035D5 80FB04              <1>       cmp   bl, 4
7359 000035D8 7263                <1>       jb    short gfi_5
7360 000035DA 7441                <1>       je    short gfi_3
7361 000035DC 80FB06              <1>       cmp   bl, 6
7362 000035DF 744C                <1>       je    short gfi_4
7363                              <1>       ; bh = 5 or bh > 6
7364                              <1> gfi_0:
7365 000035E1 31C0                <1>       xor   eax, eax ; 0
7366 000035E3 C3                  <1>       retn
7367                              <1> gfi_1:
7368 000035E4 A0[CE6F0000]        <1>       mov   al, [CHAR_HEIGHT]
7369 000035E9 8A25[D26F0000]      <1>       mov   ah, [VGA_ROWS]
7370 000035EF C1E010              <1>       shl   eax, 16
7371 000035F2 A0[CC6F0000]        <1>       mov   al, [CRT_COLS]
7372 000035F7 8B0D[6A960100]      <1>       mov   ecx, [VGA_INT43H]
7373 000035FD 21C9                <1>       and   ecx, ecx
7374 000035FF 7418                <1>       jz    short gfi_2 ; 0 = default font
7375                              <1>       ; 08/01/2021
7376 00003601 FECC                <1>       dec   ah ; 0FFh
7377 00003603 81F900400900        <1>       cmp   ecx, VGAFONT16USER
7378 00003609 740E                <1>       je    short gfi_2
7379 0000360B 81F900500900        <1>       cmp   ecx, VGAFONT8USER
7380 00003611 7406                <1>       je    short gfi_2
7381 00003613 8A25[CE6F0000]      <1>       mov   ah, [CHAR_HEIGHT] ; font size = height
7382                              <1> gfi_2:
7383 00003619 C1C010              <1>       rol   eax, 16
7384 0000361C C3                  <1>       retn
7385                              <1> gfi_3:
7386                              <1>       ; 08/01/2021
7387 0000361D F605[7E120300]08    <1>       test  byte [ufont], 08h ; 8x8 user font
7388 00003624 74BB                <1>       jz    short gfi_0 ; not loaded !
7389 00003626 BE00500900          <1>       mov   esi, VGAFONT8USER ; *
7390                              <1>       ;mov  bl, 8
7391                              <1>       ;jmp  short gfi_8
7392 0000362B EB4D                <1>       jmp   short gfi_10
7393                              <1> gfi_4:
7394                              <1>       ; 08/01/2021
7395 0000362D F605[7E120300]10    <1>       test  byte [ufont], 10h ; 8x16 user font
7396 00003634 74AB                <1>       jz    short gfi_0 ; not loaded !
7397 00003636 BE00400900          <1>       mov   esi, VGAFONT16USER ; *
7398 0000363B EB15                <1>       jmp   short gfi_7
7399                              <1> gfi_5:
7400 0000363D 80FB02              <1>       cmp   bl, 2
7401 00003640 7233                <1>       jb    short gfi_9
7402 00003642 7709                <1>       ja    short gfi_6
7403                              <1>       ;BL = 2 -> vgafont14
7404 00003644 BE[54680100]        <1>       mov   esi, vgafont14 ; *
7405 00003649 B30E                <1>       mov   bl, 14
7406 0000364B EB07                <1>       jmp   short gfi_8
7407                              <1> gfi_6:
7408                              <1>       ;BL = 3 -> vgafont16
```

```
7409 0000364D BE[54760100]        <1>       mov   esi, vgafont16 ; *
7410                              <1> gfi_7:
7411 00003652 B310                <1>       mov   bl, 16
7412                              <1> gfi_8:
7413 00003654 89D7                <1>       mov   edi, edx ; **
7414 00003656 09C9                <1>       or    ecx, ecx
7415 00003658 7424                <1>       jz    short gfi_11 ; all chars from the 00h
7416 0000365A 88F8                <1>       mov   al, bh ; character index
7417 0000365C F6E3                <1>       mul   bl ; char index * char height/size
7418 0000365E 0FB7D0              <1>       movzx edx, ax
7419 00003661 01D6                <1>       add   esi, edx ; *
7420 00003663 66BAFF00            <1>       mov   dx, 255
7421 00003667 28FA                <1>       sub   dl, bh
7422 00003669 6642                <1>       inc   dx
7423 0000366B 39D1                <1>       cmp   ecx, edx
7424 0000366D 770F                <1>       ja    short gfi_11
7425 0000366F 7412                <1>       je    short gfi_12
7426 00003671 89D1                <1>       mov   ecx, edx
7427 00003673 EB0E                <1>       jmp   short gfi_12
7428                              <1> gfi_9:
7429                              <1>       ;BL = 1 -> vgafont8
7430 00003675 BE[54600100]        <1>       mov   esi, vgafont8 ; *
7431                              <1> gfi_10:
7432 0000367A B308                <1>       mov   bl, 8
7433 0000367C EBD6                <1>       jmp   short gfi_8
7434                              <1> gfi_11:
7435 0000367E B900010000          <1>       mov   ecx, 256
7436                              <1> gfi_12:
7437                              <1>       ; 08/01/2021
7438 00003683 89C8                <1>       mov   eax, ecx ; character count
7439 00003685 30FF                <1>       xor   bh, bh
7440 00003687 66F7E3              <1>       mul   bx ; char count * char height/size
7441 0000368A 89C1                <1>       mov   ecx, eax
7442                              <1>
7443                              <1>       ; ESI = source address in system space
7444                              <1>       ; EDI = user's buffer address
7445                              <1>       ; ECX = transfer (byte) count
7446 0000368C E861E40000          <1>       call  transfer_to_user_buffer
7447 00003691 89C8                <1>       mov   eax, ecx ; actual transfer count
7448 00003693 C3                  <1>       retn
7449                              <1>
7450                              <1> vga_pal_funcs:
7451                              <1>       ; 10/08/2016
7452                              <1>       ; VGA Palette functions
7453                              <1>       ;
7454                              <1>       ; derived from 'Plex86/Bochs VGABios' source code
7455                              <1>       ; vgabios-0.7a (2011)
7456                              <1>       ; by the LGPL VGABios developers Team (2001-2008)
7457                              <1>       ; 'vgabios.c', 'vgarom.asm'
7458                              <1>
7459 00003694 3C00                <1>       cmp   al, 0
7460 00003696 0F848F000000        <1>         je      set_single_palette_reg
7461                              <1> vga_palf_1001:
7462 0000369C 3C01                <1>       cmp   al, 1
7463 0000369E 0F84B4000000        <1>         je      set_overscan_border_color
7464                              <1> vga_palf_1002:
7465 000036A4 3C02                <1>       cmp   al, 2
7466 000036A6 0F84B0000000        <1>         je      set_all_palette_reg
7467                              <1> vga_palf_1003:
7468 000036AC 3C03                <1>       cmp   al, 3
7469 000036AE 0F84E8000000        <1>         je      toggle_intensity
7470                              <1> vga_palf_1007:
7471 000036B4 3C07                <1>       cmp   al, 7
7472 000036B6 0F840D010000        <1>         je      get_single_palette_reg
7473 000036BC 7266                <1>       jb    short vga_palf_unknown
7474                              <1> vga_palf_1008:
7475 000036BE 3C08                <1>       cmp   al, 8
7476 000036C0 0F8437010000        <1>         je      read_overscan_border_color
7477                              <1> vga_palf_1009:
7478 000036C6 3C09                <1>       cmp   al, 9
7479 000036C8 0F8433010000        <1>       je    get_all_palette_reg
7480                              <1> vga_palf_1010:
7481 000036CE 3C10                <1>       cmp   al, 10h
7482 000036D0 0F8487010000        <1>       je    set_single_dac_reg
7483 000036D6 724C                <1>       jb    short vga_palf_unknown
7484                              <1> vga_palf_1012:
7485 000036D8 3C12                <1>       cmp   al, 12h
7486 000036DA 0F8498010000        <1>         je      set_all_dac_reg
7487 000036E0 7242                <1>       jb    short vga_palf_unknown
7488                              <1> vga_palf_1013:
7489 000036E2 3C13                <1>       cmp   al, 13h
7490 000036E4 0F84CC010000        <1>         je      select_video_dac_color_page
7491                              <1> vga_palf_1015:
7492 000036EA 3C15                <1>       cmp   al, 15h
7493 000036EC 0F8412020000        <1>       je    read_single_dac_reg
7494 000036F2 7230                <1>       jb    short vga_palf_unknown
7495                              <1> vga_palf_1017:
7496 000036F4 3C17                <1>       cmp   al, 17h
7497 000036F6 0F8428020000        <1>         je      read_all_dac_reg
7498 000036FC 7226                <1>       jb    short vga_palf_unknown
7499                              <1> vga_palf_1018:
7500 000036FE 3C18                <1>       cmp   al, 18h
7501 00003700 0F845E020000        <1>         je      set_pel_mask
7502                              <1> vga_palf_1019:
7503 00003706 3C19                <1>       cmp   al, 19h
7504 00003708 0F8462020000        <1>         je      read_pel_mask
7505                              <1> vga_palf_101A:
7506 0000370E 3C1A                <1>       cmp   al, 1Ah
7507 00003710 0F8468020000        <1>         je      read_video_dac_state
7508                              <1> vga_palf_101B:
7509 00003716 3C1B                <1>       cmp   al, 1Bh
7510                              <1>       ;jne  short vga_palf_unknown
7511 00003718 770A                <1>       ja    short vga_palf_unknown
7512                              <1>
7513 0000371A E8C3F5FFFF          <1>       call  gray_scale_summing
```

```
7514 0000371F E933E4FFFF          <1>         jmp     VIDEO_RETURN
7515                              <1>
7516                              <1> vga_palf_unknown:
7517 00003724 29C0                <1>         sub     eax, eax ; 0 = invalid function
7518 00003726 E931E4FFFF          <1>         jmp     _video_return
7519                              <1>
7520                              <1> set_single_palette_reg:
7521                              <1>         ; 10/08/2016
7522                              <1>         ; Set One Palette Register
7523                              <1>         ; BL = register number to set
7524                              <1>         ;      (a 4-bit attribute nibble: 00h-0Fh)
7525                              <1>         ; BH = 6-bit RGB color to display
7526                              <1>         ;       for that attribute
7527                              <1>
7528 0000372B 80FB14              <1>         cmp     bl, 14h
7529                              <1>         ;ja     short no_actl_reg1
7530 0000372E 0F8723E4FFFF        <1>         ja      VIDEO_RETURN
7531 00003734 6650                <1>         push    ax
7532 00003736 6652                <1>         push    dx
7533 00003738 66BADA03            <1>         mov     dx, 3DAh ; VGAREG_ACTL_RESET
7534 0000373C EC                  <1>         in      al, dx
7535 0000373D 66BAC003            <1>         mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
7536 00003741 88D8                <1>         mov     al, bl
7537 00003743 EE                  <1>         out     dx, al
7538 00003744 88F8                <1>         mov     al, bh
7539 00003746 EE                  <1>         out     dx, al
7540 00003747 B020                <1>         mov     al, 20h
7541 00003749 EE                  <1>         out     dx, al
7542                              <1>         ; ifdef VBOX
7543 0000374A 66BADA03            <1>         mov     dx, 3DAh ; VGAREG_ACTL_RESET
7544 0000374E EC                  <1>         in      al, dx
7545                              <1>         ; endif ; VBOX
7546 0000374F 665A                <1>         pop     dx
7547 00003751 6658                <1>         pop     ax
7548                              <1> ;no_actl_reg1:
7549 00003753 E9FFE3FFFF          <1>         jmp     VIDEO_RETURN
7550                              <1>
7551                              <1> set_overscan_border_color:
7552                              <1>         ; 10/08/2016
7553                              <1>         ; Set Overscan/Border Color Register
7554                              <1>         ; BH = 6-bit RGB color to display
7555                              <1>         ;       for that attribute
7556                              <1>
7557 00003758 B311                <1>         mov     bl, 11h
7558 0000375A EBCF                <1>         jmp     short set_single_palette_reg
7559                              <1>
7560                              <1> set_all_palette_reg:
7561                              <1>         ; 10/08/2016
7562                              <1>         ; Set All Palette Registers and Overscan
7563                              <1>         ; EDX = Address of 17 bytes;
7564                              <1>         ; an rgbRGB value for each of 16 palette
7565                              <1>         ; registers plus one for the border.
7566                              <1>
7567 0000375C 89D6                <1>         mov     esi, edx ; user buffer
7568 0000375E B911000000          <1>         mov     ecx, 17
7569 00003763 89E7                <1>         mov     edi, esp
7570 00003765 83EC14              <1>         sub     esp, 20
7571 00003768 E8CFE30000          <1>         call    transfer_from_user_buffer
7572                              <1>         ;jc     VIDEO_RETURN
7573                              <1>
7574 0000376D 66BADA03            <1>         mov     dx, 3DAh ; VGAREG_ACTL_RESET
7575 00003771 EC                  <1>         in      al, dx
7576 00003772 B100                <1>         mov     cl, 0
7577 00003774 66BAC003            <1>         mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
7578                              <1> set_palette_loop:
7579 00003778 88C8                <1>         mov     al, cl
7580 0000377A EE                  <1>         out     dx, al
7581 0000377B 8A07                <1>         mov     al, [edi]
7582 0000377D EE                  <1>         out     dx, al
7583 0000377E 47                  <1>         inc     edi
7584 0000377F FEC1                <1>         inc     cl
7585 00003781 80F910              <1>         cmp     cl, 10h
7586 00003784 75F2                <1>         jne     short set_palette_loop
7587 00003786 B011                <1>         mov     al, 11h
7588 00003788 EE                  <1>         out     dx, al
7589 00003789 8A07                <1>         mov     al, [edi]
7590 0000378B EE                  <1>         out     dx, al
7591 0000378C B020                <1>         mov     al, 20h
7592 0000378E EE                  <1>         out     dx, al
7593                              <1>         ; ifdef VBOX
7594 0000378F 66BADA03            <1>         mov     dx, 3DAh ; VGAREG_ACTL_RESET
7595 00003793 EC                  <1>         in      al, dx
7596                              <1>         ; endif ; VBOX
7597 00003794 83C414              <1>         add     esp, 20
7598 00003797 E9BBE3FFFF          <1>         jmp     VIDEO_RETURN
7599                              <1>
7600                              <1> toggle_intensity:
7601                              <1>         ; 10/08/2016
7602                              <1>         ; Select Foreground Blink or Bold Background
7603                              <1>         ; BL = 00h = enable bold backgrounds
7604                              <1>         ;          (16 background colors)
7605                              <1>           ;      01h = enable blinking foreground
7606                              <1>         ;          (8 background colors)
7607                              <1>
7608 0000379C 66BADA03            <1>         mov     dx, 3DAh ; VGAREG_ACTL_RESET
7609 000037A0 EC                  <1>         in      al, dx
7610 000037A1 66BAC003            <1>         mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
7611 000037A5 B010                <1>         mov     al, 10h
7612 000037A7 EE                  <1>         out     dx, al
7613 000037A8 66BAC103            <1>         mov     dx, 3C1h ; VGAREG_ACTL_READ_DATA
7614 000037AC EC                  <1>         in      al, dx
7615 000037AD 24F7                <1>         and     al, 0F7h
7616 000037AF 80E301              <1>         and     bl, 01h
7617 000037B2 C0E303              <1>         shl     bl, 3
7618 000037B5 08D8                <1>         or      al, bl
```

```
7619 000037B7 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
7620 000037BB EE                  <1>       out    dx, al
7621 000037BC B020                <1>       mov    al, 20h
7622 000037BE EE                  <1>       out    dx, al
7623                              <1>       ; ifdef VBOX
7624 000037BF 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
7625 000037C3 EC                  <1>       in     al, dx
7626                              <1>       ; endif ; VBOX
7627 000037C4 E98EE3FFFF          <1>       jmp    VIDEO_RETURN
7628                              <1>
7629                              <1> get_single_palette_reg:
7630                              <1>       ; 10/08/2016
7631                              <1>       ; Read One Palette Register
7632                              <1>         ; INPUT:
7633                              <1>       ; BL = Palette register to read (00h-0Fh)
7634                              <1>       ; OUTPUT:
7635                              <1>       ; BH = Current rgbRGB value of specified register
7636                              <1>       ;      for that attribute
7637                              <1>
7638 000037C9 80FB14              <1>       cmp    bl, 14h
7639                              <1>       ;ja    short no_actl_reg2
7640 000037CC 0F8785E3FFFF        <1>       ja     VIDEO_RETURN
7641                              <1>
7642 000037D2 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
7643 000037D6 EC                  <1>       in     al, dx
7644 000037D7 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
7645 000037DB 88D8                <1>       mov    al, bl
7646 000037DD EE                  <1>       out    dx, al
7647 000037DE 66BAC103            <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
7648 000037E2 EC                  <1>       in     al, dx
7649 000037E3 8844240D            <1>       mov    [esp+13], al ; bh
7650 000037E7 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
7651 000037EB EC                  <1>       in     al, dx
7652 000037EC 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
7653 000037F0 B020                <1>       mov    al, 20h
7654 000037F2 EE                  <1>       out    dx, al
7655                              <1>       ; ifdef VBOX
7656 000037F3 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
7657 000037F7 EC                  <1>       in     al, dx
7658                              <1>       ; endif ; VBOX
7659 000037F8 E95AE3FFFF          <1>       jmp    VIDEO_RETURN
7660                              <1>
7661                              <1> read_overscan_border_color:
7662                              <1>       ; 10/08/2016
7663                              <1>       ; Read Overscan Register
7664                              <1>       ; OUTPUT:
7665                              <1>       ; BH = current rgbRGB value
7666                              <1>       ;      of the overscan/border register
7667                              <1>
7668 000037FD B311                <1>       mov    bl, 11h
7669 000037FF EBC8                <1>       jmp    short get_single_palette_reg
7670                              <1>
7671                              <1> get_all_palette_reg:
7672                              <1>       ; 10/08/2016
7673                              <1>       ; Read All Palette Registers
7674                              <1>       ; EDX = Address of 17-byte buffer
7675                              <1>       ;      to receive data
7676                              <1>
7677 00003801 89D7                <1>       mov    edi, edx
7678 00003803 89E3                <1>       mov    ebx, esp
7679 00003805 89DE                <1>       mov    esi, ebx
7680 00003807 83EC14              <1>       sub    esp, 20
7681                              <1>
7682 0000380A B100                <1>       mov    cl, 0
7683                              <1> get_palette_loop:
7684 0000380C 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
7685 00003810 EC                  <1>       in     al, dx
7686 00003811 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
7687 00003815 88C8                <1>       mov    al, cl
7688 00003817 EE                  <1>       out    dx, al
7689 00003818 66BAC103            <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
7690 0000381C EC                  <1>       in     al, dx
7691 0000381D 8803                <1>       mov    [ebx], al
7692 0000381F 43                  <1>       inc    ebx
7693 00003820 FEC1                <1>       inc    cl
7694 00003822 80F910              <1>       cmp    cl, 10h
7695 00003825 75E5                <1>       jne    short get_palette_loop
7696 00003827 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
7697 0000382B EC                  <1>       in     al, dx
7698 0000382C 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
7699 00003830 B011                <1>       mov    al, 11h
7700 00003832 EE                  <1>       out    dx, al
7701 00003833 66BAC103            <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
7702 00003837 EC                  <1>       in     al, dx
7703 00003838 8803                <1>       mov    [ebx], al
7704 0000383A 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
7705 0000383E EC                  <1>       in     al, dx
7706 0000383F 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
7707 00003843 B020                <1>       mov    al, 20h
7708 00003845 EE                  <1>       out    dx, al
7709                              <1>       ; ifdef VBOX
7710 00003846 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
7711 0000384A EC                  <1>       in     al, dx
7712                              <1>       ; endif ; VBOX
7713                              <1>
7714 0000384B B911000000          <1>       mov    ecx, 17 ; transfer (byte) count
7715                              <1>       ; ESI = source address in system space
7716                              <1>       ; EDI = user's buffer address
7717 00003850 E89DE20000          <1>       call   transfer_to_user_buffer
7718                              <1>
7719 00003855 83C414              <1>       add    esp, 20
7720 00003858 E9FAE2FFFF          <1>       jmp    VIDEO_RETURN
7721                              <1>
7722                              <1> set_single_dac_reg:
7723                              <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>
0000385D 6652           <1>       push   dx
0000385F 66BAC803       <1>       mov    dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
00003863 88D8           <1>       mov    al, bl
00003865 EE             <1>       out    dx, al
                        <1>       ;mov   dx, 3C9h ; VGAREG_DAC_DATA
00003866 6642           <1>       inc    dx
00003868 6658           <1>       pop    ax
0000386A 88E0           <1>       mov    al, ah
0000386C EE             <1>       out    dx, al
0000386D 88E8           <1>       mov    al, ch
0000386F EE             <1>       out    dx, al
00003870 88C8           <1>       mov    al, cl
00003872 EE             <1>       out    dx, al
00003873 E9DFE2FFFF     <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>
00003878 89D6           <1>       mov    esi, edx ; user buffer
0000387A 89CA           <1>       mov    edx, ecx
0000387C 66D1E1         <1>       shl    cx, 1 ; *2
0000387F 01D1           <1>       add    ecx, edx ; ecx = 3*ecx
00003881 89E5           <1>       mov    ebp, esp
00003883 89EF           <1>       mov    edi, ebp
00003885 29CF           <1>       sub    edi, ecx
00003887 6683E7FC       <1>       and    di, 0FFFCh ; (dword alignment)
0000388B 89FC           <1>       mov    esp, edi
0000388D E8AAE20000     <1>       call   transfer_from_user_buffer
                        <1>       ;jc    VIDEO_RETURN
                        <1>
00003892 89D1           <1>       mov    ecx, edx
00003894 66BAC803       <1>       mov    dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
00003898 88D8           <1>       mov    al, bl
0000389A EE             <1>       out    dx, al
0000389B 66BAC903       <1>       mov    dx, 3C9h ; VGAREG_DAC_DATA
                        <1> set_dac_loop:
0000389F 8A07           <1>       mov    al, [edi]
000038A1 EE             <1>       out    dx, al
000038A2 47             <1>       inc    edi
000038A3 8A07           <1>       mov    al, [edi]
000038A5 EE             <1>       out    dx, al
000038A6 47             <1>       inc    edi
000038A7 8A07           <1>       mov    al, [edi]
000038A9 EE             <1>       out    dx, al
000038AA 47             <1>       inc    edi
000038AB 6649           <1>       dec    cx
000038AD 75F0           <1>       jnz    short set_dac_loop
000038AF 89EC           <1>       mov    esp, ebp
000038B1 E9A1E2FFFF     <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>
000038B6 66BADA03       <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
000038BA EC             <1>       in     al, dx
000038BB 66BAC003       <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
000038BF B010           <1>       mov    al, 10h
000038C1 EE             <1>       out    dx, al
000038C2 66BAC103       <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
000038C6 EC             <1>       in     al, dx
000038C7 80E301         <1>       and    bl, 01h
000038CA 750E           <1>       jnz    short set_dac_page
000038CC 247F           <1>       and    al, 07Fh
000038CE C0E707         <1>       shl    bh, 7
000038D1 08F8           <1>       or     al, bh
000038D3 66BAC003       <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
000038D7 EE             <1>       out    dx, al
000038D8 EB1D           <1>       jmp    short set_actl_normal
                        <1> set_dac_page:
000038DA 6650           <1>       push   ax
000038DC 66BADA03       <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
000038E0 EC             <1>       in     al, dx
000038E1 66BAC003       <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
000038E5 B014           <1>       mov    al, 14h
000038E7 EE             <1>       out    dx, al
000038E8 6658           <1>       pop    ax
000038EA 2480           <1>       and    al, 80h
000038EC 7503           <1>       jnz    short set_dac_16_page
000038EE C0E702         <1>       shl    bh, 2
                        <1> set_dac_16_page:
000038F1 80E70F         <1>       and    bh, 0Fh
000038F4 88F8           <1>       mov    al, bh
000038F6 EE             <1>       out    dx, al
```

```
                                <1> set_actl_normal:
000038F7 B020                   <1>       mov    al, 20h
000038F9 EE                     <1>       out    dx, al
                                <1>       ; ifdef VBOX
000038FA 66BADA03               <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
000038FE EC                     <1>       in     al, dx
                                <1>       ; endif ; VBOX
000038FF E953E2FFFF             <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>
00003904 66BAC703               <1>       mov    dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
00003908 88D8                   <1>       mov    al, bl
0000390A EE                     <1>       out    dx, al
0000390B 66BAC903               <1>       mov    dx, 3C9h ; VGAREG_DAC_DATA
0000390F EC                     <1>       in     al, dx
00003910 88442415               <1>       mov    [esp+21], al ; dh
00003914 EC                     <1>       in     al, dx
00003915 88C5                   <1>       mov    ch, al
00003917 EC                     <1>       in     al, dx
00003918 88C1                   <1>       mov    cl, al
0000391A 66894C2410             <1>       mov    [esp+16], cx ; cx
0000391F E933E2FFFF             <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>
00003924 89D7                   <1>       mov    edi, edx ; user buffer
00003926 89CA                   <1>       mov    edx, ecx
00003928 66D1E2                 <1>       shl    dx, 1 ; *2
0000392B 01CA                   <1>       add    edx, ecx ; edx = 3*ecx
0000392D 89E5                   <1>       mov    ebp, esp
0000392F 89EE                   <1>       mov    esi, ebp
00003931 29D6                   <1>       sub    esi, edx
00003933 6683E6FC               <1>       and    si, 0FFFCh ; (dword alignment)
00003937 89F4                   <1>       mov    esp, esi
00003939 52                     <1>       push   edx ; 3*ecx
0000393A 66BAC703               <1>       mov    dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
0000393E 88D8                   <1>       mov    al, bl
00003940 EE                     <1>       out    dx, al
00003941 66BAC903               <1>       mov    dx, 3C9h ; VGAREG_DAC_DATA
00003945 89F3                   <1>       mov    ebx, esi
                                <1> read_dac_loop:
00003947 EC                     <1>       in     al, dx
00003948 8803                   <1>       mov    [ebx], al
0000394A 43                     <1>       inc    ebx
0000394B EC                     <1>       in     al, dx
0000394C 8803                   <1>       mov    [ebx], al
0000394E 43                     <1>       inc    ebx
0000394F EC                     <1>       in     al, dx
00003950 8803                   <1>       mov    [ebx], al
00003952 43                     <1>       inc    ebx
00003953 6649                   <1>       dec    cx
00003955 75F0                   <1>       jnz    short read_dac_loop
00003957 59                     <1>       pop    ecx ; 3*ecx
                                <1>       ; ECX = transfer (byte) count
                                <1>       ; ESI = source address in system space
                                <1>       ; EDI = user's buffer address
00003958 E895E10000             <1>       call   transfer_to_user_buffer
0000395D 89EC                   <1>       mov    esp, ebp
0000395F E9F3E1FFFF             <1>       jmp    VIDEO_RETURN
                                <1>
                                <1> set_pel_mask:
                                <1>       ; 10/08/2016
                                <1>       ; BL = mask value
00003964 66BAC603               <1>       mov    dx, 3C6h ; VGAREG_PEL_MASK
00003968 88D8                   <1>       mov    al, bl
0000396A EE                     <1>       out    dx, al
0000396B E9E7E1FFFF             <1>       jmp    VIDEO_RETURN
                                <1>
                                <1> read_pel_mask:
                                <1>       ; 10/08/2016
                                <1>       ; Output: BL = mask value
00003970 66BAC603               <1>       mov    dx, 3C6h ; VGAREG_PEL_MASK
00003974 EC                     <1>       in     al, dx
00003975 8844240C               <1>       mov    [esp+12], al ; bl
00003979 E9D9E1FFFF             <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>
0000397E 66BADA03               <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
00003982 EC                     <1>       in     al, dx
00003983 66BAC003               <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
00003987 B010                   <1>       mov    al, 10h
00003989 EE                     <1>       out    dx, al
```

```
7934 0000398A 66BAC103            <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
7935 0000398E EC                  <1>       in     al, dx
7936 0000398F 88C3                <1>       mov    bl, al
7937 00003991 C0EB07              <1>       shr    bl, 7
7938 00003994 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
7939 00003998 EC                  <1>       in     al, dx
7940 00003999 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
7941 0000399D B014                <1>       mov    al, 14h
7942 0000399F EE                  <1>       out    dx, al
7943 000039A0 66BAC103            <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
7944 000039A4 EC                  <1>       in     al, dx
7945 000039A5 88C7                <1>       mov    bh, al
7946 000039A7 80E70F              <1>       and    bh, 0Fh
7947 000039AA F6C301              <1>       test   bl, 01
7948 000039AD 7503                <1>       jnz    short get_dac_16_page
7949 000039AF C0EF02              <1>       shr    bh, 2
7950                              <1> get_dac_16_page:
7951 000039B2 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
7952 000039B6 EC                  <1>       in     al, dx
7953 000039B7 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
7954 000039BB B020                <1>       mov    al, 20h
7955 000039BD EE                  <1>       out    dx, al
7956                              <1>       ; ifdef VBOX
7957 000039BE 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
7958 000039C2 EC                  <1>       in     al, dx
7959                              <1>       ; endif ; VBOX
7960 000039C3 66895C240C          <1>       mov    [esp+12], bx ; bx
7961 000039C8 E98AE1FFFF          <1>       jmp    VIDEO_RETURN
7962                              <1>
7963                              <1> ; 23/11/2020 - TRDOS 386 v2.0.3
7964                              <1> ; VBE 2 BOCHS/QEMU emulator extensions
7965                              <1> ;     for TRDOS 386 v2 kernel (video bios)
7966                              <1>
7967                              <1> ; BOCH/QEMU VBE2 VGA BIOS code
7968                              <1> ;     by Jeroen Janssen (2002)
7969                              <1> ;     by Volker Rupper (2003-2020)
7970                              <1> ; vbe.c (02/01/2020)
7971                              <1>
7972                              <1> ; vbe.h (02/01/2020)
7973                              <1>
7974                              <1> VBE_DISPI_BANK_ADDRESS    equ    0A0000h
7975                              <1> VBE_DISPI_BANK_SIZE_KB    equ    64
7976                              <1>
7977                              <1> VBE_DISPI_MAX_XRES equ    2560
7978                              <1> VBE_DISPI_MAX_YRES equ    1600
7979                              <1>
7980                              <1> VBE_DISPI_IOPORT_INDEX    equ    01CEh
7981                              <1> VBE_DISPI_IOPORT_DATA     equ    01CFh
7982                              <1>
7983                              <1> VBE_DISPI_INDEX_ID equ    00h
7984                              <1> VBE_DISPI_INDEX_XRES      equ    01h
7985                              <1> VBE_DISPI_INDEX_YRES      equ    02h
7986                              <1> VBE_DISPI_INDEX_BPP       equ    03h
7987                              <1> VBE_DISPI_INDEX_ENABLE    equ    04h
7988                              <1> VBE_DISPI_INDEX_BANK      equ    05h
7989                              <1> VBE_DISPI_INDEX_VIRT_WIDTH equ  06h
7990                              <1> VBE_DISPI_INDEX_VIRT_HEIGHT equ 07h
7991                              <1> VBE_DISPI_INDEX_X_OFFSET equ    08h
7992                              <1> VBE_DISPI_INDEX_Y_OFFSET equ    09h
7993                              <1> VBE_DISPI_INDEX_VIDEO_MEMORY_64K equ 0Ah
7994                              <1> VBE_DISPI_INDEX_DDC       equ    0Bh
7995                              <1>
7996                              <1> VBE_DISPI_ID0             equ    0B0C0h
7997                              <1> VBE_DISPI_ID1             equ    0B0C1h
7998                              <1> VBE_DISPI_ID2             equ    0B0C2h
7999                              <1> VBE_DISPI_ID3             equ    0B0C3h
8000                              <1> VBE_DISPI_ID4             equ    0B0C4h
8001                              <1> VBE_DISPI_ID5             equ    0B0C5h
8002                              <1>
8003                              <1> VBE_DISPI_DISABLED equ    00h
8004                              <1> VBE_DISPI_ENABLED  equ    01h
8005                              <1> VBE_DISPI_GETCAPS  equ    02h
8006                              <1> VBE_DISPI_8BIT_DAC equ    20h
8007                              <1> VBE_DISPI_LFB_ENABLED     equ    40h
8008                              <1> VBE_DISPI_NOCLEARMEM      equ    80h
8009                              <1>
8010                              <1> VBE_DISPI_LFB_PHYSICAL_ADDRESS equ 0E0000000h
8011                              <1>
8012                              <1> ; ***
8013                              <1>
8014                              <1> ;// VBE Return Status Info
8015                              <1> ;// AL
8016                              <1> VBE_RETURN_STATUS_SUPPORTED     equ    4Fh
8017                              <1> VBE_RETURN_STATUS_UNSUPPORTED   equ    00h
8018                              <1> ;// AH
8019                              <1> VBE_RETURN_STATUS_SUCCESSFULL   equ    00h
8020                              <1> VBE_RETURN_STATUS_FAILED  equ    01h
8021                              <1> VBE_RETURN_STATUS_NOT_SUPPORTED equ    02h
8022                              <1> VBE_RETURN_STATUS_INVALID equ    03h
8023                              <1>
8024                              <1> ;// VBE Mode Numbers
8025                              <1>
8026                              <1> VBE_MODE_VESA_DEFINED           equ    0100h
8027                              <1> VBE_MODE_REFRESH_RATE_USE_CRTC  equ    0800h
8028                              <1> VBE_MODE_LINEAR_FRAME_BUFFER    equ    4000h
8029                              <1> VBE_MODE_PRESERVE_DISPLAY_MEMORY equ   8000h
8030                              <1>
8031                              <1> ;// Mode Attributes
8032                              <1>
8033                              <1> VBE_MODE_ATTRIBUTE_SUPPORTED              equ 0001h
8034                              <1> VBE_MODE_ATTRIBUTE_EXTENDED_INFO_AVAILABLE equ      0002h
8035                              <1> VBE_MODE_ATTRIBUTE_COLOR_MODE             equ 0008h
8036                              <1> VBE_MODE_ATTRIBUTE_GRAPHICS_MODE   equ 0010h
8037                              <1> VBE_MODE_ATTRIBUTE_LINEAR_FRAME_BUFFER_MODE equ     0080h
8038                              <1> VBE_MODE_ATTRIBUTE_DOUBLE_SCAN_MODE       equ 0100h
```

```
8039                                    <1> VBE_MODE_ATTRIBUTE_INTERLACE_MODE         equ 0200h
8040                                    <1>
8041                                    <1> ;// Window attributes
8042                                    <1>
8043                                    <1> VBE_WINDOW_ATTRIBUTE_RELOCATABLE equ   01h
8044                                    <1> VBE_WINDOW_ATTRIBUTE_READABLE    equ   02h
8045                                    <1> VBE_WINDOW_ATTRIBUTE_WRITEABLE   equ   04h
8046                                    <1>
8047                                    <1> ;/* Video memory */
8048                                    <1> VGAMEM_GRAPH equ 0A000h
8049                                    <1> VGAMEM_CTEXT equ 0B800h
8050                                    <1> ;VGAMEM_MTEXT equ 0B000h
8051                                    <1>
8052                                    <1> ;// Memory model
8053                                    <1>
8054                                    <1> ;VBE_MEMORYMODEL_TEXT_MODE       equ    00h
8055                                    <1> ;VBE_MEMORYMODEL_CGA_GRAPHICS    equ    01h
8056                                    <1> ;VBE_MEMORYMODEL_PLANAR          equ    03h
8057                                    <1> VBE_MEMORYMODEL_PACKED_PIXEL     equ    04h
8058                                    <1> ;VBE_MEMORYMODEL_NON_CHAIN_4_256 equ    05h
8059                                    <1> VBE_MEMORYMODEL_DIRECT_COLOR     equ    06h
8060                                    <1> ;VBE_MEMORYMODEL_YUV             equ    07h
8061                                    <1>
8062                                    <1> ;// DirectColorModeInfo
8063                                    <1>
8064                                    <1> ;VBE_DIRECTCOLOR_COLOR_RAMP_PROGRAMMABLE equ 01h
8065                                    <1> VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE equ 02h
8066                                    <1>
8067                                    <1> VBE_DISPI_TOTAL_VIDEO_MEMORY_MB equ 16
8068                                    <1>
8069                                    <1> ; 24/11/2020
8070                                    <1> ; vbe.c
8071                                    <1>
8072                                    <1> %if 1
8073                                    <1>
8074                                    <1> _vbe_biosfn_return_mode_info:
8075                                    <1>       ; 15/12/2020
8076                                    <1>       ; 12/12/2020
8077                                    <1>       ; Return VBE Mode Information
8078                                    <1>       ; (call from 'sysvideo')
8079                                    <1>       ;
8080                                    <1>       ; Input:
8081                                    <1>       ;      cx = video (bios) mode
8082                                    <1>       ; Output:
8083                                    <1>       ;      cf = 0 -> (successful)
8084                                    <1>       ;         MODE_INFO_LIST addr contains MODEINFO
8085                                    <1>       ;      cf = 1 -> error
8086                                    <1>       ;
8087                                    <1>       ; Modified registers: eax, edx, edi
8088                                    <1>       ;
8089                                    <1>
8090                                    <1>       ; pushes for subroutine stack pops compatibility
8091                                    <1>
8092                                    <1>       ;push  ds  ; *
8093                                    <1>       ;push  es  ; **
8094                                    <1>
8095 000039CD 55                       <1>       push   ebp ; ***
8096 000039CE 56                       <1>       push   esi ; ****
8097                                    <1>
8098 000039CF 31FF                     <1>       xor    edi, edi  ; mov edi, 0
8099                                    <1>
8100 000039D1 803D[54090000]03         <1>       cmp    byte [vbe3], 3
8101 000039D8 7221                     <1>       jb     short _vbe_rmi_1
8102                                    <1>
8103                                    <1>       ;sub   edi, edi  ; 0 = kernel call (sign)
8104                                    <1>       ;      ; no transfer to user's buffer
8105                                    <1>
8106                                    <1>       ; cx = Video mode (for 4F01h, with LFB flag)
8107                                    <1>
8108 000039DA 66B8014F                 <1>       mov    ax, 4F01h
8109                                    <1>
8110 000039DE E820DFFFFF               <1>       call   _vbe3_pmfn_return_mode_info
8111                                    <1>
8112 000039E3 6683F84F                 <1>       cmp    ax, 004Fh
8113 000039E7 7533                     <1>       jne    short _vbe_rmi_2 ; fail
8114                                    <1>
8115                                    <1>       ; 15/12/2020
8116                                    <1>       ; cx = vbe video mode
8117 000039E9 80E501                   <1>       and    ch, 01h ; clear LFB flag
8118 000039EC BEFE7B0900               <1>       mov    esi, VBE3MODEINFOBLOCK - 2
8119 000039F1 66890E                   <1>       mov    [esi], cx ; MODEINFO.mode
8120 000039F4 E8A6000000               <1>       call   set_lfbinfo_table
8121 000039F9 EB22                     <1>       jmp    short _vbe_rmi_3  ; cf = 0
8122                                    <1> _vbe_rmi_1:
8123 000039FB 803D[54090000]02         <1>       cmp    byte [vbe3], 2
8124 00003A02 7219                     <1>       jb     short _vbe_rmi_3 ; cf = 1
8125 00003A04 A0[55090000]             <1>       mov    al, [vbe2bios] ; 0C0h-0C5h for emu (*)
8126 00003A09 3CC0                     <1>       cmp    al, 0C0h ; BOCHS/QEMU/VIRTUALBOX (*) ?
8127 00003A0B 7210                     <1>       jb     short _vbe_rmi_3  ; cf = 1
8128 00003A0D 3CC5                     <1>       cmp    al, 0C5h ; (*)
8129 00003A0F 770B                     <1>       ja     short _vbe_rmi_2 ; unknown vbios !?
8130                                    <1>
8131                                    <1>       ;xor   edi, edi  ; 0 = kernel call (sign)
8132                                    <1>       ;      ; no transfer to user's buffer
8133                                    <1>
8134                                    <1>       ;mov   ax, 4F01h
8135                                    <1>
8136                                    <1>       ; cx = Video mode (for 4F01h, with LFB flag)
8137                                    <1>
8138 00003A11 E80A000000               <1>       call   vbe_biosfn_return_mode_info
8139 00003A16 6683F84F                 <1>       cmp    ax, 004Fh ; successful ?
8140 00003A1A 7401                     <1>       je     short _vbe_rmi_3  ; cf = 0
8141                                    <1> _vbe_rmi_2:
8142 00003A1C F9                       <1>       stc
8143                                    <1>       ; cf = 1
```

```
                                <1> _vbe_rmi_3:
00003A1D 5E                     <1>         pop    esi ; ****
00003A1E 5D                     <1>         pop    ebp ; ***
                                <1>
                                <1>         ;pop   es  ; **
                                <1>         ;pop   ss  ; *
                                <1>
00003A1F C3                     <1>         retn
                                <1>
                                <1>
                                <1> ; * (TRDOS 386, INT 31h, VESA Video Bios functions)
                                <1> ; * ------------------------------------------------------------
                                <1> ; * Function 01h - Return VBE Mode Information
                                <1> ; * ------------------------------------------------------------
                                <1> ; * Input:
                                <1> ; *          AX = 4F01h
                                <1> ; *          CX = Mode number
                                <1> ; *     (ES:DI) EDI = Pointer to ModeInfoBlock structure
                                <1> ; * Output:
                                <1> ; *          AX = VBE Return Status
                                <1> ; *
                                <1> ; *------------------------------------------------------------
                                <1> ; *
                                <1> vbe_biosfn_return_mode_info:
                                <1>         ; 15/12/2020
                                <1>         ; 14/12/2020
                                <1>         ; 12/12/2020
                                <1>         ; 11/12/2020 (TRDOS 386 v2.0.3)
                                <1>         ;
                                <1>         ; Input:
                                <1>         ;      cx = video (bios) mode
                                <1>         ;      edi = ModeInfoBlock buffer address
                                <1>         ;           (in user's memory space)
                                <1>         ;      (ax = 4F01h)
                                <1>         ; Output:
                                <1>         ;      ax = 004Fh (successful)
                                <1>         ;      ah > 0 -> error
                                <1>         ;
                                <1>         ; Modified registers: esi
                                <1>
                                <1>         ;;push ds  ; *
                                <1>         ;;push es  ; **
                                <1>         ;;push ebp ; ***
                                <1>         ;;push esi ; ****
                                <1>
00003A20 F6C501                 <1>         test   ch, 1
00003A23 7505                   <1>         jnz    short vbe_rmi_1
                                <1>
                                <1>         ; mode number < 100h
                                <1>         ; CGA/VGA mode is not proper this VBE function
                                <1>
00003A25 29C0                   <1>         sub    eax, eax
                                <1> vbe_rmi_0:
                                <1>         ;mov   ax, 0100h  ; Function is not supported
00003A27 B401                   <1>         mov    ah, 1
00003A29 C3                     <1>         retn
                                <1> vbe_rmi_1:
00003A2A 52                     <1>         push   edx ; *****
00003A2B 51                     <1>         push   ecx ; ******
00003A2C 53                     <1>         push   ebx ; *******
00003A2D 57                     <1>         push   edi ; ********
                                <1>
                                <1>         ; 14/12/2020
00003A2E 89CB                   <1>         mov    ebx, ecx
                                <1>
                                <1>         ;xor   eax, eax
00003A30 80E7C1                 <1>         and    bh, 0C1h ; use bit 15, 14, 8 only (for bh)
00003A33 883D[1D120300]         <1>         mov    [vbe_mode_x], bh
                                <1>         ;and   bx, 1FFh
00003A39 80E701                 <1>         and    bh, 1
                                <1>         ;mov   bh, 1
                                <1>
                                <1>         ; Alternative 2 (instead of 'Mode_info_find_mode')
00003A3C E88A060000             <1>         call   set_mode_info_list ; (alternative 2)
                                <1>
                                <1>         ; eax = 0
                                <1>
                                <1>         ;mov   bx, [esi] ; mode
                                <1>
                                <1>         ; Alternative 1 (instead of 'set_mode_info_list')
                                <1>         ;call  mode_info_find_mode ; (alternative 1)
                                <1>
00003A41 09F6                   <1>         or     esi, esi
                                <1>         ; 14/12/2020
00003A43 744D                   <1>         jz     short vbe_rmi_4     ; VBE mode number is wrong
                                <1>                                    ; or it is not supported
                                <1>
                                <1>         ; 15/12/2020
                                <1>         ;mov   bx, [esi] ; mode
                                <1>
                                <1>         ; 12/12/2020
                                <1>         ;call  set_lfbinfo_table
                                <1>
00003A45 F605[1D120300]40       <1>         test   byte [vbe_mode_x], 40h ; LFB model ?
00003A4C 7404                   <1>         jz     short vbe_rmi_2
                                <1>
00003A4E C6461C01               <1>         mov    byte [esi+MODEINFO.NumberOfBanks], 1
                                <1> vbe_rmi_2:
                                <1>         ; (vbe.c, 02/01/2020, vruppert)
                                <1>         ; 11/12/2020 (Erdogan Tan, video.s)
                                <1>         ; Bochs Graphics Adapter
                                <1>         ; vendor_id: 1111h, device id: 1234h
                                <1>
00003A52 E855070000             <1>         call   pci_get_lfb_addr
```

```
8249                                    <1>       ;or    eax, eax
8250 00003A57 7404                      <1>       jz     short vbe_rmi_3
8251                                    <1>       ; zf = 0, ax > 0 (high word of LFB address)
8252                                    <1>       ; set/change LFB address in MODEINFO structure
8253 00003A59 6689462C                  <1>       mov    [esi+MODEINFO.PhysBasePtr+2], ax
8254                                    <1>       ; 12/12/2020
8255                                    <1>       ;mov   [edi+LFBINFO.LFB_addr+2], ax
8256                                    <1> vbe_rmi_3:
8257                                    <1>       ;test  byte [esi+MODEINFO.WinAAttributes], 1
8258                                    <1>       ;            ; VBE_WINDOW_ATTRIBUTE_RELOCATABLE = 1
8259                                    <1>         ;jz  short vbe_rmi_4
8260                                    <1>       ;; 11/12/2020
8261                                    <1>       ;; In fact, this is far call address in (Bochs/BGA) Video Bios
8262                                    <1>       ;; Direct user access to kernel subroutines is not possible
8263                                    <1>       ;; in TRDOS 386. Also, TRDOS 386 kernel will support only LFB.
8264                                    <1>       ;; Bank select may be a seperate sysvideo function in future
8265                                    <1>       ;; (if it will be required).
8266                                    <1>       ;mov   dword [esi+MODEINFO.WinFuncPtr], dispi_set_bank_farcall
8267                                    <1> ;vbe_rmi_4:
8268                                    <1>       ; 12/12/2020
8269 00003A5D E83D000000               <1>       call   set_lfbinfo_table
8270                                    <1>
8271                                    <1>       ; 11/12/2020
8272                                    <1>       ; copy 68 bytes of MODE_INFO_LIST to user
8273                                    <1>
8274 00003A62 8B3C24                    <1>       mov    edi, [esp]  ; user's buffer address
8275                                    <1>       ; 12/12/2020
8276 00003A65 09FF                      <1>       or     edi, edi ; 0 = kernel call
8277                                    <1>                       ; (call from '_vbe_biosfn_return_mode_info')
8278 00003A67 7432                      <1>       jz     short vbe_rmi_6
8279                                    <1>
8280                                    <1>       ; 15/12/2020
8281                                    <1>       ; prepare 256 bytes MODEINFO buffer at VBE3MODEINFOBLOCK
8282                                    <1>       ; and then, copy buffer conttent to user's buffer
8283 00003A69 57                        <1>       push   edi
8284 00003A6A BE[3C120300]              <1>       mov    esi, MODE_INFO_LIST + 2   ; MODEINFO.ModeAttributes
8285 00003A6F BF007C0900                <1>       mov    edi, VBE3MODEINFOBLOCK
8286 00003A74 B910000000                <1>       mov    ecx, 66/4  ; 66 bytes
8287 00003A79 F3A5                      <1>       rep    movsd
8288 00003A7B 31C0                      <1>       xor    eax, eax
8289 00003A7D B12F                      <1>       mov    cl, (256-68)/4 ; 188 bytes
8290 00003A7F F3AB                      <1>       rep    stosd
8291 00003A81 66AB                      <1>       stosw  ; 2 bytes
8292 00003A83 5F                        <1>       pop    edi
8293 00003A84 BE007C0900                <1>       mov    esi, VBE3MODEINFOBLOCK
8294                                    <1>       ;mov   cx, 256
8295 00003A89 FEC5                      <1>       inc    ch ; cx = 256
8296 00003A8B E862E00000                <1>       call   transfer_to_user_buffer
8297 00003A90 7309                      <1>       jnc    short vbe_rmi_5
8298                                    <1> vbe_rmi_4:
8299                                    <1>       ;mov   eax, 014Fh ; fail/error
8300 00003A92 31C0                      <1>       xor    eax, eax
8301 00003A94 B401                      <1>       mov    ah, 01h
8302                                    <1>       ;jmp   short vbe_rmi_6
8303 00003A96 E981000000                <1>       jmp    vbe_sm_ret1 ; 11/12/2020
8304                                    <1> vbe_rmi_5:
8305                                    <1>       ; 256 bytes of MODEINFO have been transferred to user
8306                                    <1>       ;mov   eax, 4Fh ; succesfull
8307                                    <1> vbe_rmi_6: ; 12/12/2020
8308 00003A9B 31C0                      <1>       xor    eax, eax
8309                                    <1> ;vbe_rmi_6:
8310 00003A9D EB7D                      <1>       jmp    vbe_sm_ret1 ; 11/12/2020
8311                                    <1>
8312                                    <1>       ;pop   edi ; ********
8313                                    <1>       ;pop   ebx ; *******
8314                                    <1>       ;pop   ecx ; ******
8315                                    <1>       ;pop   edx ; *****
8316                                    <1>
8317                                    <1>       ;;pop  esi ; ****
8318                                    <1>       ;;pop  ebp ; ***
8319                                    <1>       ;;pop  es  ; **
8320                                    <1>       ;;pop  ds  ; *
8321                                    <1>
8322                                    <1>       ;retn
8323                                    <1>
8324                                    <1> set_lfbinfo_table:
8325                                    <1>       ; 19/12/2020
8326                                    <1>       ; 11/12/2020
8327                                    <1>       ; Set/Fill LFBINFO structure/table
8328                                    <1>       ;
8329                                    <1>       ; Input:
8330                                    <1>       ;     esi = Mode info list address
8331                                    <1>       ; Output:
8332                                    <1>       ;     LFB_Info address is filled with LFBINFO
8333                                    <1>       ;     edi = LFB_Info address
8334                                    <1>       ;
8335                                    <1>       ; Modified registers: eax, edx (=0), edi
8336                                    <1>
8337 00003A9F BF[2A120300]              <1>       mov    edi, LFB_Info
8338 00003AA4 8B462A                    <1>       mov    eax, [esi+MODEINFO.PhysBasePtr]
8339 00003AA7 894702                    <1>       mov    [edi+LFBINFO.LFB_addr], eax ; LFB address
8340                                    <1>       ;mov   ax, [esi+MODEINFO.mode]
8341 00003AAA 668B06                    <1>       mov    ax, [esi]
8342 00003AAD 668907                    <1>       mov    [edi+LFBINFO.mode],ax
8343 00003AB0 8A461B                    <1>       mov    al, [esi+MODEINFO.BitsPerPixel]
8344 00003AB3 88470E                    <1>       mov    [edi+LFBINFO.bpp], al
8345 00003AB6 29C0                      <1>       sub    eax, eax
8346 00003AB8 668B4614                  <1>       mov    ax, [esi+MODEINFO.XResolution]
8347 00003ABC 6689470A                  <1>       mov    [edi+LFBINFO.X_res], ax
8348 00003AC0 89C2                      <1>       mov    edx, eax ; 19/12/2020
8349 00003AC2 668B4616                  <1>       mov    ax, [esi+MODEINFO.YResolution]
8350 00003AC6 6689470C                  <1>       mov    [edi+LFBINFO.Y_res], ax
8351                                    <1>       ; eax = Y_res ; screen height
8352                                    <1>       ; 19/12/2020
8353 00003ACA F7E2                      <1>       mul    edx ; X_res*Y_res
```

```
8354                                    <1>       ; edx = 0
8355 00003ACC 8A570E                    <1>       mov   dl, [edi+LFBINFO.bpp]
8356                                    <1>       ; Note:
8357                                    <1>       ; Bits per pixel may be 8,16,24,32 for TRDOS 386 v2.
8358                                    <1>       ; (4 bits for pixel is not used for VESA modes here)
8359 00003ACF C0EA03                    <1>       shr   dl, 3 ; convert bits to byte
8360 00003AD2 F7E2                      <1>       mul   edx
8361                                    <1>       ; eax = screen/page/buffer size in bytes
8362 00003AD4 894706                    <1>       mov   [edi+LFBINFO.LFB_size], eax
8363                                    <1>       ; edx = 0
8364                                    <1>       ; clear reserved byte in LFBINFO structure/table
8365 00003AD7 88570F                    <1>       mov   [edi+LFBINFO.reserved], dl ; not necessary
8366 00003ADA C3                        <1>       retn
8367                                    <1>
8368                                    <1> ; * (TRDOS 386, INT 31h, VESA Video Bios functions)
8369                                    <1> ; * ------------------------------------------------------------
8370                                    <1> ; * Function 02h - Set VBE Mode
8371                                    <1> ; * ------------------------------------------------------------
8372                                    <1> ; * Input:
8373                                    <1> ; *         AX = 4F02h
8374                                    <1> ; *         BX = Desired Mode to set
8375                                    <1> ; * Output:
8376                                    <1> ; *         AX = VBE Return Status
8377                                    <1> ; *
8378                                    <1> ; *------------------------------------------------------------
8379                                    <1> ; *
8380                                    <1>
8381                                    <1> vbe_biosfn_set_mode:
8382                                    <1>       ; 12/12/2020
8383                                    <1>       ; 11/12/2020 (LFBINFO table for VESA VBE modes)
8384                                    <1>       ; 27/11/2020
8385                                    <1>       ; 25/11/2020
8386                                    <1>       ; 23/11/2020 (TRDOS 386 v2.0.3)
8387                                    <1>       ; (ref: vbe.c, 02/01/2020, vruppert)
8388                                    <1>       ;
8389                                    <1>       ; Input:
8390                                    <1>       ;     bx = video (bios) mode
8391                                    <1>       ;     ax = 4F02h
8392                                    <1>       ; Output:
8393                                    <1>       ;     ax = 004Fh (successful)
8394                                    <1>       ;     ah > 0 -> error
8395                                    <1>       ;
8396                                    <1>       ; Modified registers: esi
8397                                    <1>
8398                                    <1>       ; 27/11/2020
8399                                    <1>
8400                                    <1>       ;;push ds  ; *
8401                                    <1>       ;;push es  ; **
8402                                    <1>       ;;push ebp ; ***
8403                                    <1>       ;;push esi ; ****
8404                                    <1>
8405                                    <1>       ; 11/12/2020
8406 00003ADB 52                        <1>       push  edx ; *****
8407 00003ADC 51                        <1>       push  ecx ; ******
8408 00003ADD 53                        <1>       push  ebx ; *******
8409 00003ADE 57                        <1>       push  edi ; ********
8410                                    <1>
8411                                    <1>       ;xor  eax, eax
8412 00003ADF 80E7C1                    <1>       and   bh, 0C1h ; use bit 15, 14, 8 only (for bh)
8413 00003AE2 883D[1D120300]            <1>       mov   [vbe_mode_x], bh
8414 00003AE8 80E701                    <1>       and   bh, 1
8415 00003AEB 753C                      <1>       jnz   short vbe_sm_3  ; VESA VBE mode
8416                                    <1>
8417                                    <1>       ;;test bx, 4000h ; VBE_MODE_LINEAR_FRAME_BUFFER
8418                                    <1>       ;test  bh, 40h
8419                                    <1>       ;jz    short vbe_sm_0
8420                                    <1>       ;; lfb_flag
8421                                    <1>       ;mov   al, 40h      ; VBE_DISPI_LFB_ENABLED
8422                                    <1> vbe_sm_0:
8423                                    <1>       ; 27/11/2020
8424 00003AED B080                      <1>       mov   al, 80h
8425                                    <1>       ;test  bh, 80h ; VBE_MODE_PRESERVE_DISPLAY_MEMORY
8426                                    <1>       ;jnz   short vbe_sm_1 ; no_clear
8427                                    <1>       ;; clear
8428                                    <1>       ;sub   al, al ; 0
8429 00003AEF 8405[1D120300]            <1>       test  [vbe_mode_x], al ; 80h
8430 00003AF5 7402                      <1>       jz    short vbe_sm_1 ; clear display memory
8431                                    <1>       ; no_clear
8432 00003AF7 08C3                      <1>       or    bl, al ; VBE_MODE_PRESERVE_DISPLAY_MEMORY
8433                                    <1> vbe_sm_1:
8434                                    <1>       ; check non vesa mode
8435                                    <1>       ;;cmp  bx, 100h ; VBE_MODE_VESA_DEFINED
8436                                    <1>       ;;jna  short vbe_sm_2
8437                                    <1>       ;and   bh, 1
8438                                    <1>       ;jnz   short vbe_sm_3
8439                                    <1>
8440                                    <1>       ; BX <= 1FFh
8441                                    <1>
8442                                    <1>       ; 27/11/2020
8443                                    <1>       ;or    bl, al ; al = 80h if no_clear option is set
8444                                    <1>       ;             ; al = 0 if no_clear option is not set
8445                                    <1>
8446                                    <1>       ; 25/11/2020
8447                                    <1>       ; VBE DISPI will be disabled in 'biosfn_set_video_mode'
8448                                    <1>
8449                                    <1>       ;xor   al, al ; 0 ; VBE_DISPI_DISABLED
8450                                    <1>       ;call  dispi_set_enable
8451                                    <1>
8452                                    <1>       ; call the vgabios in order to set the video mode
8453                                    <1>       ; this allows for going back to textmode with a VBE call
8454                                    <1>       ; (some applications expect that to work)
8455                                    <1>
8456                                    <1>       ;and   bx, 0FFh
8457                                    <1>
8458                                    <1>       ; 27/11/2020
```

```
                            <1> biosfn_set_video_mode:
                            <1>       ; _call: call subroutine
                            <1>       ; 26/11/2020 (TRDOS 386 v2.0.3)
                            <1>       ; (ref: vgabios.c, 02/01/2020, vruppert)
                            <1>       ; Input:
                            <1>       ;     bl = VGA video (bios) mode
                            <1>       ; Output:
                            <1>       ;     cf = 1 -> error
                            <1>       ;     cf = 0 -> ok
                            <1>       ;
                            <1>       ; Modified registers: esi
                            <1>
                            <1>       ; 'dispi_set_enable(VBE_DISPI_DISABLED);'
                            <1>
                            <1>       ;mov   ax, 0 ; VBE_DISPI_DISABLED
00003AF9 31C0               <1>       xor    eax, eax ; 0
00003AFB E8A3040000         <1>       call   dispi_set_enable
                            <1>
00003B00 88D8               <1>       mov    al, bl
                            <1>       ;jmp   _set_mode ; (in 'biosfn_set_video_mode' sub)
00003B02 E861E0FFFF         <1>       call   _set_mode ; will return with cf=1 only if
                            <1>                        ; desired mode is not implemented
                            <1>       ; _retn: return from subroutine
00003B07 721A               <1>       jc     short vbe_sm_2 ; 25/11/2020
                            <1>
                            <1>       ; 26/11/2020
00003B09 31C0               <1>       xor    eax, eax
00003B0B A0[CA6F0000]       <1>       mov    al, [CRT_MODE]
                            <1>       ; 27/11/2020
00003B10 8A25[57960100]     <1>       mov    ah, [noclearmem] ; 80h or 0
                            <1>       ;and   ah 80h
00003B16 66A3[1E120300]     <1>       mov    [video_mode], ax ; bit 15 = no_clear flag
                            <1>                               ; bit 14 = 0 (not LFB model)
                            <1> vbe_sm_ret1:
                            <1>       ; 11/12/2020
                            <1>       ; (vbe_rmi_4 and vbe_rmi_6 jump here)
                            <1>       ; 27/11/2020
00003B1C B04F               <1>       mov    al, 4Fh ; Function call successful
                            <1>       ; eax = 004Fh
                            <1> vbe_sm_ret2:
                            <1>       ; 11/12/2020
00003B1E 5F                 <1>       pop    edi ; ********
00003B1F 5B                 <1>       pop    ebx ; *******
00003B20 59                 <1>       pop    ecx ; ******
00003B21 5A                 <1>       pop    edx ; *****
                            <1>
                            <1>       ;;pop  esi ; ****
                            <1>       ;;pop  ebp ; ***
                            <1>       ;;pop  es  ; **
                            <1>       ;;pop  ds  ; *
                            <1>
00003B22 C3                 <1>       retn
                            <1>
                            <1> vbe_sm_2:
                            <1>       ;mov   ax, 0100h ; Function is not supported
                            <1>       ; 27/11/2020
00003B23 31C0               <1>       xor    eax, eax
00003B25 B401               <1>       mov    ah, 01h
                            <1>       ; eax = 0100h
                            <1>       ;retn
00003B27 EBF5               <1>       jmp    short vbe_sm_ret2
                            <1>
                            <1> vbe_sm_3:
                            <1>       ; 12/12/2020
                            <1>       ; check current mode, if it is 03h
                            <1>       ; save page contents and cursor positions
00003B29 803D[CA6F0000]03   <1>       cmp    byte [CRT_MODE], 03h
00003B30 75BB               <1>       jne    short vbe_sm_0
00003B32 E8E6E2FFFF         <1>       call   save_mode3_multiscreen
                            <1>       ; set current mode to extended (SVGA) mode
                            <1>       ;mov   byte [CRT_MODE], 0FFh ; VESA VBE mode
                            <1> vbe_sm_4:
                            <1>       ; 27/11/2020
                            <1>       ; bx = mode (bit 0 to 8)
                            <1>
                            <1>       ; 25/11/2020
                            <1>
                            <1>       ; Alternative 2 (instead of 'Mode_info_find_mode')
                            <1>       ;push  edi
00003B37 E88F050000         <1>       call   set_mode_info_list ; (alternative 2)
                            <1>       ;pop   edi
                            <1>
                            <1>       ;mov   bx, [esi] ; mode
                            <1>
                            <1>       ; Alternative 1 (instead of 'set_mode_info_list')
                            <1>       ;call  mode_info_find_mode ; (alternative 1)
                            <1>
00003B3C 09F6               <1>       or     esi, esi
00003B3E 74E3               <1>       jz     short vbe_sm_2 ; VBE mode number is wrong
                            <1>                           ; or it is not supported
                            <1>
                            <1>       ; 11/12/2020
00003B40 668B1E             <1>       mov    bx, [esi] ; mode
                            <1>
                            <1>       ; 27/11/2020
00003B43 0A3D[1D120300]     <1>       or     bh, [vbe_mode_x]
                            <1>
                            <1>       ; save VESA VBE mode
00003B49 66891D[1E120300]   <1>       mov    [video_mode], bx
                            <1>              ; 27/11/2020
                            <1>              ; bit 0 to 8 = VESA VBE mode
                            <1>              ; bit 9 to 13 = 0 (bit 0 to 13 = mode)
                            <1>              ; bit 14 = Linear/Flat Frame Buffer flag
                            <1>              ; bit 15 = 'memory not cleared
                            <1>              ;         at last mode set' flag
```

```
8564                                    <1>
8565                                    <1>     ; first disable current mode
8566                                    <1>     ; (when switching between vesa modes)
8567                                    <1>     ; 'dispi_set_enable(VBE_DISPI_DISABLED);'
8568                                    <1>
8569                                    <1>     ;mov   ax, VBE_DISPI_DISABLED ; 0
8570 00003B50 29C0                      <1>     sub    eax, eax ; 0
8571                                    <1>
8572 00003B52 E84C040000                <1>     call   dispi_set_enable
8573                                    <1>
8574                                    <1>     ; 11/12/2020
8575 00003B57 8A461B                    <1>     mov    al, [esi+MODEINFO.BitsPerPixel]
8576                                    <1>     ; ah = 0
8577                                    <1>
8578                                    <1>     ;cmp   byte [esi+MODEINFO.BitsPerPixel], 8
8579 00003B5A 3C08                      <1>     cmp    al, 8
8580 00003B5C 750B                      <1>     jne    short vbe_sm_5
8581                                    <1>
8582                                    <1>     ; 11/12/2020
8583                                    <1>     ;push  edi
8584 00003B5E 50                        <1>     push   eax
8585                                    <1>     ; 'load_dac_palette(3);'
8586 00003B5F 56                        <1>     push   esi
8587 00003B60 B403                      <1>     mov    ah, 3  ; palette3, 256 colors
8588 00003B62 E828F1FFFF                <1>     call   load_dac_palette
8589 00003B67 5E                        <1>     pop    esi
8590                                    <1>     ; 11/12/2020
8591 00003B68 58                        <1>     pop    eax
8592                                    <1>     ;pop   edi
8593                                    <1> vbe_sm_5:
8594                                    <1>     ;'dispi_set_bpp(cur_info->info.BitsPerPixel);'
8595                                    <1>     ; 11/12/2020 (al = bits per pixel, ah = 0)
8596                                    <1>     ;xor   ah, ah
8597                                    <1>     ;mov   al, [esi+MODEINFO.BitsPerPixel]
8598 00003B69 E849040000                <1>     call   dispi_set_bpp
8599                                    <1>       ;'dispi_set_xres(cur_info->info.XResolution);'
8600 00003B6E 668B4614                  <1>     mov    ax, [esi+MODEINFO.XResolution]
8601 00003B72 E846040000                <1>     call   dispi_set_xres
8602                                    <1>       ;'dispi_set_yres(cur_info->info.YResolution);'
8603 00003B77 668B4616                  <1>     mov    ax, [esi+MODEINFO.YResolution]
8604 00003B7B E843040000                <1>     call   dispi_set_yres
8605                                    <1>
8606                                    <1>     ;'dispi_set_bank(0);'
8607                                    <1>     ;xor   ax, ax
8608 00003B80 31C0                      <1>     xor    eax, eax ; 0
8609 00003B82 E842040000                <1>     call   dispi_set_bank
8610                                    <1>       ;'dispi_set_enable(VBE_DISPI_ENABLED|no_clear|lfb_flag);'
8611                                    <1>     ;mov   ax, di
8612                                    <1>     ; ah = 0 ; 27/11/2020
8613 00003B87 A0[1D120300]              <1>     mov    al, [vbe_mode_x] ; restore VBE mode bit 14 & 15
8614 00003B8C 0C01                      <1>     or     al, 1 ; VBE_DISPI_ENABLED
8615 00003B8E E810040000                <1>     call   dispi_set_enable
8616                                    <1>
8617                                    <1>       ; 'vga_compat_setup();'
8618 00003B93 E846040000                <1>     call   vga_compat_setup
8619                                    <1>
8620                                    <1>     ; 11/12/2020
8621 00003B98 E802FFFFFF                <1>     call   set_lfbinfo_table
8622                                    <1>
8623                                    <1>     ; 26/11/2020
8624 00003B9D 31C0                      <1>     xor    eax, eax
8625 00003B9F FEC8                      <1>     dec    al
8626 00003BA1 A2[CA6F0000]              <1>     mov    [CRT_MODE], al ; 0FFh = VESA VBE mode sign
8627                                    <1>
8628                                    <1>     ; 27/11/2020
8629 00003BA6 E971FFFFFF                <1>     jmp    vbe_sm_ret1 ; Function call successful
8630                                    <1>
8631                                    <1>     ; 27/11/2020
8632                                    <1>     ;mov   al, 4Fh
8633                                    <1>     ;      ; eax = 004Fh = Function call successful
8634                                    <1>     ;jmp   short vbe_sm_ret2
8635                                    <1>
8636                                    <1> ; * (TRDOS 386, INT 31h, VESA Video Bios functions)
8637                                    <1> ; * ------------------------------------------------------------
8638                                    <1> ; * Function 03h - Return Current VBE Mode
8639                                    <1> ; * ------------------------------------------------------------
8640                                    <1> ; * Input:
8641                                    <1> ; *          AX = 4F03h
8642                                    <1> ; * Output:
8643                                    <1> ; *          AX = VBE Return Status
8644                                    <1> ; *          BX = Current VBE Mode
8645                                    <1> ; *
8646                                    <1> ; *------------------------------------------------------------
8647                                    <1> ; *
8648                                    <1>
8649                                    <1> vbe_biosfn_return_current_mode:
8650                                    <1>       ; 11/12/2020
8651                                    <1>       ; 27/11/2020 (TRDOS 386 v2.0.3)
8652                                    <1>       ; (ref: vbe.c, 02/01/2020, vruppert)
8653                                    <1>       ;
8654                                    <1>       ; Input:
8655                                    <1>       ;      none
8656                                    <1>       ; Output:
8657                                    <1>       ;      ax = 004Fh (successful)
8658                                    <1>       ;      ah > 0 -> error
8659                                    <1>       ;      bx = current video (bios) mode (if ah = 0)
8660                                    <1>       ;
8661                                    <1>       ; Modified registers: eax, ebx
8662                                    <1>
8663                                    <1>       ; 27/11/2020
8664                                    <1>
8665                                    <1>       ;;push ds  ; *
8666                                    <1>       ;;push es  ; **
8667                                    <1>       ;;push ebp ; ***
8668                                    <1>       ;;push esi ; ****
```

```
                                <1>
                                <1>       ;push  edx ; *****
                                <1>
                                <1>       ; (vbe.c)
                                <1>       ;call  dispi_get_enable
                                <1>       ;      ; ax = vbe display interface status
                                <1>       ;and   al, 1 ; VBE_DISPI_ENABLED
                                <1>       ;jnz   short vbe_gm_1  ; VBE graphics mode
                                <1>
00003BAB A0[CA6F0000]           <1>       mov    al, [CRT_MODE] ; current cga/vga mode
00003BB0 3CFF                   <1>       cmp    al, 0FFh ; VBE extension signature
00003BB2 720E                   <1>       jb     short vbe_gm_1 ; get CGA/VGA mode
                                <1>
                                <1>       ; get VBE mode
                                <1> vbe_gm_0:
00003BB4 66A1[1E120300]         <1>       mov    ax, [video_mode]
                                <1>             ; BX bits:
                                <1>             ; bit 0 to 8 = VESA VBE video mode
                                <1>             ; bit 9 to 13 = 0
                                <1>             ; bit 14 = last mode set LFB option
                                <1>             ;         1 - linear/flat frame buffer
                                <1>             ;         0 - windowed frame buffer
                                <1>             ; bit 15 = last mode set no_clear option
                                <1>             ;         0 - video memory cleared
                                <1>             ;         1 - video memory not cleared
                                <1>
                                <1> vbe_gm_return:
                                <1>       ;pop   edx ; ******
00003BBA 0FB7D8                 <1>       movzx  ebx, ax
                                <1> ;vbe_srs_retn:
00003BBD 31C0                   <1>       xor    eax, eax ; 0
00003BBF B04F                   <1>       mov    al, 4Fh ; ax = 004Fh (successful)
00003BC1 C3                     <1>       retn
                                <1>
                                <1> vbe_gm_1:
                                <1>       ; legacy (old, standard) CGA/VGA bios video mode
00003BC2 8A25[57960100]         <1>       mov    ah, [noclearmem] ; 80h or 0
                                <1>             ; BX bits:
                                <1>             ; bit 0 to 7 = video mode
                                <1>             ; bit 8 to 13 = 0
                                <1>             ; bit 14 = 0 (not LFB mode) CGA/VGA
                                <1>             ; bit 15 = 1 if [noclearmem] = 80h
                                <1>             ;         0 if [noclearmem] = 0
00003BC8 EBF0                   <1>       jmp    short vbe_gm_return
                                <1>
                                <1> ; * (TRDOS 386, INT 31h, VESA Video Bios functions)
                                <1> ; * ------------------------------------------------------------
                                <1> ; * Function 04h - Save/Restore State
                                <1> ; * ------------------------------------------------------------
                                <1> ; * Input:
                                <1> ; *         AX = 4F04h
                                <1> ; *             DL = 00h Return Save/Restore State buff size
                                <1> ; *                  01h Save State
                                <1> ; *                  02h Restore State
                                <1> ; *             CX = Requested states
                                <1> ; *               bit 0 - controller hardware state
                                <1> ; *               bit 1 - BIOS data state
                                <1> ; *               bit 2 - DAC state
                                <1> ; *               bit 3 - register state
                                <1> ; *    (ES:BX) EBX = Pointer to buffer (if DL <> 00h)
                                <1> ; * Output:
                                <1> ; *         AX = VBE Return Status
                                <1> ; *         BX = Number of 64-byte blocks
                                <1> ; *              to hold the state buffer (if DL=00h)
                                <1> ; *
                                <1> ; *------------------------------------------------------------
                                <1> ; *
                                <1>
                                <1> vbe_biosfn_save_restore_state:
                                <1>       ; 23/01/2021
                                <1>       ; 16/01/2021
                                <1>       ; 14/01/2021
                                <1>       ; 13/01/2021
                                <1>       ; 12/01/2021
                                <1>       ; 11/01/2021 (TRDOS 386 v2.0.3)
                                <1>       ; (ref: vbe.c, 02/01/2020, vruppert)
                                <1>       ;
                                <1>       ; Input:
                                <1>       ;      dl = sub function
                                <1>       ;      cl = requested state
                                <1>       ;      ebx = pointer to buffer (if dl<>00h)
                                <1>       ; Output:
                                <1>       ;      ax = 004Fh (successful)
                                <1>       ;      ah > 0 -> error
                                <1>       ;      bx = Number of 64-byte blocks
                                <1>       ;           to hold the state buffer (if DL=00h)
                                <1>
                                <1>       ; Modified registers: eax, ebx, edi
                                <1>
                                <1>       ; 14/01/2021
00003BCA 09DB                   <1>       or     ebx, ebx ; user's buffer address
00003BCC 750A                   <1>       jnz    short _vbe_biosfn_save_restore_state
                                <1>
00003BCE 20D2                   <1>       and    dl, dl
00003BD0 7406                   <1>       jz     short _vbe_biosfn_save_restore_state
                                <1>
                                <1>       ; function failed
                                <1>       ;mov   eax, 0100h
                                <1>       ;xor   eax, eax
                                <1>       ;inc   ah  ; eax = 0100h
                                <1>       ; 16/01/2021
00003BD2 B84F010000             <1>       mov    eax, 014Fh
00003BD7 C3                     <1>       retn
                                <1>
                                <1> _vbe_biosfn_save_restore_state:
```

```
                                <1>       ; 23/01/2021
                                <1>       ; 14/01/2021
                                <1>       ; ebx = 0 if the caller is kernel ('sysvideo')
                                <1>
                                <1>       ; 13/01/2021
00003BD8 57                     <1>       push   edi
00003BD9 52                     <1>       push   edx
00003BDA 51                     <1>       push   ecx
                                <1>
                                <1>       ; 23/01/2021
                                <1>       ; 12/01/2021
00003BDB 80FA02                 <1>       cmp    dl, 2
00003BDE 7757                   <1>       ja     short vbe_srs_7 ; 23/01/2021
                                <1>                      ; invalid sub function
00003BE0 83F90F                 <1>       cmp    ecx, 0Fh
00003BE3 7752                   <1>       ja     short vbe_srs_7 ; invalid !
                                <1>
00003BE5 20D2                   <1>       and    dl, dl
00003BE7 7515                   <1>       jnz    short vbe_srs_4
                                <1>
                                <1>       ; DL = 0
                                <1>       ; Return Save/Restore State buffer size
                                <1>
                                <1>       ;mov   ebx, ecx
                                <1>       ;shl   bl, 1
                                <1>       ;mov   bx, [ebx+vbestatebufsize]
00003BE9 E881000000             <1>       call   vbe_srs_gbs
                                <1>
                                <1> ;     ; 11/01/2021
                                <1> ;     test   cl, 8
                                <1> ;     jz     short vbe_srs_3
                                <1> ;     ; vbe_biosfn_read_video_state_size();
                                <1> ;     ; return 9 * 2;
                                <1> ;     mov    bl, 18 ; register state size
                                <1> ;vbe_srs_0:
                                <1> ;     test   cl, 1
                                <1> ;     jz     short vbe_srs_1
                                <1> ;     ; size += 0x46;
                                <1> ;     add    bl, 70 ; controller state size
                                <1> ;vbe_srs_1:
                                <1> ;     test   cl, 2
                                <1> ;     jz     short vbe_srs_2
                                <1> ;     ; size += (5 + 8 + 5) * 2 + 6;
                                <1> ;     ;add   bl, 42 ; BIOS data state size ; Bochs/Plex86
                                <1> ;     ; 12/01/2021
                                <1> ;     add    bl, 40 ; TRDOS 386 v2 VBIOS data state size
                                <1> ;vbe_srs_2:
                                <1> ;     test   cl, 4
                                <1> ;     jz     short vbe_srs_3
                                <1> ;     ; size += 3 + 256 * 3 + 1;
                                <1> ;     add    bx, 772 ; DAC state size
                                <1>
                                <1> vbe_srs_3:
00003BEE 6683C33F               <1>       add    bx, 63
00003BF2 66C1EB06               <1>       shr    bx, 6 ; / 64
                                <1>
                                <1> vbe_srs_retn:
00003BF6 31C0                   <1>       xor    eax, eax ; 0
                                <1> vbe_srs_0:  ; 16/01/2021
00003BF8 B04F                   <1>       mov    al, 4Fh ; ax = 004Fh (successful)
                                <1> ;vbe_srs_0:
                                <1>       ; 13/01/2021
00003BFA 59                     <1>       pop    ecx
00003BFB 5A                     <1>       pop    edx
00003BFC 5F                     <1>       pop    edi
                                <1>
00003BFD C3                     <1>       retn
                                <1>
                                <1>       ; 23/01/2021
                                <1> ;vbe_srs_10:
                                <1>       ;; 14/01/2021
                                <1>       ; return to 'sysvideo'
                                <1>       ;mov   ebx, ecx ; transfer count
                                <1>       ;      ; (byte count for saving current video state)
                                <1>       ;jmp   short vbe_srs_retn
                                <1>
                                <1> vbe_srs_4:
                                <1>       ; 23/01/2021
00003BFE 80E10F                 <1>       and    cl, 0Fh ; 8, 4, 2, 1
00003C01 7434                   <1>       jz     short vbe_srs_7 ; cx = 0 -> invalid !
                                <1>
00003C03 BF00760900             <1>       mov    edi, VBE3SAVERESTOREBLOCK
                                <1>
00003C08 80FA01                 <1>       cmp    dl, 1
00003C0B 7730                   <1>       ja     short vbe_srs_8
                                <1>
                                <1>       ; save video state
                                <1>
00003C0D F6C107                 <1>       test   cl, 07h ; 4, 2, 1
00003C10 740A                   <1>       jz     short vbe_srs_5  ; vbe dispi regs state
                                <1>
00003C12 E884000000             <1>       call   biosfn_save_video_state
                                <1>       ; edi = current position
                                <1>       ;      in VBE3SAVERESTOREBLOCK
                                <1>       ;      (VGA save_state offset)
                                <1>       ; modified regs: edi, eax, edx, ch
00003C17 F6C108                 <1>       test   cl, 8
00003C1A 7405                   <1>       jz     short vbe_srs_6
                                <1> vbe_srs_5:
00003C1C E8AC010000             <1>       call   vbe_biosfn_save_video_state
                                <1>       ; edi = end position
                                <1>       ;      in VBE3SAVERESTOREBLOCK
                                <1>       ;      (VGA save_state offset)
                                <1>       ; modified regs: edi, eax, edx, ch
                                <1> vbe_srs_6:
```

```
8879                                    <1>       ; 23/01/2021
8880 00003C21 21DB                      <1>       and    ebx, ebx
8881 00003C23 74D1                      <1>       jz     short vbe_srs_retn ; the caller is kernel
8882                                    <1>
8883 00003C25 BE00760900                <1>       mov    esi, VBE3SAVERESTOREBLOCK
8884 00003C2A 29F7                      <1>       sub    edi, esi
8885 00003C2C 89F9                      <1>       mov    ecx, edi ; transfer count in bytes
8886                                    <1>
8887                                    <1>       ;; 14/01/2021
8888                                    <1>       ;and   ebx, ebx
8889                                    <1>       ;jz    short vbe_srs_10 ; the caller is kernel
8890                                    <1>
8891 00003C2E 89DF                      <1>       mov    edi, ebx ; user's buffer address
8892 00003C30 E8BDDE0000                <1>       call   transfer_to_user_buffer
8893 00003C35 73BF                      <1>       jnc    short vbe_srs_retn
8894                                    <1> vbe_srs_7:
8895                                    <1>       ; // function failed
8896                                    <1>       ;mov   eax, 0100h
8897 00003C37 31C0                      <1>       xor    eax, eax
8898 00003C39 FEC4                      <1>       inc    ah  ; eax = 0100h
8899                                    <1>       ; 16/01/2021
8900                                    <1>       ; ax = 0014Fh
8901                                    <1>       ;retn
8902                                    <1>       ; 13/01/2021
8903 00003C3B EBBB                      <1>       jmp    short vbe_srs_0
8904                                    <1> vbe_srs_8:
8905                                    <1>       ;cmp   dl, 2
8906                                    <1>       ;jne   short vbe_srs_7
8907                                    <1>       ;            ; invalid sub function
8908                                    <1>
8909                                    <1>       ; 14/01/2021
8910 00003C3D 09DB                      <1>       or     ebx, ebx ; user's buffer address
8911                                    <1>       ;jnz   short vbe_srs_11
8912                                    <1>
8913                                    <1>       ; the caller is kernel ('sysvideo')
8914                                    <1>       ;jmp   short vbe_srs_12
8915                                    <1>       ; 23/01/2021
8916 00003C3F 7414                      <1>       jz     short vbe_srs_12 ; 'sysvideo' call
8917                                    <1> vbe_srs_11:
8918 00003C41 89DE                      <1>       mov    esi, ebx ; user's buffer address
8919                                    <1>       ; 23/01/2021
8920                                    <1>       ;push  ebx
8921                                    <1>
8922 00003C43 E827000000                <1>       call   vbe_srs_gbs
8923                                    <1>
8924                                    <1>       ; restore video state
8925                                    <1>
8926                                    <1>       ;mov   edi, VBE3SAVERESTOREBLOCK
8927 00003C48 51                        <1>       push   ecx
8928 00003C49 89D9                      <1>       mov    ecx, ebx ; transfer count in bytes
8929 00003C4B E8ECDE0000                <1>       call   transfer_from_user_buffer
8930 00003C50 59                        <1>       pop    ecx
8931                                    <1>       ; 23/01/2021
8932                                    <1>       ;pop   ebx
8933 00003C51 89F3                      <1>       mov    ebx, esi
8934 00003C53 72E2                      <1>       jc     short vbe_srs_7
8935                                    <1>
8936                                    <1> vbe_srs_12:
8937                                    <1>       ;mov   esi, VBE3SAVERESTOREBLOCK
8938 00003C55 89FE                      <1>       mov    esi, edi
8939                                    <1>
8940 00003C57 F6C107                    <1>       test   cl, 07h ; 4, 2, 1
8941 00003C5A 740C                      <1>       jz     short vbe_srs_9     ; vbe dispi regs state
8942                                    <1>
8943 00003C5C E8A8010000                <1>       call   biosfn_restore_video_state
8944 00003C61 72D4                      <1>       jc     short vbe_srs_7 ; invalid buffer content !
8945                                    <1>       ; esi = current position
8946                                    <1>       ;      in VBE3SAVERESTOREBLOCK
8947                                    <1>       ;      (VGA save_state offset)
8948                                    <1>       ; modified regs: esi, eax, edx, ch
8949 00003C63 F6C108                    <1>       test   cl, 8
8950                                    <1>       ;jz    short vbe_srs_10
8951                                    <1>       ; 23/01/2020
8952 00003C66 EB8E                      <1>       jmp    short vbe_srs_retn
8953                                    <1> vbe_srs_9:
8954 00003C68 E8F8020000                <1>       call   vbe_biosfn_restore_video_state
8955                                    <1>
8956                                    <1>       ; modified regs: esi, eax, edx, ch
8957                                    <1>
8958 00003C6D EB87                      <1>       jmp    short vbe_srs_retn
8959                                    <1>
8960                                    <1> ;vbe_srs_10:
8961                                    <1> ;     ; successful
8962                                    <1> ;     xor    eax, eax ; 0
8963                                    <1> ;     mov    al, 4Fh ; ax = 004Fh (successful)
8964                                    <1> ;     retn
8965                                    <1>
8966                                    <1> vbe_srs_gbs:
8967                                    <1>       ; return buffer size according to flags
8968 00003C6F 89CB                      <1>       mov    ebx, ecx ; options/flags
8969 00003C71 D0E3                      <1>       shl    bl, 1
8970 00003C73 668B9B[7B3C0000]          <1>       mov    bx, [ebx+vbestatebufsize]
8971 00003C7A C3                        <1>       retn
8972                                    <1>
8973                                    <1> vbestatebufsize:
8974                                    <1>       ; ---------------------------------------
8975                                    <1>       ; CL = 0  1   2    3    4    5    6    7
8976                                    <1>       ; ---------------------------------------
8977 00003C7B 0000460028006E0004-       <1>       dw     0, 70, 40, 110, 772, 842, 812, 882
8977 00003C84 034A032C037203            <1>
8978                                    <1>       ; ---------------------------------------
8979                                    <1>       ; CL = 8   9   10  11   12   13   14   15
8980                                    <1>       ; ---------------------------------------
8981 00003C8B 120058003A00800016-       <1>       dw     18, 88, 58, 128, 790, 860, 830, 900
8981 00003C94 035C033E038403            <1>
```

```
                                        <1>
                                        <1> ; 11/01/2021
                                        <1> VGAREG_ACTL_ADDRESS       equ 3C0h
                                        <1> VGAREG_ACTL_WRITE_DATA    equ 3C0h
                                        <1> VGAREG_ACTL_READ_DATA     equ 3C1h
                                        <1>
                                        <1> VGAREG_INPUT_STATUS       equ 3C2h
                                        <1> VGAREG_WRITE_MISC_OUTPUT equ 3C2h
                                        <1> VGAREG_VIDEO_ENABLE       equ 3C3h
                                        <1> VGAREG_SEQU_ADDRESS       equ 3C4h
                                        <1> VGAREG_SEQU_DATA   equ 3C5h
                                        <1>
                                        <1> VGAREG_PEL_MASK           equ 3C6h
                                        <1> VGAREG_DAC_STATE   equ 3C7h
                                        <1> VGAREG_DAC_READ_ADDRESS   equ 3C7h
                                        <1> VGAREG_DAC_WRITE_ADDRESS equ 3C8h
                                        <1> VGAREG_DAC_DATA           equ 3C9h
                                        <1>
                                        <1> VGAREG_READ_FEATURE_CTL   equ 3CAh
                                        <1> VGAREG_READ_MISC_OUTPUT   equ 3CCh
                                        <1>
                                        <1> VGAREG_GRDC_ADDRESS       equ 3CEh
                                        <1> VGAREG_GRDC_DATA   equ 3CFh
                                        <1>
                                        <1> ;VGAREG_MDA_CRTC_ADDRESS equ 3B4h
                                        <1> ;VGAREG_MDA_CRTC_DATA     equ 3B5h
                                        <1> VGAREG_VGA_CRTC_ADDRESS   equ 3D4h
                                        <1> VGAREG_VGA_CRTC_DATA      equ 3D5h
                                        <1>
                                        <1> ;VGAREG_MDA_WRITE_FEATURE_CTL equ 3BAh
                                        <1> VGAREG_VGA_WRITE_FEATURE_CTL equ 3DAh
                                        <1> VGAREG_ACTL_RESET  equ 3DAh
                                        <1>
                                        <1> ;VGAREG_MDA_MODECTL       equ 3B8h
                                        <1> VGAREG_CGA_MODECTL equ 3D8h
                                        <1> VGAREG_CGA_PALETTE equ 3D9h
                                        <1>
                                        <1> biosfn_save_video_state:
                                        <1>       ; 22/01/2021
                                        <1>       ; 12/01/2021
                                        <1>       ; 11/01/2021 (TRDOS 386 v2.0.3)
                                        <1>       ; (vgabios.c)
                                        <1>
                                        <1>       ; modified registers: eax, edx, edi, ch
                                        <1>
                                        <1>       ;mov  edi, VBE3SAVERESTOREBLOCK
                                        <1>
                                        <1>       ; input: edi = state buffer address
                                        <1>
00003C9B F6C101                         <1>       test  cl, 1
00003C9E 0F8485000000                   <1>       jz    bfn_svs_4
                                        <1>
00003CA4 66BAC403                       <1>       mov   dx, VGAREG_SEQU_ADDRESS ; 3C7h
00003CA8 EC                             <1>       in    al, dx
00003CA9 AA                             <1>       stosb
                                        <1>       ;mov  dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
00003CAA B2D4                           <1>       mov   dl, 0D4h
00003CAC EC                             <1>       in    al, dx
00003CAD AA                             <1>       stosb
                                        <1>       ;mov  dx, VGAREG_GRDC_ADDRESS ; 3CEh
00003CAE B2CE                           <1>         mov dl, 0CEh
00003CB0 EC                             <1>       in    al, dx
00003CB1 AA                             <1>       stosb
                                        <1>       ;mov  dx, VGAREG_ACTL_RESET ; 3DAh
00003CB2 B2DA                           <1>       mov   dl, 0DAh
00003CB4 EC                             <1>       in    al, dx
                                        <1>       ;mov  dx, VGAREG_ACTL_ADDRESS ; 3C0h
00003CB5 B2C0                           <1>       mov   dl, 0C0h
00003CB7 EC                             <1>       in    al, dx
00003CB8 AA                             <1>       stosb
00003CB9 88C4                           <1>       mov   ah, al ; ar_index
                                        <1>       ;mov  dx, VGAREG_READ_FEATURE_CTL ; 3CAh
00003CBB B2CA                           <1>       mov   dl, 0CAh
00003CBD EC                             <1>       in    al, dx
00003CBE AA                             <1>       stosb
                                        <1>       ; (5 bytes are writen above)
                                        <1>
                                        <1>       ; for(i=1;i<=4;i++){
00003CBF B001                           <1>       mov   al, 1
                                        <1>       ;;mov  dx, VGAREG_SEQU_ADDRESS ; 3C4h
                                        <1>       ;mov  dl, 0C4h
00003CC1 B504                           <1>       mov   ch, 4
                                        <1> bfn_svs_0:
                                        <1>       ; outb(VGAREG_SEQU_ADDRESS, i);
                                        <1>       ;mov  dx, VGAREG_SEQU_ADDRESS ; 3C4h
00003CC3 B2C4                           <1>       mov   dl, 0C4h
00003CC5 EE                             <1>       out   dx, al
                                        <1>       ;mov  dx, VGAREG_SEQU_DATA  ; 3C5h
00003CC6 FEC2                           <1>       inc   dl  ; dx = 3C5h
                                        <1>       ; inb(VGAREG_SEQU_DATA)
00003CC8 50                             <1>       push  eax
00003CC9 EC                             <1>       in    al, dx
00003CCA AA                             <1>       stosb ; (4 bytes in loop)
00003CCB 58                             <1>       pop   eax
                                        <1>       ;mov  dx, VGAREG_SEQU_ADDRESS ; 3C4h
                                        <1>       ;dec  dl
00003CCC FEC0                           <1>       inc   al  ; i++
00003CCE FECD                           <1>       dec   ch
00003CD0 75F1                           <1>       jnz   short bfn_svs_0
                                        <1>
                                        <1>       ; outb(VGAREG_SEQU_ADDRESS, 0);
00003CD2 28C0                           <1>       sub   al, al ; 0
00003CD4 EE                             <1>       out   dx, al
                                        <1>       ; inb(VGAREG_SEQU_DATA)
                                        <1>       ;mov  dx, VGAREG_SEQU_DATA ; 3C5h
```

```
9087 00003CD5 FEC2              <1>       inc    dl  ; dx = 3C5h
9088 00003CD7 EC                <1>       in     al, dx
9089 00003CD8 AA                <1>       stosb  ; (+1 byte)
9090                            <1>
9091                            <1>         ; for(i=0;i<=0x18;i++) {
9092 00003CD9 28C0              <1>       sub    al, al ; 0
9093                            <1>       ;;mov  dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
9094                            <1>       ;mov   dl, 0D4h
9095 00003CDB B519              <1>       mov    ch, 25
9096                            <1> bfn_svs_1:
9097                            <1>         ; outb(crtc_addr,i);
9098                            <1>       ;mov   dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
9099 00003CDD B2D4              <1>       mov    dl, 0D4h
9100 00003CDF EE                <1>       out    dx, al
9101                            <1>       ;mov   dx, VGAREG_VGA_CRTC_DATA ; 3D5h
9102 00003CE0 FEC2              <1>       inc    dl  ; dx = 3D5h
9103                            <1>       ; inb(crtc_addr+1)
9104 00003CE2 50                <1>       push   eax
9105 00003CE3 EC                <1>       in     al, dx
9106 00003CE4 AA                <1>       stosb  ; (25 bytes in loop)
9107 00003CE5 58                <1>       pop    eax
9108                            <1>       ;mov   dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
9109                            <1>       ;dec   dl
9110 00003CE6 FEC0              <1>       inc    al  ; i++
9111 00003CE8 FECD              <1>       dec    ch
9112 00003CEA 75F1              <1>       jnz    short bfn_svs_1
9113                            <1>
9114 00003CEC 80E420            <1>       and    ah, 20h  ; (ar_index & 0x20)
9115                            <1>         ; for(i=0;i<=0x13;i++) {
9116 00003CEF 28C0              <1>       sub    al, al ; 0
9117 00003CF1 B514              <1>       mov    ch, 20
9118                            <1> bfn_svs_2:
9119                            <1>       ; inb(VGAREG_ACTL_RESET);
9120                            <1>       ;mov   dx, VGAREG_ACTL_RESET ; 3DAh
9121 00003CF3 B2DA              <1>       mov    dl, 0DAh
9122 00003CF5 50                <1>       push   eax
9123 00003CF6 EC                <1>       in     al, dx
9124 00003CF7 8A0424            <1>       mov    al, [esp]
9125                            <1>       ; outb(VGAREG_ACTL_ADDRESS, i | (ar_index & 0x20));
9126 00003CFA 08E0              <1>       or     al, ah
9127                            <1>       ;mov   dx, VGAREG_ACTL_ADDRESS ; 3C0h
9128 00003CFC B2C0              <1>       mov    dl, 0C0h
9129 00003CFE EE                <1>       out    dx, al
9130                            <1>       ;mov   dx, VGAREG_ACTL_READ_DATA ; 3C1h
9131                            <1>       ;mov   dl, 0C1h
9132 00003CFF FEC2              <1>       inc    dl
9133 00003D01 EC                <1>       in     al, dx
9134 00003D02 AA                <1>       stosb  ; (20 bytes in loop)
9135 00003D03 58                <1>       pop    eax
9136 00003D04 FEC0              <1>       inc    al  ; i++
9137 00003D06 FECD              <1>       dec    ch
9138 00003D08 75E9              <1>       jnz    short bfn_svs_2
9139                            <1>
9140                            <1>       ; inb(VGAREG_ACTL_RESET);
9141                            <1>       ;mov   dx, VGAREG_ACTL_RESET ; 3DAh
9142 00003D0A B2DA              <1>       mov    dl, 0DAh
9143 00003D0C EC                <1>       in     al, dx
9144                            <1>
9145                            <1>         ; for(i=0;i<=8;i++) {
9146 00003D0D 28C0              <1>       sub    al, al ; 0
9147                            <1>       ;;mov  dx, VGAREG_GRDC_ADDRESS ; 3CEh
9148                            <1>       ;mov   dl, 0CEh
9149 00003D0F B509              <1>       mov    ch, 9
9150                            <1> bfn_svs_3:
9151                            <1>       ; outb(VGAREG_GRDC_ADDRESS,i)
9152                            <1>       ;mov   dx, VGAREG_GRDC_ADDRESS ; 3CEh
9153 00003D11 B2CE              <1>       mov    dl, 0CEh
9154 00003D13 EE                <1>       out    dx, al
9155                            <1>       ; inb(VGAREG_ACTL_READ_DATA)
9156 00003D14 50                <1>       push   eax
9157                            <1>       ;mov   dx, VGAREG_GRDC_DATA ; 3CFh
9158                            <1>       ;mov   dl, 0CFh
9159 00003D15 FEC2              <1>       inc    dl
9160 00003D17 EC                <1>       in     al, dx
9161 00003D18 AA                <1>       stosb  ; (9 bytes in loop)
9162 00003D19 58                <1>       pop    eax
9163                            <1>       ;dec   dl
9164 00003D1A FEC0              <1>       inc    al  ; i++
9165 00003D1C FECD              <1>       dec    ch
9166 00003D1E 75F1              <1>       jnz    short bfn_svs_3
9167                            <1>
9168                            <1>       ; write_word(ES, BX, crtc_addr); BX+= 2;
9169                            <1>       ; (offset 64)
9170 00003D20 66B8D403          <1>       mov    ax, 3D4h ; VGAREG_VGA_CRTC_ADDRESS
9171 00003D24 66AB              <1>       stosw  ; (2 bytes (1 word))
9172                            <1>
9173                            <1>         ; /* XXX: read plane latches */
9174 00003D26 31C0              <1>       xor    eax, eax  ; 0
9175 00003D28 AB                <1>              stosd  ; (4 bytes)
9176                            <1>
9177                            <1>       ; (total 70 bytes are written above as controller hardware state)
9178                            <1>
9179                            <1> bfn_svs_4:
9180                            <1>       ; 12/01/2021 (TRDOS 386 v2.0.3)
9181 00003D29 F6C102            <1>       test   cl, 2
9182 00003D2C 7476              <1>       jz     short bfn_svs_6
9183                            <1>
9184                            <1>       ; VIDEO BIOS DATA
9185                            <1>       ; !!! this data is valid for TRDOS 386 v2 kernel only !!!
9186                            <1>       ; (this is not same with BOCHS/PLEX86 video bios, BIOS data)
9187                            <1>
9188                            <1>       ; if (CX & 2) {
9189                            <1>         ;write_byte(ES, BX, read_byte(BIOSMEM_SEG,BIOSMEM_CURRENT_MODE)); BX++;
9190                            <1>         ;write_word(ES, BX, read_word(BIOSMEM_SEG,BIOSMEM_NB_COLS)); BX += 2;
9191                            <1>         ;write_word(ES, BX, read_word(BIOSMEM_SEG,BIOSMEM_PAGE_SIZE)); BX += 2;
```

```
                                <1>         ;write_word(ES, BX, read_word(BIOSMEM_SEG,BIOSMEM_CRTC_ADDRESS)); BX += 2;
                                <1>         ;write_byte(ES, BX, read_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS)); BX++;
                                <1>         ;write_word(ES, BX, read_word(BIOSMEM_SEG,BIOSMEM_CHAR_HEIGHT)); BX += 2;
                                <1>         ;write_byte(ES, BX, read_byte(BIOSMEM_SEG,BIOSMEM_VIDEO_CTL)); BX++;
                                <1>         ;write_byte(ES, BX, read_byte(BIOSMEM_SEG,BIOSMEM_SWITCHES)); BX++;
                                <1>         ;write_byte(ES, BX, read_byte(BIOSMEM_SEG,BIOSMEM_MODESET_CTL)); BX++;
                                <1>         ;write_word(ES, BX, read_word(BIOSMEM_SEG,BIOSMEM_CURSOR_TYPE)); BX += 2;
                                <1>         ;for(i=0;i<8;i++) {
                                <1>         ;   write_word(ES, BX, read_word(BIOSMEM_SEG, BIOSMEM_CURSOR_POS+2*i));
                                <1>         ;   BX += 2;
                                <1>         ;}
                                <1>         ;write_word(ES, BX, read_word(BIOSMEM_SEG,BIOSMEM_CURRENT_START)); BX += 2;
                                <1>         ;write_byte(ES, BX, read_byte(BIOSMEM_SEG,BIOSMEM_CURRENT_PAGE)); BX++;
                                <1>         ;/* current font */
                                <1>         ;write_word(ES, BX, read_word(0, 0x1f * 4)); BX += 2;
                                <1>         ;write_word(ES, BX, read_word(0, 0x1f * 4 + 2)); BX += 2;
                                <1>         ;write_word(ES, BX, read_word(0, 0x43 * 4)); BX += 2;
                                <1>         ;write_word(ES, BX, read_word(0, 0x43 * 4 + 2)); BX += 2;
                                <1>
                                <1>       ; !!! save TRDOS 386 v2 kernel spesific video bios data !!!
                                <1>       ; (which is/are used by 'SET_MODE' function and/or it's sub functions)
                                <1>
00003D2E 66B8D403               <1>       mov   ax, 3D4h ; CRTC_ADDR, always 3D4h (color VGA) for TRDOS 386 v2
00003D32 66AB                   <1>       stosw
00003D34 A0[CA6F0000]           <1>       mov   al, [CRT_MODE] ; Current video mode (0FFh for VESA VBE modes)
00003D39 AA                     <1>       stosb
00003D3A A0[CB6F0000]           <1>       mov   al, [CRT_MODE_SET] ; 29h for mode 03h ; TRDOS 386 feature only !
00003D3F AA                     <1>       stosb
00003D40 66A1[1E120300]         <1>       mov   ax, [video_mode] ; Current VESA VBE (SVGA, extended VGA) mode
00003D46 66AB                   <1>       stosw                  ; (valid if [CRT_MODE] = 0FFh)
00003D48 66A1[58960100]         <1>       mov   ax, [CRT_LEN] ; page size (in bytes)
00003D4E 66AB                   <1>       stosw
00003D50 66A1[DC890100]         <1>       mov   ax, [CRT_START] ; video page start offset
00003D56 66AB                   <1>       stosw
00003D58 A0[CC6F0000]           <1>       mov   al, [CRT_COLS] ; nbcols, characters per row
00003D5D AA                     <1>       stosb
00003D5E A0[D26F0000]           <1>       mov   al, [VGA_ROWS] ; nbrows, (character) rows per page (not rows-1)
00003D63 AA                     <1>       stosb
00003D64 A0[CE6F0000]           <1>       mov   al, [CHAR_HEIGHT] ; character font height (8 or 16 or 14)
00003D69 AA                     <1>       stosb
00003D6A A0[CF6F0000]           <1>       mov   al, [VGA_VIDEO_CTL] ; ROM BIOS DATA AREA Offset 87h
00003D6F AA                     <1>       stosb
00003D70 A0[D06F0000]           <1>       mov   al, [VGA_SWITCHES] ; feature bit switches
00003D75 AA                     <1>       stosb
00003D76 A0[D16F0000]           <1>       mov   al, [VGA_MODESET_CTL] ; basic mode set options
00003D7B AA                     <1>       stosb
                                <1>       ; followings are only used by TRDOS 386 v2 (IBM PC/AT ROMBIOS) code
                                <1>       ; (bochs/plex86 does not use and return those)
00003D7C A0[CD6F0000]           <1>       mov   al, [CRT_PALETTE] ; current color palette ; TRDOS 386 feature only !
00003D81 AA                     <1>       stosb
00003D82 A0[EE890100]           <1>       mov   al, [ACTIVE_PAGE] ; current video page
00003D87 AA                     <1>       stosb
00003D88 66A1[E36F0000]         <1>       mov   ax, [CURSOR_MODE] ; cursor type
00003D8E 66AB                   <1>       stosw
                                <1>       ;mov   eax, [CURSOR_POSN] ; cursor position for video page 0 and 1
                                <1>       ;stosd
                                <1>       ;mov   eax, [CURSOR_POSN+4] ; cursor position for video page 2 and 3
                                <1>       ;stosd
                                <1>       ;mov   eax, [CURSOR_POSN+8] ; cursor position for video page 4 and 5
                                <1>       ;stosd
                                <1>       ;mov   eax, [CURSOR_POSN+12] ; cursor position for video page 6 and 7
                                <1>       ;stosd
00003D90 56                     <1>       push  esi
00003D91 B504                   <1>       mov   ch, 4
00003D93 BE[DE890100]           <1>       mov   esi, CURSOR_POSN
                                <1> bfn_svs_5:
00003D98 A5                     <1>       movsd
00003D99 FECD                   <1>       dec   ch
00003D9B 75FB                   <1>       jnz   short bfn_svs_5
00003D9D 5E                     <1>       pop   esi
                                <1>       ; (font addr) protected mode address in kernel's/system memory space
                                <1>       ; (not accessable/meaningful address value by user)
00003D9E A1[6A960100]           <1>       mov   eax, [VGA_INT43H] ; VGA current (default) font address
00003DA3 AB                     <1>       stosd
                                <1>
                                <1>       ; (total 40 bytes are written above as BIOS data state)
                                <1>
                                <1> bfn_svs_6:
                                <1>       ; 12/01/2021
00003DA4 F6C104                 <1>       test  cl, 4
00003DA7 7423                   <1>       jz    short bfn_svs_8
                                <1>
                                <1>       ;/* XXX: check this */
                                <1>             ; /* read/write mode dac */
                                <1>         ;write_byte(ES, BX, inb(VGAREG_DAC_STATE)); BX++;
                                <1>       ;       ; /* pix address */
                                <1>         ;write_byte(ES, BX, inb(VGAREG_DAC_WRITE_ADDRESS)); BX++;
                                <1>         ;write_byte(ES, BX, inb(VGAREG_PEL_MASK)); BX++;
                                <1>         ;// Set the whole dac always, from 0
                                <1>         ;outb(VGAREG_DAC_WRITE_ADDRESS,0x00);
                                <1>         ;for(i=0;i<256*3;i++) {
                                <1>         ;   write_byte(ES, BX, inb(VGAREG_DAC_DATA)); BX++;
                                <1>         ;}
                                <1>         ;write_byte(ES, BX, 0); BX++; /* color select register */
                                <1>
                                <1>       ; /* read/write mode dac */
00003DA9 66BAC703               <1>       mov   dx, 3C7h ; VGAREG_DAC_STATE
00003DAD EC                     <1>       in    al, dx
00003DAE AA                     <1>       stosb
                                <1>       ; /* pix address */
                                <1>       ;mov   dx, VGAREG_DAC_WRITE_ADDRESS ; 3C8h
                                <1>       ;mov   dl, 0C8h
00003DAF FEC2                   <1>       inc   dl
00003DB1 EC                     <1>       in    al, dx
00003DB2 AA                     <1>       stosb
```

```
9297                                    <1>       ;mov   dx, VGAREG_PEL_MASK  ; 3C6h
9298 00003DB3 B2C6                      <1>       mov    dl, 0C6h
9299 00003DB5 EC                        <1>       in     al, dx
9300 00003DB6 AA                        <1>       stosb
9301                                    <1>       ;// Set the whole dac always, from 0
9302 00003DB7 30C0                      <1>       xor    al, al ; 0
9303                                    <1>       ;mov   dx, VGAREG_DAC_WRITE_ADDRESS ; 3C8h
9304 00003DB9 B2C8                      <1>       mov    dl, 0C8h
9305 00003DBB EE                        <1>       out    dx, al
9306                                    <1>
9307 00003DBC 51                        <1>       push   ecx ; 22/01/2021
9308                                    <1>       ;for(i=0;i<256*3;i++) {
9309 00003DBD B900030000                <1>       mov    ecx, 256*3 ; 768 bytes
9310                                    <1>       ;mov   dx, VGAREG_DAC_DATA ; 3C9h
9311                                    <1>       ;mov   dl, 0C9h
9312 00003DC2 FEC2                      <1>       inc    dl ; dx = 3C9h
9313                                    <1> bfn_svs_7:
9314 00003DC4 EC                        <1>       in     al, dx
9315 00003DC5 AA                        <1>       stosb
9316 00003DC6 E2FC                      <1>       loop   bfn_svs_7
9317 00003DC8 59                        <1>       pop    ecx ; 22/01/2021
9318                                    <1>
9319                                    <1>       ; /* color select register */
9320 00003DC9 28C0                      <1>       sub    al, al ; 0
9321 00003DCB AA                        <1>       stosb
9322                                    <1>
9323                                    <1>       ; (total 772 bytes are written above as DAC state)
9324                                    <1> bfn_svs_8:
9325 00003DCC C3                        <1>       retn
9326                                    <1>
9327                                    <1> vbe_biosfn_save_video_state:
9328                                    <1>       ; 23/01/2021
9329                                    <1>       ; 13/01/2021
9330                                    <1>       ; 12/01/2021 (TRDOS 386 v2.0.3)
9331                                    <1>       ; (vbe.c)
9332                                    <1>
9333                                    <1>       ; modified registers: eax, edx, edi, ch
9334                                    <1>
9335                                    <1>       ; input: edi = state buffer address
9336                                    <1>       ; output:
9337                                    <1>       ;       VBE DISPI register contents will be saved
9338                                    <1>       ;       (18 bytes, 9 words)
9339                                    <1>
9340                                    <1>       ; outw(VBE_DISPI_IOPORT_INDEX,VBE_DISPI_INDEX_ENABLE);
9341                                    <1>       ; enable = inw(VBE_DISPI_IOPORT_DATA);
9342                                    <1>       ; write_word(ES, BX, enable);
9343                                    <1>       ; BX += 2;
9344                                    <1>       ; if (!(enable & VBE_DISPI_ENABLED))
9345                                    <1>       ;     return;
9346                                    <1>       ; for(i = VBE_DISPI_INDEX_XRES;
9347                                    <1>       ;             i <= VBE_DISPI_INDEX_Y_OFFSET; i++) {
9348                                    <1>       ;    if (i != VBE_DISPI_INDEX_ENABLE) {
9349                                    <1>       ;        outw(VBE_DISPI_IOPORT_INDEX, i);
9350                                    <1>       ;        write_word(ES, BX, inw(VBE_DISPI_IOPORT_DATA));
9351                                    <1>         ;    BX += 2;
9352                                    <1>       ;        }
9353                                    <1>       ; }
9354                                    <1>
9355 00003DCD 66BACE01                  <1>       mov    dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
9356 00003DD1 B804000000                <1>       mov    eax, 04h  ; VBE_DISPI_INDEX_ENABLE
9357 00003DD6 66EF                      <1>       out    dx, ax
9358                                    <1>       ;mov   dx, 01CFh ; VBE_DISPI_IOPORT_DATA
9359 00003DD8 FEC2                      <1>       inc    dl
9360 00003DDA 66ED                      <1>       in     ax, dx ; enable (status)
9361 00003DDC 66AB                      <1>       stosw
9362 00003DDE 6683E001                  <1>       and    ax, 1 ; VBE_DISPI_ENABLED
9363 00003DE2 7505                      <1>       jnz    short vbe_bfn_svs_0
9364                                    <1>       ; 23/01/2021
9365                                    <1>       ; ax = 0
9366                                    <1>       ; VBE_DISPI_DISABLED
9367                                    <1>       ; 13/01/2021
9368                                    <1>       ; clear remain 8 bytes
9369                                    <1>       ;xor   eax, eax
9370 00003DE4 AB                        <1>       stosd  ; 2
9371 00003DE5 AB                        <1>       stosd  ; 2
9372 00003DE6 AB                        <1>       stosd  ; 2
9373 00003DE7 AB                        <1>       stosd  ; 2
9374 00003DE8 C3                        <1>       retn
9375                                    <1> vbe_bfn_svs_0:
9376                                    <1>       ; VBE_DISPI_ENABLED
9377                                    <1>
9378                                    <1>       ;sub   eax, eax
9379 00003DE9 28C0                      <1>       sub    al, al ; eax = 0
9380                                    <1>
9381                                    <1>       ; from VBE_DISPI_INDEX_XRES
9382                                    <1>       ;   to VBE_DISPI_INDEX_BPP
9383                                    <1>
9384 00003DEB B503                      <1>       mov    ch, 3
9385                                    <1>       ; al = 0 ; VBE_DISPI_INDEX_XRES - 1
9386                                    <1>
9387 00003DED E804000000                <1>       call   vbe_bfn_svs_1
9388                                    <1>
9389                                    <1>       ; from VBE_DISPI_INDEX_BANK
9390                                    <1>       ;   to VBE_DISPI_INDEX_Y_OFFSET
9391                                    <1>
9392 00003DF2 FEC0                      <1>       inc    al
9393                                    <1>       ; al = 4 ; VBE_DISPI_INDEX_BANK - 1
9394                                    <1>
9395 00003DF4 B505                      <1>       mov    ch, 5
9396                                    <1> vbe_bfn_svs_1:
9397 00003DF6 FEC0                      <1>       inc    al ; from VBE_DISPI_INDEX_XRES
9398                                    <1>                  ;   to VBE_DISPI_INDEX_BPP
9399                                    <1>       ;mov   dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
9400 00003DF8 FECA                      <1>       dec    dl ; 1CEh
9401 00003DFA 66EF                      <1>       out    dx, ax
```

```
9402 00003DFC 50                <1>       push   eax
9403                            <1>       ;mov   dx, 01CFh ; VBE_DISPI_IOPORT_DATA
9404 00003DFD FEC2              <1>       inc    dl ; 1CFh
9405 00003DFF 66ED              <1>       in     ax, dx
9406 00003E01 66AB              <1>       stosw
9407 00003E03 58                <1>       pop    eax
9408 00003E04 FECD              <1>       dec    ch
9409 00003E06 75EE              <1>       jnz    short vbe_bfn_svs_1
9410 00003E08 C3                <1>       retn
9411                            <1>
9412                            <1> ; 11/01/2021
9413                            <1> VGAREG_ACTL_ADDRESS       equ 3C0h
9414                            <1> VGAREG_ACTL_WRITE_DATA    equ 3C0h
9415                            <1> VGAREG_ACTL_READ_DATA     equ 3C1h
9416                            <1>
9417                            <1> VGAREG_INPUT_STATUS       equ 3C2h
9418                            <1> VGAREG_WRITE_MISC_OUTPUT equ 3C2h
9419                            <1> VGAREG_VIDEO_ENABLE       equ 3C3h
9420                            <1> VGAREG_SEQU_ADDRESS       equ 3C4h
9421                            <1> VGAREG_SEQU_DATA   equ 3C5h
9422                            <1>
9423                            <1> VGAREG_PEL_MASK           equ 3C6h
9424                            <1> VGAREG_DAC_STATE   equ 3C7h
9425                            <1> VGAREG_DAC_READ_ADDRESS   equ 3C7h
9426                            <1> VGAREG_DAC_WRITE_ADDRESS equ 3C8h
9427                            <1> VGAREG_DAC_DATA           equ 3C9h
9428                            <1>
9429                            <1> VGAREG_READ_FEATURE_CTL   equ 3CAh
9430                            <1> VGAREG_READ_MISC_OUTPUT   equ 3CCh
9431                            <1>
9432                            <1> VGAREG_GRDC_ADDRESS       equ 3CEh
9433                            <1> VGAREG_GRDC_DATA   equ 3CFh
9434                            <1>
9435                            <1> ;VGAREG_MDA_CRTC_ADDRESS equ 3B4h
9436                            <1> ;VGAREG_MDA_CRTC_DATA     equ 3B5h
9437                            <1> VGAREG_VGA_CRTC_ADDRESS   equ 3D4h
9438                            <1> VGAREG_VGA_CRTC_DATA      equ 3D5h
9439                            <1>
9440                            <1> ;VGAREG_MDA_WRITE_FEATURE_CTL equ 3BAh
9441                            <1> VGAREG_VGA_WRITE_FEATURE_CTL equ 3DAh
9442                            <1> VGAREG_ACTL_RESET  equ 3DAh
9443                            <1>
9444                            <1> ;VGAREG_MDA_MODECTL       equ 3B8h
9445                            <1> VGAREG_CGA_MODECTL equ 3D8h
9446                            <1> VGAREG_CGA_PALETTE equ 3D9h
9447                            <1>
9448                            <1> biosfn_restore_video_state:
9449                            <1>       ; 22/01/2021
9450                            <1>       ; 13/01/2021
9451                            <1>       ; 12/01/2021 (TRDOS 386 v2.0.3)
9452                            <1>       ; (vgabios.c)
9453                            <1>
9454                            <1>       ; modified registers: eax, edx, esi, edi, ch
9455                            <1>
9456                            <1>       ;mov   esi, VBE3SAVERESTOREBLOCK
9457                            <1>
9458                            <1>       ; input: esi = state buffer address
9459                            <1>
9460 00003E09 F6C101            <1>       test   cl, 1
9461 00003E0C 0F84A9000000      <1>       jz     bfn_rvs_6
9462                            <1>
9463 00003E12 66817E40D403      <1>       cmp    word [esi+64], 3D4h ; must be 3D4h
9464 00003E18 7402              <1>       je     short bfn_rvs_0
9465                            <1>                     ; it is seen as valid buffer
9466 00003E1A F9                <1>       stc
9467 00003E1B C3                <1>       retn
9468                            <1>
9469                            <1> bfn_rvs_0:
9470 00003E1C 89F7              <1>       mov    edi, esi ; addr1
9471 00003E1E 83C605            <1>       add    esi, 5 ; skip 1st 5 bytes for now
9472                            <1>
9473                            <1>       ; // Reset Attribute Ctl flip-flop
9474                            <1>         ; inb(VGAREG_ACTL_RESET);
9475 00003E21 66BADA03          <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
9476 00003E25 EC                <1>       in     al, dx
9477                            <1>
9478                            <1>       ; for(i=1;i<=4;i++){
9479 00003E26 B001              <1>       mov    al, 1
9480                            <1>       ;;mov  dx, VGAREG_SEQU_ADDRESS ; 3C4h
9481                            <1>       ;mov   dl, 0C4h
9482 00003E28 B504              <1>       mov    ch, 4
9483                            <1> bfn_rvs_1:
9484                            <1>       ; outb(VGAREG_SEQU_ADDRESS, i);
9485                            <1>       ;mov   dx, VGAREG_SEQU_ADDRESS ; 3C4h
9486 00003E2A B2C4              <1>       mov    dl, 0C4h
9487 00003E2C EE                <1>       out    dx, al
9488                            <1>       ;mov   dx, VGAREG_SEQU_DATA  ; 3C5h
9489 00003E2D FEC2              <1>       inc    dl  ; dx = 3C5h
9490                            <1>       ; outb(VGAREG_SEQU_DATA)
9491 00003E2F 50                <1>       push   eax
9492 00003E30 AC                <1>       lodsb  ; (4 bytes in loop)
9493 00003E31 EE                <1>       out    dx, al
9494 00003E32 58                <1>       pop    eax
9495                            <1>       ;mov   dx, VGAREG_SEQU_ADDRESS ; 3C4h
9496                            <1>       ;dec   dl
9497 00003E33 FEC0              <1>       inc    al  ; i++
9498 00003E35 FECD              <1>       dec    ch
9499 00003E37 75F1              <1>       jnz    short bfn_rvs_1
9500                            <1>
9501                            <1>       ; outb(VGAREG_SEQU_ADDRESS, 0);
9502 00003E39 28C0              <1>       sub    al, al ; 0
9503 00003E3B EE                <1>       out    dx, al
9504                            <1>       ; outb(VGAREG_SEQU_DATA)
9505                            <1>       ;mov   dx, VGAREG_SEQU_DATA ; 3C5h
9506 00003E3C FEC2              <1>       inc    dl  ; dx = 3C5h
```

```
9507 00003E3E AC                <1>       lodsb  ; (+1 byte)
9508 00003E3F EE                <1>       out    dx, al
9509                            <1>
9510                            <1>       ; // Disable CRTC write protection
9511                            <1>       ; outw(crtc_addr,0x0011);
9512                            <1>       ;mov   dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
9513 00003E40 B2D4              <1>       mov    dl, 0D4h
9514 00003E42 66B81100          <1>       mov    ax, 11h
9515 00003E46 66EF              <1>       out    dx, ax
9516                            <1>
9517                            <1>       ; // Set CRTC regs
9518                            <1>
9519                            <1>       ; for(i=0;i<=0x18;i++) {
9520                            <1>         ;   if (i != 0x11) {
9521 00003E48 28C0              <1>       sub    al, al ; 0
9522                            <1>       ;;mov  dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
9523                            <1>       ;mov   dl, 0D4h
9524 00003E4A B519              <1>       mov    ch, 25
9525                            <1> bfn_rvs_2:
9526                            <1>         ; outb(crtc_addr,i);
9527                            <1>       ;mov   dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
9528 00003E4C B2D4              <1>       mov    dl, 0D4h
9529 00003E4E EE                <1>       out    dx, al
9530                            <1>       ;mov   dx, VGAREG_VGA_CRTC_DATA ; 3D5h
9531 00003E4F FEC2              <1>       inc    dl  ; dx = 3D5h
9532                            <1>       ; inb(crtc_addr+1)
9533 00003E51 50                <1>       push   eax
9534 00003E52 AC                <1>       lodsb  ; (25 bytes in loop)
9535 00003E53 EE                <1>       out    dx, al
9536 00003E54 58                <1>       pop    eax
9537                            <1>       ;mov   dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
9538                            <1>       ;dec   dl
9539 00003E55 FEC0              <1>       inc    al  ; i++
9540 00003E57 3C11              <1>       cmp    al, 17 ; 11h
9541 00003E59 7505              <1>       jne    short bfn_rvs_3
9542 00003E5B AC                <1>       lodsb
9543 00003E5C 88C4              <1>       mov    ah, al ; *
9544 00003E5E B012              <1>       mov    al, 18
9545                            <1> bfn_rvs_3:
9546 00003E60 FECD              <1>       dec    ch
9547 00003E62 75E8              <1>       jnz    short bfn_rvs_2
9548                            <1>
9549                            <1>       ; // select crtc base address
9550                            <1>         ; v = inb(VGAREG_READ_MISC_OUTPUT) & ~0x01;
9551                            <1>       ;if (crtc_addr = 0x3d4)
9552                            <1>       ;   v |= 0x01;
9553                            <1>       ; outb(VGAREG_WRITE_MISC_OUTPUT, v);
9554                            <1>
9555                            <1>       ;;mov  dx, VGAREG_READ_MISC_OUTPUT ; 3CCh
9556                            <1>       ;mov   dl, 0CCh
9557                            <1>       ;in    al, dl
9558                            <1>       ;and   al, 1
9559                            <1>       ;;mov  dx, VGAREG_WRITE_MISC_OUTPUT ; 3C2h
9560                            <1>       ;mov   dl, 0C2h
9561                            <1>       ;or    al, 1
9562                            <1>       ;out   dx, al
9563                            <1>
9564                            <1>       ; // enable write protection if needed
9565                            <1>       ;outb(crtc_addr, 0x11);
9566                            <1>         ;outb(crtc_addr+1, read_byte(ES, BX - 0x18 + 0x11));
9567                            <1>       ;mov   dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
9568 00003E64 B2D4              <1>       mov    dl, 0D4h
9569 00003E66 B011              <1>       mov    al, 11h
9570 00003E68 EE                <1>       out    dx, al
9571 00003E69 88E0              <1>       mov    al, ah ; *
9572 00003E6B FEC2              <1>       inc    dl ; dx = 3D5h
9573 00003E6D EE                <1>       out    dx, al
9574                            <1>
9575                            <1>       ; // Set Attribute Ctl
9576 00003E6E 8A6703            <1>       mov    ah, [edi+3] ; addr1+3, ah = ar_index
9577 00003E71 80E420            <1>       and    ah, 20h  ; (ar_index & 0x20)
9578                            <1>
9579                            <1>         ; inb(VGAREG_ACTL_RESET);
9580                            <1>       ;mov   dx, 3DAh ; VGAREG_ACTL_RESET
9581 00003E74 B2DA              <1>       mov    dl, 0DAh
9582 00003E76 EC                <1>       in     al, dx
9583                            <1>
9584                            <1>         ; for(i=0;i<=0x13;i++) {
9585 00003E77 28C0              <1>       sub    al, al ; 0
9586 00003E79 B514              <1>       mov    ch, 20
9587                            <1> bfn_rvs_4:
9588                            <1>       ; outb(VGAREG_ACTL_ADDRESS, i | (ar_index & 0x20));
9589 00003E7B 50                <1>       push   eax
9590 00003E7C 08E0              <1>       or     al, ah
9591                            <1>       ;mov   dx, VGAREG_ACTL_ADDRESS ; 3C0h
9592 00003E7E B2C0              <1>       mov    dl, 0C0h
9593 00003E80 EE                <1>       out    dx, al
9594                            <1>       ;mov   dx, VGAREG_ACTL_WRITE_DATA ; 3C0h
9595                            <1>       ;mov   dl, 0C0h
9596 00003E81 AC                <1>       lodsb  ; (20 bytes in loop)
9597 00003E82 EE                <1>       out    dx, al
9598 00003E83 58                <1>       pop    eax
9599 00003E84 FEC0              <1>       inc    al  ; i++
9600 00003E86 FECD              <1>       dec    ch
9601 00003E88 75F1              <1>       jnz    short bfn_rvs_4
9602                            <1>
9603                            <1>       ; outb(VGAREG_ACTL_ADDRESS, ar_index);
9604                            <1>       ;mov   dx, VGAREG_ACTL_ADDRESS ; 3C0h
9605                            <1>       ;mov   dl, 0C0h
9606 00003E8A 88E0              <1>       mov    al, ah ; ar_index
9607 00003E8C EE                <1>       out    dx, al
9608                            <1>
9609                            <1>       ; inb(VGAREG_ACTL_RESET);
9610                            <1>       ;mov   dx, VGAREG_ACTL_RESET ; 3DAh
9611 00003E8D B2DA              <1>       mov    dl, 0DAh
```

```
9612 00003E8F EC                <1>       in     al, dx
9613                            <1>
9614                            <1>         ; for(i=0;i<=8;i++) {
9615 00003E90 28C0              <1>       sub    al, al ; 0
9616                            <1>       ;;mov  dx, VGAREG_GRDC_ADDRESS ; 3CEh
9617                            <1>       ;mov   dl, 0CEh
9618 00003E92 B509              <1>       mov    ch, 9
9619                            <1> bfn_rvs_5:
9620                            <1>       ; outb(VGAREG_GRDC_ADDRESS,i)
9621                            <1>       ;mov   dx, VGAREG_GRDC_ADDRESS ; 3CEh
9622 00003E94 B2CE              <1>       mov    dl, 0CEh
9623 00003E96 EE                <1>       out    dx, al
9624                            <1>       ; outb(VGAREG_ACTL_READ_DATA)
9625 00003E97 50                <1>       push   eax
9626                            <1>       ;mov   dx, VGAREG_GRDC_DATA ; 3CFh
9627                            <1>       ;mov   dl, 0CFh
9628 00003E98 FEC2              <1>       inc    dl
9629 00003E9A AC                <1>       lodsb  ; (9 bytes in loop)
9630 00003E9B EE                <1>       out    dx, al
9631 00003E9C 58                <1>       pop    eax
9632                            <1>       ;dec   dl
9633 00003E9D FEC0              <1>       inc    al  ; i++
9634 00003E9F FECD              <1>       dec    ch
9635 00003EA1 75F1              <1>       jnz    short bfn_rvs_5
9636                            <1>
9637                            <1>       ; BX += 2; /* crtc_addr */     ; 3D4h
9638                            <1>         ; BX += 4; /* plane latches */ ; 0
9639 00003EA3 83C606            <1>       add    esi, 6
9640 00003EA6 56                <1>       push   esi ; *
9641                            <1>
9642                            <1>       ;outb(VGAREG_SEQU_ADDRESS, read_byte(ES, addr1)); addr1++;
9643                            <1>         ;outb(crtc_addr, read_byte(ES, addr1)); addr1++;
9644                            <1>         ;outb(VGAREG_GRDC_ADDRESS, read_byte(ES, addr1)); addr1++;
9645                            <1>         ;addr1++;
9646                            <1>         ;outb(crtc_addr - 0x4 + 0xa, read_byte(ES, addr1)); addr1++;
9647                            <1>
9648 00003EA7 89FE              <1>       mov    esi, edi ; start of state buffer
9649                            <1>
9650                            <1>       ;mov   dx, VGAREG_SEQU_ADDRESS ; 3C7h
9651 00003EA9 B2C7              <1>       mov    dl, 0C7h
9652 00003EAB AC                <1>       lodsb
9653 00003EAC EE                <1>       out    dx, al
9654                            <1>       ;mov   dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
9655 00003EAD B2D4              <1>       mov    dl, 0D4h
9656 00003EAF AC                <1>       lodsb
9657 00003EB0 EE                <1>       out    dx, al
9658                            <1>       ;mov   dx, VGAREG_GRDC_ADDRESS ; 3CEh
9659 00003EB1 B2CE              <1>         mov  dl, 0CEh
9660 00003EB3 AC                <1>       lodsb
9661 00003EB4 EE                <1>       out    dx, al
9662 00003EB5 AC                <1>       lodsb  ; addr1++
9663                            <1>       ;mov   dx, VGAREG_VGA_WRITE_FEATURE_CTL ; 3DAh
9664 00003EB6 B2DA              <1>       mov    dl, 0DAh
9665 00003EB8 AC                <1>       lodsb
9666 00003EB9 EE                <1>       out    dx, al
9667                            <1>
9668 00003EBA 5E                <1>       pop    esi ; *
9669                            <1>
9670                            <1>       ; (total 70 bytes are read above as controller hardware state)
9671                            <1>
9672                            <1> bfn_rvs_6:
9673                            <1>       ; 13/01/2021
9674 00003EBB F6C102            <1>       test   cl, 2
9675 00003EBE 747D              <1>       jz     short bfn_rvs_9
9676                            <1>
9677                            <1>       ; VIDEO BIOS DATA
9678                            <1>       ; !!! this data is valid for TRDOS 386 v2 kernel only !!!
9679                            <1>       ; (this is not same with BOCHS/PLEX86 video bios, BIOS data)
9680                            <1>
9681                            <1>       ; if (CX & 2) {
9682                            <1>       ;write_byte(BIOSMEM_SEG,BIOSMEM_CURRENT_MODE, read_byte(ES, BX)); BX++;
9683                            <1>         ;write word(BIOSMEM SEG,BIOSMEM NB COLS, read word(ES, BX)); BX += 2;
9684                            <1>         ;write_word(BIOSMEM_SEG,BIOSMEM_PAGE_SIZE, read_word(ES, BX)); BX += 2;
9685                            <1>         ;write_word(BIOSMEM_SEG,BIOSMEM_CRTC_ADDRESS, read_word(ES, BX)); BX += 2;
9686                            <1>         ;write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, read_byte(ES, BX)); BX++;
9687                            <1>         ;write_word(BIOSMEM_SEG,BIOSMEM_CHAR_HEIGHT, read_word(ES, BX)); BX += 2;
9688                            <1>         ;write_byte(BIOSMEM_SEG,BIOSMEM_VIDEO_CTL, read_byte(ES, BX)); BX++;
9689                            <1>         ;write_byte(BIOSMEM_SEG,BIOSMEM_SWITCHES, read_byte(ES, BX)); BX++;
9690                            <1>         ;write_byte(BIOSMEM_SEG,BIOSMEM_MODESET_CTL, read_byte(ES, BX)); BX++;
9691                            <1>         ;write_word(BIOSMEM_SEG,BIOSMEM_CURSOR_TYPE, read_word(ES, BX)); BX += 2;
9692                            <1>         ;for(i=0;i<8;i++) {
9693                            <1>         ;   write_word(BIOSMEM_SEG, BIOSMEM_CURSOR_POS+2*i, read_word(ES, BX));
9694                            <1>         ;   BX += 2;
9695                            <1>         ;}
9696                            <1>         ;write_word(BIOSMEM_SEG,BIOSMEM_CURRENT_START, read_word(ES, BX)); BX += 2;
9697                            <1>         ;write_byte(BIOSMEM_SEG,BIOSMEM_CURRENT_PAGE, read_byte(ES, BX)); BX++;
9698                            <1>         ;/* current font */
9699                            <1>         ;write_word(0, 0x1f * 4, read_word(ES, BX)); BX += 2;
9700                            <1>         ;write_word(0, 0x1f * 4 + 2, read_word(ES, BX)); BX += 2;
9701                            <1>         ;write_word(0, 0x43 * 4, read_word(ES, BX)); BX += 2;
9702                            <1>         ;write_word(0, 0x43 * 4 + 2, read_word(ES, BX)); BX += 2;
9703                            <1>
9704                            <1>       ; !!! save TRDOS 386 v2 kernel spesific video bios data !!!
9705                            <1>       ; (which is/are used by 'SET_MODE' function and/or it's sub functions)
9706                            <1>
9707 00003EC0 66AD              <1>       lodsw   ; CRTC_ADDR, always 3D4h (color VGA) for TRDOS 386 v2
9708                            <1>       ; skip 3D4h check if it is already checked
9709 00003EC2 F6C101            <1>       test   cl, 1
9710 00003EC5 7508              <1>       jnz    short bfn_rvs_7
9711 00003EC7 663DD403          <1>       cmp    ax, 3D4h
9712 00003ECB 7402              <1>       je     short bfn_rvs_7
9713 00003ECD F9                <1>       stc
9714 00003ECE C3                <1>       retn
9715                            <1> bfn_rvs_7:
9716 00003ECF AC                <1>       lodsb
```

```
9717 00003ED0 A2[CA6F0000]        <1>       mov   [CRT_MODE], al ; Current video mode (0FFh for VESA VBE modes)
9718 00003ED5 AC                  <1>       lodsb
9719 00003ED6 A2[CB6F0000]        <1>       mov   [CRT_MODE_SET], al ; 29h for mode 03h ; TRDOS 386 feature only !
9720 00003EDB 66AD                <1>       lodsw
9721 00003EDD 66A3[1E120300]      <1>       mov   [video_mode], ax ; Current VESA VBE (SVGA, extended VGA) mode
9722 00003EE3 66AD                <1>       lodsw         ; (valid if [CRT_MODE] = 0FFh)
9723 00003EE5 66A3[58960100]      <1>       mov   [CRT_LEN], ax ; page size (in bytes)
9724 00003EEB 66AD                <1>       lodsw
9725 00003EED 66A3[DC890100]      <1>       mov   [CRT_START], ax ; video page start offset
9726 00003EF3 AC                  <1>       lodsb
9727 00003EF4 A2[CC6F0000]        <1>       mov   [CRT_COLS], al ; nbcols, characters per row
9728 00003EF9 AC                  <1>       lodsb
9729 00003EFA A2[D26F0000]        <1>       mov   [VGA_ROWS], al ; nbrows, (character) rows per page (not rows-1)
9730 00003EFF AC                  <1>       lodsb
9731 00003F00 A2[CE6F0000]        <1>       mov   [CHAR_HEIGHT], al ; character font height (8 or 16 or 14)
9732 00003F05 AC                  <1>       lodsb
9733 00003F06 A2[CF6F0000]        <1>       mov   [VGA_VIDEO_CTL], al ; ROM BIOS DATA AREA Offset 87h
9734 00003F0B AC                  <1>       lodsb
9735 00003F0C A2[D06F0000]        <1>       mov   [VGA_SWITCHES], al ; feature bit switches
9736 00003F11 AC                  <1>       lodsb
9737 00003F12 A2[D16F0000]        <1>       mov   [VGA_MODESET_CTL], al ; basic mode set options
9738                              <1>       ; followings are only used by TRDOS 386 v2 (IBM PC/AT ROMBIOS) code
9739                              <1>       ; (bochs/plex86 does not use and return those)
9740 00003F17 AC                  <1>       lodsb
9741 00003F18 A2[CD6F0000]        <1>       mov   [CRT_PALETTE], al ; current color palette ; TRDOS 386 feature only !
9742 00003F1D AC                  <1>       lodsb
9743 00003F1E A2[EE890100]        <1>       mov   [ACTIVE_PAGE], al ; current video page
9744 00003F23 66AD                <1>       lodsw
9745 00003F25 66A3[E36F0000]      <1>       mov   [CURSOR_MODE], ax ; cursor type
9746                              <1>       ;lodsd
9747                              <1>       ;mov   [CURSOR_POSN], eax ; cursor position for video page 0 and 1
9748                              <1>       ;lodsd
9749                              <1>       ;mov   [CURSOR_POSN+4], eax ; cursor position for video page 2 and 3
9750                              <1>       ;lodsd
9751                              <1>       ;mov   [CURSOR_POSN+8], eax ; cursor position for video page 4 and 5
9752                              <1>       ;lodsd
9753                              <1>       ;mov   [CURSOR_POSN+12], eax ; cursor position for video page 6 and 7
9754 00003F2B B504                <1>       mov   ch, 4
9755 00003F2D BF[DE890100]        <1>       mov   edi, CURSOR_POSN
9756                              <1> bfn_rvs_8:
9757 00003F32 A5                  <1>       movsd
9758 00003F33 FECD                <1>       dec   ch
9759 00003F35 75FB                <1>       jnz   short bfn_rvs_8
9760                              <1>       ; (font addr) protected mode address in kernel's/system memory space
9761                              <1>       ; (not accessable/meaningful address value by user)
9762 00003F37 AD                  <1>       lodsd
9763 00003F38 A3[6A960100]        <1>       mov   [VGA_INT43H], eax ; VGA current (default) font address
9764                              <1>
9765                              <1>       ; (total 40 bytes are read&written above as BIOS data state)
9766                              <1> bfn_rvs_9:
9767                              <1>       ; 13/01/2021
9768 00003F3D F6C104              <1>       test  cl, 4
9769 00003F40 7422                <1>       jz    short bfn_rvs_11
9770                              <1>
9771                              <1>       ;BX++;
9772                              <1>         ;v = read_byte(ES, BX); BX++;
9773                              <1>         ;outb(VGAREG_PEL_MASK, read_byte(ES, BX)); BX++;
9774                              <1>         ;// Set the whole dac always, from 0
9775                              <1>         ;outb(VGAREG_DAC_WRITE_ADDRESS,0x00);
9776                              <1>         ;for(i=0;i<256*3;i++) {
9777                              <1>         ;   outb(VGAREG_DAC_DATA, read_byte(ES, BX)); BX++;
9778                              <1>         ;}
9779                              <1>         ;BX++;
9780                              <1>         ;outb(VGAREG_DAC_WRITE_ADDRESS, v);
9781                              <1>
9782                              <1>       ; /* read/write mode dac */
9783 00003F42 AC                  <1>       lodsb  ; skip ; VGAREG_DAC_STATE
9784 00003F43 AC                  <1>       lodsb
9785 00003F44 88C4                <1>       mov   ah, al ; * ; v
9786 00003F46 AC                  <1>       lodsb
9787 00003F47 66BAC603            <1>       mov   dx, VGAREG_PEL_MASK  ; 3C6h
9788 00003F4B EE                  <1>       out   dx, al
9789                              <1>       ;// Set the whole dac always, from 0
9790 00003F4C 30C0                <1>       xor   al, al ; 0
9791                              <1>       ;mov   dx, VGAREG_DAC_WRITE_ADDRESS ; 3C8h
9792 00003F4E B2C8                <1>       mov   dl, 0C8h
9793 00003F50 EE                  <1>       out   dx, al
9794                              <1>
9795 00003F51 51                  <1>       push  ecx ; 22/01/2021
9796                              <1>       ;for(i=0;i<256*3;i++) {
9797 00003F52 B900030000          <1>       mov   ecx, 256*3 ; 768 bytes
9798                              <1>       ;mov   dx, VGAREG_DAC_DATA ; 3C9h
9799                              <1>       ;mov   dl, 0C9h
9800 00003F57 FEC2                <1>       inc   dl ; dx = 3C9h
9801                              <1> bfn_rvs_10:
9802 00003F59 AC                  <1>       lodsb
9803 00003F5A EE                  <1>       out   dx, al
9804 00003F5B E2FC                <1>       loop  bfn_rvs_10
9805 00003F5D 59                  <1>       pop   ecx ; 22/01/2021
9806                              <1>
9807                              <1>       ; /* color select register */
9808 00003F5E AC                  <1>       lodsb  ; skip
9809                              <1>
9810 00003F5F 88E0                <1>       mov   al, ah ; * ; v
9811                              <1>
9812                              <1>       ;mov   dx, VGAREG_DAC_WRITE_ADDRESS ; 3C8h
9813                              <1>       ;mov   dl, 0C8h
9814 00003F61 FECA                <1>       dec   dl ; dx  = 3C8h
9815 00003F63 EE                  <1>       out   dx, al ; * ; v
9816                              <1>
9817                              <1>       ; (total 772 bytes are read above as DAC state)
9818                              <1> bfn_rvs_11:
9819 00003F64 C3                  <1>       retn
9820                              <1>
9821                              <1> vbe_biosfn_restore_video_state:
```

```
                        <1>       ; 23/01/2021
                        <1>       ; 13/01/2021 (TRDOS 386 v2.0.3)
                        <1>       ; (vbe.c)
                        <1>
                        <1>       ; modified registers: eax, edx, esi, ch
                        <1>
                        <1>       ; input: esi = state buffer address
                        <1>       ; output:
                        <1>       ;      VBE DISPI register contents will be restored
                        <1>       ;      (18 bytes, 9 words)
                        <1>
                        <1>       ; enable = read_word(ES, BX);
                        <1>       ; BX += 2;
                        <1>       ;
                        <1>       ; if (!(enable & VBE_DISPI_ENABLED)) {
                        <1>         ;   outw(VBE_DISPI_IOPORT_INDEX,VBE_DISPI_INDEX_ENABLE);
                        <1>         ;   outw(VBE_DISPI_IOPORT_DATA, enable);
                        <1>       ; } else {
                        <1>         ;   outw(VBE_DISPI_IOPORT_INDEX, VBE_DISPI_INDEX_XRES);
                        <1>         ;   outw(VBE_DISPI_IOPORT_DATA, read_word(ES, BX));
                        <1>         ;   BX += 2;
                        <1>         ;   outw(VBE_DISPI_IOPORT_INDEX, VBE_DISPI_INDEX_YRES);
                        <1>         ;   outw(VBE_DISPI_IOPORT_DATA, read_word(ES, BX));
                        <1>         ;   BX += 2;
                        <1>         ;   outw(VBE_DISPI_IOPORT_INDEX, VBE_DISPI_INDEX_BPP);
                        <1>         ;   outw(VBE_DISPI_IOPORT_DATA, read_word(ES, BX));
                        <1>         ;   BX += 2;
                        <1>         ;   outw(VBE_DISPI_IOPORT_INDEX,VBE_DISPI_INDEX_ENABLE);
                        <1>         ;   outw(VBE_DISPI_IOPORT_DATA, enable);
                        <1>       ;
                        <1>         ;  for(i = VBE_DISPI_INDEX_BANK; i <= VBE_DISPI_INDEX_Y_OFFSET; i++)
                        <1>       ;    {
                        <1>         ;     outw(VBE_DISPI_IOPORT_INDEX, i);
                        <1>         ;     outw(VBE_DISPI_IOPORT_DATA, read_word(ES, BX));
                        <1>         ;     BX += 2;
                        <1>         ;    }
                        <1>         ; }
                        <1>
00003F65 66AD           <1>       lodsw  ; enable (status, enabled=1, disabled=0)
00003F67 66BACE01       <1>       mov    dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
                        <1>       ; 23/01/2021
00003F6B 6683E001       <1>       and    ax, 1 ; VBE_DISPI_ENABLED
00003F6F 750B           <1>       jnz    short vbe_bfn_rvs_1
                        <1>       ; ax = 0
                        <1>       ; VBE_DISPI_DISABLED
                        <1> vbe_bfn_rvs_0:
                        <1>       ; enable (disable) dispi
                        <1>       ; dx = 01CEh ; VBE_DISPI_IOPORT_INDEX
                        <1>       ; ah = 0
00003F71 50             <1>       push   eax
00003F72 B004           <1>       mov    al, 04h      ; VBE_DISPI_INDEX_ENABLE
00003F74 66EF           <1>       out    dx, ax
                        <1>       ;mov   dx, 01CFh ; VBE_DISPI_IOPORT_DATA
00003F76 FEC2           <1>       inc    dl
00003F78 58             <1>       pop    eax
00003F79 66EF           <1>       out    dx, ax ; enable (or disable)
00003F7B C3             <1>       retn
                        <1> vbe_bfn_rvs_1:
                        <1>       ; VBE_DISPI_ENABLED
                        <1>
                        <1>       ; from VBE_DISPI_INDEX_XRES
                        <1>       ;   to VBE_DISPI_INDEX_BPP
                        <1>
00003F7C B503           <1>       mov    ch, 3
00003F7E 28C0           <1>       sub    al, al ; 0 ; VBE_DISPI_INDEX_XRES - 1
                        <1>       ; ax = 0
                        <1>
00003F80 E80B000000     <1>       call   vbe_bfn_rvs_2
                        <1>
                        <1>         ;outw(VBE_DISPI_IOPORT_INDEX,VBE_DISPI_INDEX_ENABLE);
                        <1>       ;outw(VBE_DISPI_IOPORT_DATA, enable);
                        <1>
                        <1>       ; 23/01/2021
00003F85 B001           <1>       mov    al, 1 ; VBE_DISPI_ENABLED
                        <1>       ; ax = 1
00003F87 E8E5FFFFFF     <1>       call   vbe_bfn_rvs_0
                        <1>
                        <1>       ; from VBE_DISPI_INDEX_BANK
                        <1>       ;   to VBE_DISPI_INDEX_Y_OFFSET
                        <1>
00003F8C B505           <1>       mov    ch, 5
                        <1>       ; 23/01/2021
00003F8E B004           <1>       mov    al, 4  ; VBE_DISPI_INDEX_BANK - 1
                        <1>       ; ax = 4
                        <1> vbe_bfn_rvs_2:
00003F90 FEC0           <1>       inc    al ; from VBE_DISPI_INDEX_XRES
                        <1>                 ; to VBE_DISPI_INDEX_BPP
                        <1>       ;mov   dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
                        <1>       ;mov   dl, 0CEh
00003F92 66EF           <1>       out    dx, ax
00003F94 50             <1>       push   eax
                        <1>       ;mov   dx, 01CFh ; VBE_DISPI_IOPORT_DATA
00003F95 FEC2           <1>       inc    dl ; 1CFh
00003F97 66AD           <1>       lodsw
00003F99 66EF           <1>       out    dx, ax
00003F9B 58             <1>       pop    eax
00003F9C FECA           <1>       dec    dl ; 1CEh
00003F9E FECD           <1>       dec    ch
00003FA0 75EE           <1>       jnz    short vbe_bfn_rvs_2
00003FA2 C3             <1>       retn
                        <1>
                        <1> ; ---------------------------------------------------------
                        <1>
                        <1> dispi_set_enable:
                        <1>       ; 23/11/2020
```

```
                                <1>       ; Input:
                                <1>       ;      ax = VBE_DISPI_ENABLED = 1
                                <1>       ;           or VBE_DISPI_DISABLED = 0
                                <1>       ;
                                <1>       ; Modified registers: none
                                <1>
                                <1>       ;push  edx
                                <1>       ;push  eax
                                <1>       ;mov   dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
                                <1>       ;mov   ax, 04h        ; VBE_DISPI_INDEX_ENABLE
                                <1>       ;out   dx, ax
                                <1>       ;pop   eax
                                <1>       ;;mov  dx, 01CFh ; VBE_DISPI_IOPORT_DATA
                                <1>       ;;mov  dl, 0CFh
                                <1>       ;inc   dl
                                <1>       ;out   dx, ax
                                <1>       ;pop   edx
                                <1>       ;retn
                                <1>
                                <1>       ; 25/11/2020
                                <1>       ; Modified registers: edx
                                <1>       ;push  edx
00003FA3 66BA0400               <1>       mov    dx, 04h ; VBE_DISPI_INDEX_ENABLE
                                <1>       ;call  dispi_set_parms
                                <1>       ;pop   edx
                                <1>       ;retn
                                <1>       ;;jmp  short dispi_set_parms
                                <1>
                                <1> dispi_set_parms:
                                <1>       ; 25/11/2020
                                <1>       ; Input:
                                <1>       ;      ax = data
                                <1>       ;      dx = vbe dispi register index
                                <1>       ;
                                <1>       ; Modified registers: edx
                                <1>
00003FA7 50                     <1>       push   eax
00003FA8 6689D0                 <1>       mov    ax, dx
00003FAB 66BACE01               <1>       mov    dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
00003FAF 66EF                   <1>       out    dx, ax
00003FB1 58                     <1>       pop    eax
                                <1>       ;mov   dx, 01CFh ; VBE_DISPI_IOPORT_DATA
                                <1>       ;mov   dl, 0CFh
00003FB2 FEC2                   <1>       inc    dl
00003FB4 66EF                   <1>       out    dx, ax
00003FB6 C3                     <1>       retn
                                <1>
                                <1> dispi_set_bpp:
                                <1>       ; 25/11/2020
                                <1>       ; Input:
                                <1>       ;      ax = Bits per pixel value
                                <1>       ;           (8,16,24,32)
                                <1>       ;
                                <1>       ; Modified registers: none
                                <1>
                                <1>       ;push  edx
                                <1>       ;push  eax
                                <1>       ;mov   dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
                                <1>       ;mov   ax, 03h        ; VBE_DISPI_INDEX_BPP
                                <1>       ;out   dx, ax
                                <1>       ;pop   eax
                                <1>       ;;mov  dx, 01CFh ; VBE_DISPI_IOPORT_DATA
                                <1>       ;;mov  dl, 0CFh
                                <1>       ;inc   dl
                                <1>       ;out   dx, ax
                                <1>       ;pop   edx
                                <1>       ;retn
                                <1>
                                <1>       ; 25/11/2020
                                <1>       ; Modified registers: edx
                                <1>       ;push  edx
00003FB7 66BA0300               <1>       mov    dx, 03h ; VBE_DISPI_INDEX_BPP
                                <1>       ;call  dispi_set_parms
                                <1>       ;pop   edx
                                <1>       ;retn
00003FBB EBEA                   <1>       jmp    short dispi_set_parms
                                <1>
                                <1> dispi_set_xres:
                                <1>       ; 25/11/2020
                                <1>       ; Input:
                                <1>       ;      ax = X resolution (screen witdh)
                                <1>       ;           (320,640,800,1024,1280,1920)
                                <1>       ;
                                <1>       ; Modified registers: none
                                <1>
                                <1>       ;push  edx
                                <1>       ;push  eax
                                <1>       ;mov   dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
                                <1>       ;mov   ax, 01h        ; VBE_DISPI_INDEX_XRES
                                <1>       ;out   dx, ax
                                <1>       ;pop   eax
                                <1>       ;;mov  dx, 01CFh ; VBE_DISPI_IOPORT_DATA
                                <1>       ;;mov  dl, 0CFh
                                <1>       ;inc   dl
                                <1>       ;out   dx, ax
                                <1>       ;pop   edx
                                <1>       ;retn
                                <1>
                                <1>       ; 25/11/2020
                                <1>       ; Modified registers: edx
                                <1>       ;push  edx
00003FBD 66BA0100               <1>       mov    dx, 01h ; VBE_DISPI_INDEX_XRES
                                <1>       ;call  dispi_set_parms
                                <1>       ;pop   edx
                                <1>       ;retn
```

```
10032 00003FC1 EBE4               <1>       jmp    short dispi_set_parms
10033                             <1>
10034                             <1> dispi_set_yres:
10035                             <1>       ; 25/11/2020
10036                             <1>       ; Input:
10037                             <1>       ;      ax = Y resolution (screen height)
10038                             <1>       ;           (200,400,600,720,768,1080)
10039                             <1>       ;
10040                             <1>       ; Modified registers: none
10041                             <1>
10042                             <1>       ;push  edx
10043                             <1>       ;push  eax
10044                             <1>       ;mov   dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
10045                             <1>       ;mov   ax, 02h        ; VBE_DISPI_INDEX_YRES
10046                             <1>       ;out   dx, ax
10047                             <1>       ;pop   eax
10048                             <1>       ;;mov  dx, 01CFh ; VBE_DISPI_IOPORT_DATA
10049                             <1>       ;;mov  dl, 0CFh
10050                             <1>       ;inc   dl
10051                             <1>       ;out   dx, ax
10052                             <1>       ;pop   edx
10053                             <1>       ;retn
10054                             <1>
10055                             <1>       ; 25/11/2020
10056                             <1>       ; Modified registers: edx
10057                             <1>       ;push  edx
10058 00003FC3 66BA0200          <1>       mov    dx, 02h  ; VBE_DISPI_INDEX_YRES
10059                             <1>       ;call  dispi_set_parms
10060                             <1>       ;pop   edx
10061                             <1>       ;retn
10062 00003FC7 EBDE               <1>       jmp    short dispi_set_parms
10063                             <1>
10064                             <1> dispi_set_bank:
10065                             <1>       ; 25/11/2020
10066                             <1>       ; Input:
10067                             <1>       ;      ax = video memory bank number
10068                             <1>       ;
10069                             <1>       ; Modified registers: none
10070                             <1>
10071                             <1>       ;push  edx
10072                             <1>       ;push  eax
10073                             <1>       ;mov   dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
10074                             <1>       ;mov   ax, 05h        ; VBE_DISPI_INDEX_BANK
10075                             <1>       ;out   dx, ax
10076                             <1>       ;pop   eax
10077                             <1>       ;;mov  dx, 01CFh ; VBE_DISPI_IOPORT_DATA
10078                             <1>       ;;mov  dl, 0CFh
10079                             <1>       ;inc   dl
10080                             <1>       ;out   dx, ax
10081                             <1>       ;pop   edx
10082                             <1>       ;retn
10083                             <1>
10084                             <1>       ; 25/11/2020
10085                             <1>       ; Modified registers: edx
10086                             <1>       ;push  edx
10087 00003FC9 66BA0500          <1>       mov    dx, 05h  ; VBE_DISPI_INDEX_BANK
10088                             <1>       ;call  dispi_set_parms
10089                             <1>       ;pop   edx
10090                             <1>       ;retn
10091 00003FCD EBD8               <1>       jmp    short dispi_set_parms
10092                             <1>
10093                             <1> dispi_get_enable:
10094                             <1>       ; 27/11/2020
10095                             <1>       ; Input:
10096                             <1>       ;      none
10097                             <1>       ; Output:
10098                             <1>       ;      ax = vbe dispi status
10099                             <1>       ;
10100                             <1>       ; Modified registers: eax
10101                             <1>
10102                             <1>       ;push  edx
10103                             <1>       ;mov   dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
10104                             <1>       ;mov   ax, 04h        ; VBE_DISPI_INDEX_ENABLE
10105                             <1>       ;out   dx, ax
10106                             <1>       ;;mov  dx, 01CFh ; VBE_DISPI_IOPORT_DATA
10107                             <1>       ;;mov  dl, 0CFh
10108                             <1>       ;inc   dl
10109                             <1>       ;in    ax, dx
10110                             <1>       ;pop   edx
10111                             <1>       ;retn
10112                             <1>
10113                             <1>       ; 27/11/2020
10114                             <1>       ; Modified registers: eax, edx
10115                             <1>       ;push  edx
10116 00003FCF 66B80400          <1>       mov    ax, 04h ; VBE_DISPI_INDEX_ENABLE
10117                             <1>       ;call  dispi_get_parms
10118                             <1>       ;pop   edx
10119                             <1>       ;retn
10120                             <1>       ;;jmp  short dispi_get_parms
10121                             <1>
10122                             <1> dispi_get_parms:
10123                             <1>       ; 25/11/2020
10124                             <1>       ; Input:
10125                             <1>       ;      ax = vbe dispi register index
10126                             <1>       ; output:
10127                             <1>       ;      ax = data
10128                             <1>       ;
10129                             <1>       ; Modified registers: eax, edx
10130                             <1>
10131 00003FD3 66BACE01          <1>       mov    dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
10132 00003FD7 66EF               <1>       out    dx, ax
10133                             <1>       ;mov   dx, 01CFh ; VBE_DISPI_IOPORT_DATA
10134                             <1>       ;mov   dl, 0CFh
10135 00003FD9 FEC2               <1>       inc    dl
10136 00003FDB 66ED               <1>       in     ax, dx
```

```
10137 00003FDD C3                <1>       retn
10138                            <1>
10139                            <1> vga_compat_setup:
10140                            <1>       ; 26/11/2020
10141                            <1>       ; 25/11/2020
10142                            <1>       ; VGA compatibility setup
10143                            <1>       ; (vbe.c, 02/01/2020, vruppert)
10144                            <1>       ;
10145                            <1>       ; Input:
10146                            <1>       ;     none
10147                            <1>       ;
10148                            <1>       ; Modified registers: eax, edx
10149                            <1>
10150                            <1>       ; 26/11/2020
10151                            <1>       ;push  eax
10152                            <1>       ;push  edx
10153                            <1>
10154                            <1>       ; set CRT X resolution
10155 00003FDE 66BACE01          <1>       mov    dx, 1CEh ; VBE_DISPI_IOPORT_INDEX
10156 00003FE2 66B80100          <1>       mov    ax, 01h  ; VBE_DISPI_INDEX_XRES
10157 00003FE6 66EF              <1>       out    dx, ax
10158                            <1>       ;mov   dx, 1CFh ; VBE_DISPI_IOPORT_DATA
10159 00003FE8 FEC2              <1>       inc    dl
10160 00003FEA 66ED              <1>       in     ax, dx
10161 00003FEC 50                <1>       push   eax
10162 00003FED 66BAD403          <1>       mov    dx, 3D4h ; VGAREG_VGA_CRTC_ADDRESS
10163 00003FF1 66B81100          <1>       mov    ax, 0011h ; Vertical retrace end register
10164 00003FF5 66EF              <1>       out    dx, ax
10165                            <1>       ;pop   eax
10166                            <1>       ;push  eax
10167 00003FF7 8B0424            <1>       mov    eax, [esp]
10168 00003FFA 66C1E803          <1>       shr    ax, 3 ; / 8 for pixel to character
10169 00003FFE 6648              <1>       dec    ax  ; - 1 (EGA or VGA?)
10170 00004000 88C4              <1>       mov    ah, al
10171 00004002 B001              <1>       mov    al, 01h ; Horizontal display end register
10172 00004004 66EF              <1>       out    dx, ax
10173 00004006 58                <1>       pop    eax
10174                            <1>
10175 00004007 E8B0000000        <1>       call   vga_set_virt_width
10176                            <1>
10177                            <1>       ; set CRT Y resolution
10178 0000400C 66BACE01          <1>       mov    dx, 1CEh ; VBE_DISPI_IOPORT_INDEX
10179 00004010 66B80200          <1>       mov    ax, 02h  ; VBE_DISPI_INDEX_YRES
10180 00004014 66EF              <1>       out    dx, ax
10181                            <1>       ;mov   dx, 1CFh ; VBE_DISPI_IOPORT_DATA
10182 00004016 FEC2              <1>       inc    dl
10183 00004018 66ED              <1>       in     ax, dx
10184 0000401A 50                <1>       push   eax
10185 0000401B 66BAD403          <1>       mov    dx, 3D4h ; VGAREG_VGA_CRTC_ADDRESS
10186 0000401F 88C4              <1>       mov    ah, al
10187 00004021 B012              <1>       mov    al, 12h ; Vertical display end register
10188 00004023 66EF              <1>       out    dx, ax
10189 00004025 58                <1>       pop    eax
10190 00004026 B007              <1>       mov    al, 07h      ; Overflow register
10191 00004028 EE                <1>       out    dx, al
10192 00004029 6642              <1>       inc    dx
10193 0000402B EC                <1>       in     al, dx ; read overflow register
10194 0000402C 24BD              <1>       and    al, 0BDh ; clear VDE 9th and 10th bits
10195 0000402E F6C401            <1>       test   ah, 01h
10196 00004031 7402              <1>       jz     short bit8_clear
10197 00004033 0C02              <1>       or     al, 02h ; VDE 9th bit (bit 8) in bit 1
10198                            <1> bit8_clear:
10199 00004035 F6C402            <1>       test   ah, 02h
10200 00004038 7402              <1>       jz     short bit9_clear
10201 0000403A 0C40              <1>       or     al, 40h ; VDE 10th bit (bit 9) in bit 6
10202                            <1> bit9_clear:
10203 0000403C EE                <1>       out    dx, al
10204                            <1>
10205                            <1>       ; other settings
10206 0000403D 66BAD403          <1>       mov    dx, 3D4h ; VGAREG_VGA_CRTC_ADDRESS
10207 00004041 66B80900          <1>       mov    ax, 0009h ; Maximum scan line register
10208 00004045 66EF              <1>       out    dx, ax ; Reset
10209 00004047 B017              <1>       mov    al, 17h ; Mode control register
10210 00004049 EE                <1>       out    dx, al
10211                            <1>       ;mov   dx, 3D5h ; VGAREG_VGA_CRTC_DATA
10212 0000404A FEC2              <1>       inc    dl
10213 0000404C EC                <1>       in     al, dx ; Read mode control register
10214 0000404D 0C03              <1>       or     al, 03h ; Set SRS and CMS bits
10215 0000404F EE                <1>       out    dx, al
10216 00004050 66BADA03          <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
10217 00004054 EC                <1>       in     al, dx  ; clear flip-flop
10218 00004055 66BAC003          <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
10219 00004059 B010              <1>       mov    al, 10h      ; Mode control register
10220 0000405B EE                <1>       out    dx, al
10221                            <1>       ;mov   dx, 3C1h ; VGAREG_ACTL_READ_DATA
10222 0000405C FEC2              <1>       inc    dl
10223 0000405E EC                <1>       in     al, dx
10224 0000405F 0C01              <1>       or     al, 01h ; select graphics mode
10225                            <1>       ;mov   dx, 3C0h ; VGAREG_ACTL_ADDRESS
10226 00004061 FECA              <1>       dec    dl
10227 00004063 EE                <1>       out    dx, al ; Write to mode control register
10228 00004064 B020              <1>       mov    al, 20h ; Palette RAM <-> display memory
10229 00004066 EE                <1>       out    dx, al ; Write to attribute addr register
10230 00004067 66BACE03          <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
10231 0000406B 66B80605          <1>       mov    ax, 0506h ; Misc. register, graph, mm 1
10232 0000406F 66EF              <1>       out    dx, ax
10233 00004071 66BAC403          <1>       mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
10234 00004075 66B8020F          <1>       mov    ax, 0F02h ; Map mask register, all planes
10235 00004079 66EF              <1>       out    dx, ax
10236                            <1>
10237                            <1>       ; settings for >= 8bpp
10238                            <1>
10239                            <1>       ;mov   dx, 1CEh ; VBE_DISPI_IOPORT_INDEX
10240                            <1>       ;mov   ax, 03h ; VBE_DISPI_INDEX_BPP
10241                            <1>       ;out   dx, ax
```

```
10242                                  <1>       ;;mov  dx, 1CFh ; VBE_DISPI_IOPORT_DATA
10243                                  <1>       ;inc   dl
10244                                  <1>       ;in    ax, dx
10245                                  <1>       ;cmp   al, 08h  ; < 8 bits per pixel
10246                                  <1>       ;jb    short vga_compat_end
10247                                  <1>
10248 0000407B 66BAD403               <1>       mov    dx, 3D4h ; VGAREG_VGA_CRTC_ADDRESS
10249 0000407F B014                   <1>       mov    al, 14h  ; Underline location register
10250 00004081 EE                     <1>       out    dx, al
10251                                  <1>       ;mov   dx, 3D5h ; VGAREG_VGA_CRTC_DATA
10252 00004082 FEC2                   <1>       inc    dl
10253 00004084 EC                     <1>       in     al, dx
10254 00004085 0C40                   <1>       or     al, 40h       ; enable double word mode
10255 00004087 EE                     <1>       out    dx, al
10256 00004088 66BADA03               <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
10257 0000408C EC                     <1>       in     al, dx  ; clear flip-flop
10258 0000408D 66BAC003               <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
10259 00004091 B010                   <1>       mov    al, 10h       ; Mode control register
10260 00004093 EE                     <1>       out    dx, al
10261                                  <1>       ;mov   dx, 3C1h ; VGAREG_ACTL_READ_DATA
10262 00004094 FEC2                   <1>       inc    dl
10263 00004096 EC                     <1>       in     al, dx
10264 00004097 0C40                   <1>       or     al, 40h  ; Pixel clock select is 1
10265                                  <1>       ;mov   dx, 3C0h ; VGAREG_ACTL_ADDRESS
10266 00004099 FECA                   <1>       dec    dl
10267 0000409B EE                     <1>       out    dx, al  ; update mode control reggister
10268 0000409C B020                   <1>       mov    al, 20h       ; select display memory as PAS
10269 0000409E EE                     <1>       out    dx, al
10270 0000409F 66BAC403               <1>       mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
10271 000040A3 B004                   <1>       mov    al, 04h      ; Memory mode register
10272 000040A5 EE                     <1>       out    dx, al
10273                                  <1>       ;mov   dx, 3C5h ; VGAREG_SEQU_DATA
10274 000040A6 FEC2                   <1>       inc    dl
10275 000040A8 EC                     <1>       in     al, dx
10276 000040A9 0C08                   <1>       or     al, 08h      ; enable chain four
10277 000040AB EE                     <1>       out    dx, al
10278 000040AC 66BACE03               <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
10279 000040B0 B005                   <1>       mov    al, 05h       ; Mode register
10280 000040B2 EE                     <1>       out    dx, al
10281                                  <1>       ;mov   dx, 3CFh ; VGAREG_GRDC_DATA
10282 000040B3 FEC2                   <1>       inc    dl
10283 000040B5 EC                     <1>       in     al, dx
10284 000040B6 249F                   <1>       and    al, 9Fh       ; clear shift register
10285 000040B8 0C40                   <1>       or     al, 40h  ; set shift register to 2
10286 000040BA EE                     <1>       out    dx, al
10287                                  <1>
10288                                  <1> vga_compat_end:
10289                                  <1>       ;pop   edx
10290                                  <1>       ;pop   eax
10291 000040BB C3                     <1>       retn
10292                                  <1>
10293                                  <1> vga_set_virt_width:
10294                                  <1>       ; 27/11/2020
10295                                  <1>       ; 25/11/2020
10296                                  <1>       ; (vbe.c, 02/01/2020, vruppert)
10297                                  <1>       ;
10298                                  <1>       ; Input:
10299                                  <1>       ;      AX = resolution (screen width)
10300                                  <1>       ;
10301                                  <1>       ; Modified registers: eax, edx
10302                                  <1>
10303                                  <1>       ;;push ebx
10304                                  <1>       ;push  edx
10305                                  <1>       ;push  eax
10306                                  <1>       ;mov   ebx, eax
10307                                  <1>       ;call  dispi_get_bpp ; bits per pixel
10308                                  <1>       ;cmp   al, 4
10309                                  <1>       ;ja    short set_width_svga  ; 8, 16, 24, 32
10310                                  <1>       ;shr   bx, 1
10311                                  <1> ;set_width_svga:
10312                                  <1>       ;shr   bx, 3
10313                                  <1>       ;mov   eax, [esp]
10314 000040BC 66C1E803               <1>       shr    ax, 3  ; / 8, bytes per row
10315 000040C0 66BAD403               <1>       mov    dx, 3D4h ; VGAREG_VGA_CRTC_ADDRESS
10316                                  <1>       ;mov   ah, bl ;
10317 000040C4 88C4                   <1>       mov    ah, al ; width in bytes
10318 000040C6 B013                   <1>       mov    al, 13h       ; offset register
10319 000040C8 66EF                   <1>       out    dx, ax ; index (3D4h) and data (3D5h)
10320                                  <1>       ;pop   eax
10321                                  <1>       ;pop   edx
10322                                  <1>       ;;pop  ebx
10323 000040CA C3                     <1>       retn
10324                                  <1>
10325                                  <1> ; 24/11/2020
10326                                  <1>
10327                                  <1> struc bmi ; BOCHS/PLEX86 MODE INFO structure/table
10328 00000000 <res 00000002>          <1>  .mode:        resw 1
10329 00000002 <res 00000002>          <1>  .width:  resw 1
10330 00000004 <res 00000002>          <1>  .height: resw 1
10331 00000006 <res 00000002>          <1>  .depth:  resw 1
10332                                  <1>  .size:
10333                                  <1> endstruc
10334                                  <1>
10335                                  <1> ; 24/11/2020
10336                                  <1> struc MODEINFO
10337 00000000 <res 00000002>          <1>  .mode:                   resw 1  ; 1XXh
10338 00000002 <res 00000002>          <1>  .ModeAttributes:  resw 1
10339 00000004 <res 00000001>          <1>  .WinAAttributes:  resb 1
10340 00000005 <res 00000001>          <1>  .WinBAttributes:  resb 1 ; = 0
10341 00000006 <res 00000002>          <1>  .WinGranularity:  resw 1
10342 00000008 <res 00000002>          <1>  .WinSize:         resw 1
10343 0000000A <res 00000002>          <1>  .WinASegment:            resw 1
10344 0000000C <res 00000002>          <1>  .WinBSegment:            resw 1 ; = 0
10345 0000000E <res 00000004>          <1>  .WinFuncPtr:             resd 1 ; = 0
10346 00000012 <res 00000002>          <1>  .BytesPerScanLine:       resw 1
```

```
10347 00000014 <res 00000002>      <1>  .XResolution:           resw 1
10348 00000016 <res 00000002>      <1>  .YResolution:           resw 1
10349 00000018 <res 00000001>      <1>  .XCharSize:       resb 1
10350 00000019 <res 00000001>      <1>  .YCharSize:       resb 1
10351 0000001A <res 00000001>      <1>  .NumberOfPlanes:  resb 1
10352 0000001B <res 00000001>      <1>  .BitsPerPixel:          resb 1
10353 0000001C <res 00000001>      <1>  .NumberOfBanks:   resb 1
10354 0000001D <res 00000001>      <1>  .MemoryModel:           resb 1
10355 0000001E <res 00000001>      <1>  .BankSize:        resb 1 ; = 0
10356 0000001F <res 00000001>      <1>  .NumberOfImagePages:    resb 1
10357 00000020 <res 00000001>      <1>  .Reserved_page:   resb 1 ; = 0
10358 00000021 <res 00000001>      <1>  .RedMaskSize:           resb 1
10359 00000022 <res 00000001>      <1>  .RedFieldPosition:      resb 1
10360 00000023 <res 00000001>      <1>  .GreenMaskSize:   resb 1
10361 00000024 <res 00000001>      <1>  .GreenFieldPosition:    resb 1
10362 00000025 <res 00000001>      <1>  .BlueMaskSize:          resb 1
10363 00000026 <res 00000001>      <1>  .BlueFieldPosition:     resb 1
10364 00000027 <res 00000001>      <1>  .RsvdMaskSize:          resb 1
10365 00000028 <res 00000001>      <1>  .RsvdFieldPosition:     resb 1
10366 00000029 <res 00000001>      <1>  .DirectColorModeInfo:   resb 1
10367 0000002A <res 00000004>      <1>  .PhysBasePtr:           resd 1
10368 0000002E <res 00000004>      <1>  .OffScreenMemOffset:    resd 1 ; = 0
10369 00000032 <res 00000002>      <1>  .OffScreenMemSize:      resw 1 ; = 0
10370 00000034 <res 00000002>      <1>  .LinBytesPerScanLine:   resw 1
10371 00000036 <res 00000001>      <1>  .BnkNumberOfPages:      resb 1
10372 00000037 <res 00000001>      <1>  .LinNumberOfPages:      resb 1
10373 00000038 <res 00000001>      <1>  .LinRedMaskSize:  resb 1
10374 00000039 <res 00000001>      <1>  .LinRedFieldPosition1: resb 1
10375 0000003A <res 00000001>      <1>  .LinGreenMaskSize1:     resb 1
10376 0000003B <res 00000001>      <1>  .LinGreenFieldPosition:resb 1
10377 0000003C <res 00000001>      <1>  .LinBlueMaskSize:       resb 1
10378 0000003D <res 00000001>      <1>  .LinBlueFieldPosition:  resb 1
10379 0000003E <res 00000001>      <1>  .LinRsvdMaskSize:       resb 1
10380 0000003F <res 00000001>      <1>  .LinRsvdFieldPosition:  resb 1
10381 00000040 <res 00000004>      <1>  .MaxPixelClock:   resd 1 ; = 0
10382                              <1> .size:
10383                              <1> endstruc
10384                              <1>
10385                              <1> ; 10/12/2020
10386                              <1> struc LFBINFO
10387 00000000 <res 00000002>      <1>  .mode:                  resw 1  ; 1XXh
10388 00000002 <res 00000004>      <1>  .LFB_addr:        resd 1
10389 00000006 <res 00000004>      <1>  .LFB_size:        resd 1
10390 0000000A <res 00000002>      <1>  .X_res:           resw 1
10391 0000000C <res 00000002>      <1>  .Y_res:           resw 1
10392 0000000E <res 00000001>      <1>  .bpp:                   resb 1
10393 0000000F <res 00000001>      <1>  .reserved:        resb 1
10394                              <1> .size:      ; 16 bytes
10395                              <1> endstruc
10396                              <1>
10397                              <1> set_mode_info_list:
10398                              <1>      ; 14/12/2020
10399                              <1>      ; 11/12/2020
10400                              <1>      ; 24/11/2020
10401                              <1>      ; (vbetables-gen.c)
10402                              <1>      ; Input:
10403                              <1>      ;     BX = VBE mode (including bochs special modes)
10404                              <1>      ; Output:
10405                              <1>      ;     ;;EAX = MODE_INFO_LIST address
10406                              <1>      ;     EAX = 0 ; 11/12/2020
10407                              <1>      ;     ESI = MODE_INFO_LIST address ; 11/12/2020
10408                              <1>      ;     (if mode is not found, ESI = 0)
10409                              <1>      ;
10410                              <1>      ; Modified registers: eax, ebx, ecx, edx, esi, edi
10411                              <1>
10412 000040CB BE[56730000]       <1>      mov   esi, b_vbe_modes ; bochs mode info base table
10413 000040D0 BF[3A120300]       <1>      mov   edi, MODE_INFO_LIST ; mode info list (4F01h)
10414                              <1> sml_0:
10415 000040D5 66AD               <1>      lodsw
10416 000040D7 6639D8             <1>      cmp   ax, bx ; is mode number same ?
10417 000040DA 7410               <1>      je    short sml_1 ; yes
10418 000040DC AD                 <1>      lodsd
10419 000040DD 66AD               <1>      lodsw
10420 000040DF 81FE[16740000]     <1>      cmp   esi, end_of_b_vbe_modes
10421 000040E5 72EE               <1>      jb    short sml_0
10422                              <1>      ; not found
10423 000040E7 31C0               <1>      xor   eax, eax ; 0
10424                              <1>      ; 11/12/2020
10425 000040E9 31F6               <1>      xor   esi, esi
10426 000040EB C3                 <1>      retn
10427                              <1> sml_1:
10428 000040EC 66AB               <1>      stosw  ; mode
10429 000040EE AD                 <1>      lodsd  ; width, height
10430                              <1>      ; 14/12/2020
10431 000040EF 89C1               <1>      mov   ecx, eax
10432 000040F1 50                 <1>      push  eax ; ***
10433 000040F2 29C0               <1>      sub   eax, eax ; clear high word of eax
10434 000040F4 66AD               <1>      lodsw ; depth
10435 000040F6 50                 <1>      push  eax ; **
10436                              <1>
10437                              <1>      ;add  al, 7 ; only for 15 bit colors (not used here)
10438 000040F7 C0E803             <1>      shr   al, 3 ; / 8
10439                              <1>      ; 14/12/2020
10440 000040FA 66F7E1             <1>      mul   cx  ; pitch = width * ((depth+7)/8)
10441                              <1>            ; ax = pitch
10442 000040FD 50                 <1>      push  eax ; * ; high word of eax = 0
10443 000040FE C1E910             <1>      shr   ecx, 16
10444                              <1>      ;mul  cx
10445                              <1>      ;mov  cx, ax
10446 00004101 31D2               <1>      xor   edx, edx  ; clear high word of edx
10447 00004103 F7E1               <1>      mul   ecx ; height * pitch
10448 00004105 89C1               <1>      mov   ecx, eax
10449 00004107 B800000001         <1>      mov   eax, VBE_DISPI_TOTAL_VIDEO_MEMORY_MB * 1024 * 1024
10450 0000410C F7F1               <1>      div   ecx
10451                              <1>            ; eax = pages = vram_size / (height*pitch)
```

```
10452                                  <1>
10453                                  <1>       ;mov   cx, ax
10454 0000410E 89C1                     <1>       mov    ecx, eax ; pages
10455                                  <1>
10456 00004110 66B89B00                  <1>       mov    ax, MODE_ATTRIBUTES
10457 00004114 66AB                      <1>       stosw  ; ModeAttributes
10458 00004116 B007                      <1>       mov    al, WINA_ATTRIBUTES
10459 00004118 AA                        <1>       stosb  ; WinAAttributes
10460 00004119 30C0                      <1>       xor    al, al ; WinBAttributes = 0
10461 0000411B AA                        <1>       stosb
10462 0000411C 66B84000                  <1>       mov    ax, VBE_DISPI_BANK_SIZE_KB
10463 00004120 66AB                      <1>       stosw  ; WinGranularity
10464 00004122 66AB                      <1>       stosw  ; WinSize
10465 00004124 66B800A0                  <1>       mov    ax, VGAMEM_GRAPH
10466 00004128 66AB                      <1>       stosw  ; WinASegment
10467 0000412A 29C0                      <1>       sub    eax, eax
10468 0000412C 66AB                      <1>       stosw  ; WinBSegment = 0
10469 0000412E AB                        <1>       stosd  ; WinFuncPtr = 0
10470                                  <1>
10471 0000412F 58                        <1>       pop    eax ; * ; pitch
10472 00004130 89C3                      <1>       mov    ebx, eax  ; high word of ebx = 0 ; 14/12/2020
10473 00004132 66AB                      <1>       stosw  ; BytesPerScanLine
10474                                  <1>
10475 00004134 5A                        <1>       pop    edx ; ** ; depth (bits per pixel)
10476 00004135 58                        <1>       pop    eax ; *** width, height
10477                                  <1>
10478                                  <1>       ; // Mandatory information for VBE 1.2 and above
10479                                  <1>
10480 00004136 66AB                      <1>       stosw  ; XResolution (width)
10481 00004138 C1E810                    <1>       shr    eax, 16
10482 0000413B 50                        <1>       push   eax ; **** height
10483 0000413C 66AB                      <1>       stosw  ; YResolution (height)
10484 0000413E B008                      <1>       mov    al, 8
10485 00004140 AA                        <1>       stosb  ; XCharSize  ; char width
10486 00004141 B010                      <1>       mov    al, 16
10487 00004143 AA                        <1>       stosb  ; YCharSize  ; char height
10488 00004144 B001                      <1>       mov    al, 1
10489 00004146 AA                        <1>       stosb  ; NumberOfPlanes
10490                                  <1>       ;movzx eax, dl
10491 00004147 88D0                      <1>       mov    al, dl ; eax <= 32
10492 00004149 AA                        <1>       stosb  ; BitsPerPixel
10493                                  <1>       ; Number of banks = (height * pitch + 65535) / 65536
10494 0000414A 58                        <1>       pop    eax ; **** ; height
10495                                  <1>       ; 14/12/2020
10496 0000414B 52                        <1>       push   edx ; ***** ; depth ; edx <= 32
10497 0000414C F7E3                      <1>       mul    ebx  ; pitch (ebx) * height (eax)
10498                                  <1>       ;mov   edx, [esp] ; *****
10499                                  <1>       ;mov   dl, [esp] ; *****
10500 0000414E 05FFFF0000                <1>       add    eax, 65535
10501 00004153 C1E810                    <1>       shr    eax, 16 ; / 65536 ; <= 127 ; 14/12/2020
10502 00004156 AA                        <1>       stosb  ; NumberOfBanks
10503                                  <1>       ; 14/12/2020
10504                                  <1>       ;cmp   dl, 8 ; 8 bits per pixel
10505 00004157 803C2408                  <1>       cmp    byte [esp], 8
10506 0000415B 7704                      <1>       ja     short sml_2
10507 0000415D B004                      <1>       mov    al, VBE_MEMORYMODEL_PACKED_PIXEL
10508 0000415F EB02                      <1>       jmp    short sml_3
10509                                  <1> sml_2:
10510                                  <1>       ; 16, 24, 32 bts per pixel
10511 00004161 B006                      <1>       mov    al, VBE_MEMORYMODEL_DIRECT_COLOR
10512                                  <1> sml_3:
10513 00004163 AA                        <1>       stosb
10514 00004164 30C0                      <1>       xor    al, al ; 0
10515 00004166 AA                        <1>       stosb  ; BankSize = 0
10516 00004167 49                        <1>       dec    ecx ; pages - 1
10517                                  <1>       ; NumberOfImagePages = 262 for 320x200x8 mode
10518                                  <1>       ;;mov  ax, 255
10519                                  <1>       ; 14/12/2020
10520                                  <1>       ;mov   al, 255
10521 00004168 FEC8                      <1>       dec    al ; 255
10522 0000416A 39C1                      <1>       cmp    ecx, eax ; ecx <= 261, eax = 255
10523                                  <1>       ;cmp   cx, ax
10524 0000416C 7302                      <1>       jnb    short sml_4
10525 0000416E 88C8                      <1>       mov    al, cl
10526                                  <1> sml_4:
10527 00004170 AA                        <1>       stosb  ; NumberOfImagePages (1 byte)
10528 00004171 28C0                      <1>       sub    al, al
10529 00004173 AA                        <1>       stosb  ; Reserved_page = 0
10530 00004174 58                        <1>       pop    eax ; ***** ; depth
10531 00004175 88C1                      <1>       mov    cl, al
10532                                  <1>       ; eax < = 32
10533 00004177 2C08                      <1>       sub    al, 8 ; 8->0, 16->8, 24->16, 32->24
10534 00004179 BE[16740000]              <1>       mov    esi, direct_color_fields
10535 0000417E 01C6                      <1>       add    esi, eax
10536 00004180 56                        <1>       push   esi ; ******
10537 00004181 AD                        <1>       lodsd  ; RedMaskSize (AL), RedFieldPosition (AH)
10538                                  <1>              ; GreenMaskSize (16), GreenFieldPosition (24)
10539 00004182 AB                        <1>       stosd
10540 00004183 AD                        <1>       lodsd  ; BlueMaskSize (AL), BlueFieldPosition (AH)
10541                                  <1>              ; RsvdMaskSize (16), RsvdFieldPosition (24)
10542 00004184 AB                        <1>       stosd
10543 00004185 5E                        <1>       pop    esi ; ******
10544                                  <1>
10545 00004186 30C0                      <1>       xor    al, al ; 0
10546 00004188 80F920                    <1>       cmp    cl, 32
10547 0000418B 7202                      <1>       jb     short sml_5
10548 0000418D B002                      <1>       mov    al, VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE
10549                                  <1> sml_5:
10550 0000418F AA                        <1>       stosb  ; DirectColorModeInfo
10551                                  <1>
10552                                  <1>       ; // Mandatory information for VBE 2.0 and above
10553                                  <1>
10554 00004190 B8000000E0                <1>       mov    eax, VBE_DISPI_LFB_PHYSICAL_ADDRESS
10555 00004195 AB                        <1>       stosd  ; PhysBasePtr
10556 00004196 29C0                      <1>       sub    eax, eax
```

```
10557 00004198 AB                <1>       stosd  ; OffScreenMemOffset = 0
10558 00004199 66AB              <1>       stosw  ; OffScreenMemSize = 0
10559                            <1>
10560                            <1>       ;// Mandatory information for VBE 3.0 and above
10561                            <1>
10562                            <1>       ; ebx = pitch
10563 0000419B 6689D8           <1>       mov    ax, bx
10564                            <1>       ;stosw
10565                            <1>
10566                            <1>       ;xor   al, al
10567                            <1>       ;stosb ; BnkNumberOfPages = 0
10568                            <1>       ;stosb ; LinNumberOfPages = 0
10569                            <1>
10570 0000419E AB               <1>       stosd  ; pitch (word), 0 (byte), 0 (byte)
10571                            <1>
10572 0000419F AD               <1>       lodsd  ; LinRedMaskSize (AL), LinRedFieldPosition (AH)
10573                            <1>              ; LinGreenMaskSize (16), LinGreenFieldPosition (24)
10574 000041A0 AB               <1>       stosd
10575 000041A1 AD               <1>       lodsd  ; LinBlueMaskSize (AL), LinBlueFieldPosition (AH)
10576                            <1>              ; LinRsvdMaskSize (16), LinRsvdFieldPosition (24)
10577 000041A2 AB               <1>       stosd
10578                            <1>
10579 000041A3 29C0             <1>       sub    eax, eax
10580 000041A5 AB               <1>       stosd  ; MaxPixelClock = 0
10581                            <1>
10582                            <1>       ;mov   eax, MODE_INFO_LIST
10583                            <1>       ; 11/12/2020
10584 000041A6 BE[3A120300]      <1>       mov    esi, MODE_INFO_LIST
10585                            <1>
10586 000041AB C3                <1>       retn
10587                            <1>
10588                            <1> ; end of set_mode_info_list ; edi = set_mode_info_list + 68
10589                            <1>
10590                            <1> pci_get_lfb_addr:
10591                            <1>       ; 11/12/2020
10592                            <1>       ; Get linear frame buffer base from PCI
10593                            <1>       ; (vgabios.c, 02/01/2020, vruppert)
10594                            <1>       ;
10595                            <1>       ; Input:
10596                            <1>       ;      ax = PCI device vendor id
10597                            <1>       ; Output:
10598                            <1>       ;      ax  = LFB address (high 16 bit) (zf=0)
10599                            <1>       ;      eax = 0 -> not found (error) (zf=1)
10600                            <1>       ;
10601                            <1>       ; Modified registers: eax
10602                            <1>
10603 000041AC 53               <1>       push   ebx
10604 000041AD 51               <1>       push   ecx
10605 000041AE 52               <1>       push   edx
10606                            <1>       ;
10607 000041AF 89C3             <1>       mov    ebx, eax
10608 000041B1 31C9             <1>       xor    ecx, ecx
10609 000041B3 28D2             <1>       sub    dl, dl ; mov dl, 0
10610 000041B5 E842000000       <1>       call   pci_read_reg
10611 000041BA 6683F8FF         <1>       cmp    ax, 0FFFFh
10612 000041BE 7417             <1>       je     short pci_get_lfb_addr_fail
10613                            <1> pci_get_lfb_addr_next_dev:
10614 000041C0 28D2             <1>       sub    dl, dl ; mov dl, 0
10615 000041C2 E835000000       <1>       call   pci_read_reg
10616 000041C7 6639D8           <1>       cmp    ax, bx ; check vendor
10617 000041CA 740F             <1>       je     short pci_get_lfb_addr_found
10618 000041CC 6683C108         <1>       add    cx, 08h
10619 000041D0 6681F90002       <1>       cmp    cx, 200h ; search bus 0 and 1
10620 000041D5 72E9             <1>       jb     short pci_get_lfb_addr_next_dev
10621                            <1> pci_get_lfb_addr_fail:
10622 000041D7 31C0             <1>       xor    eax, eax ; no LFB
10623                            <1>       ; zf = 1
10624 000041D9 EB1D             <1>       jmp    short pci_get_lfb_addr_return
10625                            <1> pci_get_lfb_addr_found:
10626 000041DB B210             <1>       mov    dl, 10h      ; I/O space 0
10627 000041DD E81A000000       <1>       call   pci_read_reg
10628 000041E2 66A9F1FF         <1>       test   ax, 0FFF1h
10629 000041E6 740D             <1>       jz     short pci_get_lfb_addr_success
10630 000041E8 B214             <1>       mov    dl, 14h ; I/O space 1
10631 000041EA E80D000000       <1>       call   pci_read_reg
10632 000041EF 66A9F1FF         <1>       test   ax, 0FFF1h
10633 000041F3 75E2             <1>       jnz    short pci_get_lfb_addr_fail
10634                            <1> pci_get_lfb_addr_success:
10635 000041F5 C1E810           <1>       shr    eax, 16 ; LFB address (hw)
10636                            <1>       ; zf = 0
10637                            <1> pci_get_lfb_addr_return:
10638 000041F8 5A               <1>       pop    edx
10639 000041F9 59               <1>       pop    ecx
10640 000041FA 5B               <1>       pop    ebx
10641 000041FB C3               <1>       retn
10642                            <1>
10643                            <1> pci_read_reg:
10644                            <1>       ; 11/12/2020
10645                            <1>       ; Read PCI register
10646                            <1>       ; (vgabios.c, 02/01/2020, vruppert)
10647                            <1>       ;
10648                            <1>       ; Input:
10649                            <1>       ;      cx = device/function
10650                            <1>       ;      dl = register
10651                            <1>       ; Output:
10652                            <1>       ;      eax = value
10653                            <1>       ;
10654                            <1>       ; Modified registers: eax, edx
10655                            <1>
10656 000041FC B800008000       <1>       mov    eax, 00800000h
10657 00004201 6689C8           <1>       mov    ax, cx
10658 00004204 C1E008           <1>       shl    eax, 8
10659 00004207 88D0             <1>       mov    al, dl
10660 00004209 66BAF80C         <1>       mov    dx, 0CF8h
10661 0000420D EF               <1>       out    dx, eax
```

```
10662 0000420E 80C204              <1>       add    dl, 4 ; mov dx, 0CFCh
10663 00004211 ED                  <1>       in     eax, dx
10664 00004212 C3                  <1>       retn
10665                              <1>
10666                              <1> %endif
10667                              <1>
10668                              <1> ; -----------
10669                              <1>
10670                              <1> %if 0
10671                              <1>
10672                              <1> mode_info_find_mode:
10673                              <1>       ; 25/11/2020
10674                              <1>       ; Input:
10675                              <1>       ;      bx = VESA VBE2 video mode (+ bochs extensions)
10676                              <1>       ; Output:
10677                              <1>       ;      esi = mode info address (for BX input)
10678                              <1>       ;      esi = 0 -> not found
10679                              <1>       ;
10680                              <1>       ; Modified registers: eax, esi
10681                              <1>
10682                              <1>       xor    eax, eax
10683                              <1>       mov    esi, MODE_INFO_LIST
10684                              <1> mifm_1:
10685                              <1>       mov    ax, [esi]
10686                              <1>       cmp    ax, bx
10687                              <1>       je     short mifm_2
10688                              <1>       add    esi, MODEINFO.size ; add esi, 68
10689                              <1>       cmp    esi, VBE_VESA_MODE_END_OF_LIST
10690                              <1>       jb     short mifm_1
10691                              <1>       ; not found
10692                              <1>       sub    esi, esi ; 0
10693                              <1> mifm_2
10694                              <1>       retn
10695                              <1>
10696                              <1> dispi_get_bpp:
10697                              <1>       ; 28/11/2020
10698                              <1>       ; Input:
10699                              <1>       ;      none
10700                              <1>       ; Output:
10701                              <1>       ;      al = Bits per pixel
10702                              <1>       ;           (8,16,24,32)
10703                              <1>       ;      ah = Bytes per pixel
10704                              <1>       ;           (1,2,3,4)
10705                              <1>       ;
10706                              <1>       ; Modified registers: none
10707                              <1>
10708                              <1>       ;push  edx
10709                              <1>       ;mov   dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
10710                              <1>       ;mov   ax, 03h        ; VBE_DISPI_INDEX_BPP
10711                              <1>       ;out   dx, ax
10712                              <1>       ;;mov  dx, 01CFh ; VBE_DISPI_IOPORT_DATA
10713                              <1>       ;;mov  dl, 0CFh
10714                              <1>       ;inc   dl
10715                              <1>       ;in    ax, dx
10716                              <1>       ;mov   ah, al
10717                              <1>       ;shr   ah, 3 ; / 8
10718                              <1>       ;;test al, 7 ; 15 bit graphics mode
10719                              <1>       ;;jz   short get_bpp_noinc
10720                              <1>       ;;inc  ah
10721                              <1> ;;get_bpp_noinc:
10722                              <1>       ;pop   edx
10723                              <1>       ;retn
10724                              <1>
10725                              <1>       ; 28/11/2020
10726                              <1>       ; Modified registers: edx
10727                              <1>       ;push  edx
10728                              <1>       mov    dx, 03h ; VBE_DISPI_INDEX_BPP
10729                              <1>       call   dispi_get_parms
10730                              <1>       ;pop   edx
10731                              <1>       ;retn
10732                              <1>       mov    ah, al
10733                              <1>       shr    ah, 3 ; / 8
10734                              <1>       ;test  al, 7 ; 15 bit graphips mode
10735                              <1>       ;jz    short get_bpp_noinc
10736                              <1>       ;inc   ah
10737                              <1> ;get_bpp_noinc:
10738                              <1>       ;pop   edx
10739                              <1>       retn
10740                              <1>
10741                              <1> restore_vesa_video_state:
10742                              <1>       ; 14/01/2021
10743                              <1>       ; 06/12/2020
10744                              <1>       ; Input:
10745                              <1>       ;      [vbe3stbufsize] <= 32 ; <= 32*64 bytes
10746                              <1>       ; Output:
10747                              <1>       ;      AX = 004Fh (successed)
10748                              <1>       ;
10749                              <1>       ;      eax = 0 -> buffer size problem
10750                              <1>       ;      eax > 0 and ax <> 004Fh -> failed
10751                              <1>       ;
10752                              <1>       ; Modified regs: eax, ebx, ecx, edx, esi, edi
10753                              <1>
10754                              <1>       ;movzx ecx, word [vbe3stbufsize]
10755                              <1>       ;cmp   cx, 32 ; 32 * 64 bytes
10756                              <1>       ;ja    short r_v_b_s_fail
10757                              <1>
10758                              <1>       movzx  ecx, byte [vbe3stbufsize]; <=32
10759                              <1>       shl    cx, 4 ; dword count for movsd
10760                              <1>       mov    edi, VBE3SAVERESTOREBLOCK ; destination
10761                              <1>                                 ; (vbe3 pmi buff)
10762                              <1>       mov    esi, VBE3VIDEOSTATE ; source (kernel buff)
10763                              <1>       rep    movsd
10764                              <1>
10765                              <1>       mov    ax, 4F04h
10766                              <1>       mov    dl, 02h ; restore
```

```
10767                                  <1>       ;mov   cx, 0Fh
10768                                  <1>       mov    cl, 0Fh
10769                                  <1>       xor    ebx, ebx ; points to VBE3SAVERESTOREBLOCK
10770                                  <1>       jmp    short int10h_32bit_pmi
10771                                  <1>
10772                                  <1> ;s_v_b_s_fail:
10773                                  <1> ;r_v_b_s_fail:
10774                                  <1> ;      xor    eax, eax
10775                                  <1> ;      retn
10776                                  <1>
10777                                  <1> save_vesa_video_state:
10778                                  <1>       ; 14/01/2021
10779                                  <1>       ; 06/12/2020
10780                                  <1>       ; Input:
10781                                  <1>       ;      [vbe3stbufsize] <= 32 ; <= 32*64 bytes
10782                                  <1>       ; Output:
10783                                  <1>       ;      AX = 004Fh (successed)
10784                                  <1>       ;
10785                                  <1>       ;      eax = 0 -> buffer size problem
10786                                  <1>       ;      eax > 0 and ax <> 004Fh -> failed
10787                                  <1>       ;
10788                                  <1>       ; Modified regs: eax, ebx, ecx, edx, esi, edi
10789                                  <1>
10790                                  <1>       ;cmp   word [vbe33stbufsize], 32
10791                                  <1>       ;                    ; 32 * 64 bytes
10792                                  <1>       ;ja    short s_v_b_s_fail
10793                                  <1>
10794                                  <1>       mov    ax, 4F04h
10795                                  <1>       mov    dl, 01h ; save
10796                                  <1>       ;mov   cx, 0Fh
10797                                  <1>       mov    cl, 0Fh
10798                                  <1>       xor    ebx, ebx ; points to VBE3SAVERESTOREBLOCK
10799                                  <1>
10800                                  <1>       call   int10h_32bit_pmi
10801                                  <1>
10802                                  <1>       movzx  ecx, byte [vbe3stbufsize]; <=32
10803                                  <1>       shl    cx, 4 ; dword count for movsd
10804                                  <1>       mov    esi, VBE3SAVERESTOREBLOCK ; destination
10805                                  <1>                                ; (vbe3 pmi buff)
10806                                  <1>       mov    edi, VBE3VIDEOSTATE ; source (kernel buff)
10807                                  <1>       rep    movsd
10808                                  <1>       retn
10809                                  <1>
10810                                  <1> dispi_set_bank_farcall:
10811                                  <1>       ; 11/12/2020
10812                                  <1>       ; (This may be 'sysvideo' function, later)
10813                                  <1>       ;
10814                                  <1>       ; Input:
10815                                  <1>       ;      bx = 0000h, set bank number
10816                                  <1>       ;         = 0100h, get bank number
10817                                  <1>       ;      dx = bank number (if bx = 0)
10818                                  <1>       ; Output:
10819                                  <1>       ;      dx = bank number
10820                                  <1>
10821                                  <1>       cmp    bx, 0100h
10822                                  <1>       je     short dispi_set_bank_farcall_get
10823                                  <1>       or     bx, bx
10824                                  <1>       jnz    dispi_set_bank_farcall_error
10825                                  <1>       mov    ax, dx
10826                                  <1>       push   dx
10827                                  <1>       push   ax
10828                                  <1>       mov    ax, VBE_DISPI_INDEX_BANK
10829                                  <1>       mov    dx, VBE_DISPI_IOPORT_INDEX
10830                                  <1>       out    dx, ax
10831                                  <1>       pop    ax
10832                                  <1>       mov    dx, VBE_DISPI_IOPORT_DATA
10833                                  <1>       out    dx, ax
10834                                  <1>       in     ax, dx
10835                                  <1>       pop    dx
10836                                  <1>       cmp    dx, ax
10837                                  <1>       jne    short dispi_set_bank_farcall_error
10838                                  <1>       mov    ax, 004Fh
10839                                  <1>       retn   ; retf for real mode far call
10840                                  <1> dispi_set_bank_farcall_get:
10841                                  <1>       mov    ax, VBE_DISPI_INDEX_BANK
10842                                  <1>       mov    dx, VBE_DISPI_IOPORT_INDEX
10843                                  <1>       out    dx, ax
10844                                  <1>       mov    dx, VBE_DISPI_IOPORT_DATA
10845                                  <1>       in     ax, dx
10846                                  <1>       mov    dx, ax
10847                                  <1>       retn   ; retf for real mode far call
10848                                  <1> dispi_set_bank_farcall_error:
10849                                  <1>       mov    ax, 014Fh
10850                                  <1>       retn   ; retf for real mode far call
10851                                  <1>
10852                                  <1> %endif
10853                                  <1>
10854                                  <1> ; % include 'vidata.s' ; VIDEO DATA
10855                                  <1>
10856                                  <1> ; /// End Of VIDEO FUNCTIONS ///
 2655
 2656                                      setup_rtc_int:
 2657                                      ; source: http://wiki.osdev.org/RTC
 2658 00004213 FA                         cli           ; disable interrupts
 2659                                     ; default int frequency is 1024 Hz (Lower 4 bits of register A is 0110b or 6)
 2660                                     ; in order to change this ...
 2661                                     ; frequency  = 32768 >> (rate-1) --> 32768 >> 5 = 1024
 2662                                     ; (rate must be above 2 and not over 15)
 2663                                     ; new rate = 15 --> 32768 >> (15-1) = 2 Hz
 2664 00004214 B08A                      mov    al, 8Ah
 2665 00004216 E670                      out    70h, al ; set index to register A, disable NMI
 2666 00004218 90                        nop
 2667 00004219 E471                      in     al, 71h ; get initial value of register A
 2668 0000421B 88C4                      mov    ah, al
 2669 0000421D 80E4F0                    and    ah, 0F0h
```

```
2670 00004220 B08A                      mov    al, 8Ah
2671 00004222 E670                      out    70h, al ; reset index to register A
2672 00004224 88E0                      mov    al, ah
2673 00004226 0C0F                      or     al, 0Fh      ; new rate (0Fh -> 15)
2674 00004228 E671                      out    71h, al ; write only our rate to A. Note, rate is the bottom 4 bits.
2675                                    ; enable RTC interrupt
2676 0000422A B08B                      mov    al, 8Bh ;
2677 0000422C E670                      out    70h, al ; select register B and disable NMI
2678 0000422E 90                        nop
2679 0000422F E471                      in     al, 71h ; read the current value of register B
2680 00004231 88C4                      mov    ah, al  ;
2681 00004233 B08B                      mov    al, 8Bh ;
2682 00004235 E670                      out    70h, al ; set the index again (a read will reset the index to register B)
2683 00004237 88E0                      mov    al, ah  ;
2684 00004239 0C40                      or     al, 40h ;
2685 0000423B E671                      out    71h, al ; write the previous value ORed with 0x40. This turns on bit 6 of register B
2686 0000423D FB                        sti
2687 0000423E C3                        retn
2688
2689                                ; Write memory information
2690                                ; 29/01/2016
2691                                ; 06/11/2014
2692                                ; 14/08/2015
2693                                memory_info:
2694 0000423F A1[C4890100]                mov    eax, [memory_size] ; in pages
2695 00004244 50                          push   eax
2696 00004245 C1E00C                      shl    eax, 12                ; in bytes
2697 00004248 BB0A000000                  mov    ebx, 10
2698 0000424D 89D9                        mov    ecx, ebx        ; 10
2699 0000424F BE[494C0100]                mov    esi, mem_total_b_str
2700 00004254 E8D7000000                  call   bintdstr
2701 00004259 58                          pop    eax
2702 0000425A B107                        mov    cl, 7
2703 0000425C BE[6D4C0100]                mov    esi, mem_total_p_str
2704 00004261 E8CA000000                  call   bintdstr
2705                                      ; 14/08/2015
2706 00004266 E8E2000000                  call   calc_free_mem
2707                                      ; edx = calculated free pages
2708                                      ; ecx = 0
2709 0000426B A1[C8890100]                mov    eax, [free_pages]
2710 00004270 39D0                        cmp    eax, edx ; calculated free mem value
2711                                             ; and initial free mem value are same or not?
2712 00004272 751D                        jne    short pmim ; print mem info with '?' if not
2713 00004274 52                          push   edx ; free memory in pages
2714                                      ;mov   eax, edx
2715 00004275 C1E00C                      shl    eax, 12 ; convert page count
2716                                                    ; to byte count
2717 00004278 B10A                        mov    cl, 10
2718 0000427A BE[8D4C0100]                mov    esi, free_mem_b_str
2719 0000427F E8AC000000                  call   bintdstr
2720 00004284 58                          pop    eax
2721 00004285 B107                        mov    cl, 7
2722 00004287 BE[B14C0100]                mov    esi, free_mem_p_str
2723 0000428C E89F000000                  call   bintdstr
2724                                pmim:
2725 00004291 BE[374C0100]                mov    esi, msg_memory_info
2726                                      ;
2727 00004296 B407                        mov    ah, 07h ; Black background,
2728                                                     ; light gray forecolor
2729                                print_kmsg: ; 29/01/2016
2730 00004298 8825[EF890100]              mov    [ccolor], ah
2731                                pkmsg_loop:
2732 0000429E AC                          lodsb
2733 0000429F 08C0                        or     al, al
2734 000042A1 7410                        jz     short pkmsg_ok
2735 000042A3 56                          push   esi
2736                                      ; 13/05/2016
2737 000042A4 0FB61D[EF890100]            movzx  ebx, byte [ccolor]
2738                                                     ; Video page 0 (bh=0)
2739 000042AB E84DE0FFFF                  call   _write_tty
2740 000042B0 5E                          pop    esi
2741 000042B1 EBEB                        jmp    short pkmsg_loop
2742                                pkmsg_ok:
2743 000042B3 C3                          retn
2744
2745                                ; 19/12/2020
2746                                ; temporary
2747                                ; Write default liner frame buffer address
2748                                ;
2749                                default_lfb_info:
2750 000042B4 66A1[E30E0000]              mov    ax, [def_LFB_addr] ; high word
2751 000042BA E829000000                  call   wordtohex
2752 000042BF A3[094D0100]                mov    dword [lfb_addr_str], eax
2753 000042C4 BE[F24C0100]                mov    esi, msg_lfb_addr
2754 000042C9 B40F                        mov    ah, 0Fh ; Black background,
2755                                                     ; white forecolor
2756 000042CB EBCB                        jmp    short print_kmsg
2757
2758                                ; Convert binary number to hexadecimal string
2759                                ; 10/05/2015
2760                                ; dsectpm.s (28/02/2015)
2761                                ; Retro UNIX 386 v1 - Kernel v0.2.0.6
2762                                ; 01/12/2014
2763                                ; 25/11/2014
2764                                ;
2765                                bytetohex:
2766                                      ; INPUT ->
2767                                      ;      AL = byte (binary number)
2768                                      ; OUTPUT ->
2769                                      ;      AX = hexadecimal string
2770                                      ;
2771 000042CD 53                          push   ebx
2772 000042CE 31DB                        xor    ebx, ebx
2773 000042D0 88C3                        mov    bl, al
2774 000042D2 C0EB04                      shr    bl, 4
```

```
2775 000042D5 8A9B[1F430000]              mov   bl, [ebx+hexchrs]
2776 000042DB 86D8                        xchg  bl, al
2777 000042DD 80E30F                      and   bl, 0Fh
2778 000042E0 8AA3[1F430000]              mov   ah, [ebx+hexchrs]
2779 000042E6 5B                          pop   ebx
2780 000042E7 C3                          retn
2781
2782                                  wordtohex:
2783                                      ; INPUT ->
2784                                      ;      AX = word (binary number)
2785                                      ; OUTPUT ->
2786                                      ;      EAX = hexadecimal string
2787                                      ;
2788 000042E8 53                          push  ebx
2789 000042E9 31DB                        xor   ebx, ebx
2790 000042EB 86E0                        xchg  ah, al
2791 000042ED 6650                        push  ax
2792 000042EF 88E3                        mov   bl, ah
2793 000042F1 C0EB04                      shr   bl, 4
2794 000042F4 8A83[1F430000]              mov   al, [ebx+hexchrs]
2795 000042FA 88E3                        mov   bl, ah
2796 000042FC 80E30F                      and   bl, 0Fh
2797 000042FF 8AA3[1F430000]              mov   ah, [ebx+hexchrs]
2798 00004305 C1E010                      shl   eax, 16
2799 00004308 6658                        pop   ax
2800 0000430A 5B                          pop   ebx
2801 0000430B EBC0                        jmp   short bytetohex
2802                                      ;mov  bl, al
2803                                      ;shr  bl, 4
2804                                      ;mov  bl, [ebx+hexchrs]
2805                                      ;xchg bl, al
2806                                      ;and  bl, 0Fh
2807                                      ;mov  ah, [ebx+hexchrs]
2808                                      ;pop  ebx
2809                                      ;retn
2810
2811                                  dwordtohex:
2812                                      ; INPUT ->
2813                                      ;      EAX = dword (binary number)
2814                                      ; OUTPUT ->
2815                                      ;      EDX:EAX = hexadecimal string
2816                                      ;
2817 0000430D 50                          push  eax
2818 0000430E C1E810                      shr   eax, 16
2819 00004311 E8D2FFFFFF                  call  wordtohex
2820 00004316 89C2                        mov   edx, eax
2821 00004318 58                          pop   eax
2822 00004319 E8CAFFFFFF                  call  wordtohex
2823 0000431E C3                          retn
2824
2825                                  ; 10/05/2015
2826                                  hex_digits:
2827                                  hexchrs:
2828 0000431F 303132333435363738-         db '0123456789ABCDEF'
2828 00004328 39414243444546
2829                                  ; 19/01/2021 - VESA EDID ready flag (4Fh)
2830 0000432F 00                      edid: db 0
2831
2832                                  ; Convert binary number to decimal/numeric string
2833                                  ; 06/11/2014
2834                                  ; Temporary Code
2835                                  ;
2836
2837                                  bintdstr:
2838                                      ; EAX = binary number
2839                                      ; ESI = decimal/numeric string address
2840                                      ; EBX = divisor (10)
2841                                      ; ECX = string length (<=10)
2842 00004330 01CE                        add   esi, ecx
2843                                  btdstr0:
2844 00004332 4E                          dec   esi
2845 00004333 31D2                        xor   edx, edx
2846 00004335 F7F3                        div   ebx
2847 00004337 80C230                      add   dl, 30h
2848 0000433A 8816                        mov   [esi], dl
2849 0000433C FEC9                        dec   cl
2850 0000433E 740C                        jz    short btdstr2 ; 08/09/2016
2851 00004340 09C0                        or    eax, eax
2852 00004342 75EE                        jnz   short btdstr0
2853                                  btdstr1:
2854 00004344 4E                          dec   esi
2855 00004345 C60620                        mov    byte [esi], 20h ; blank space
2856 00004348 FEC9                        dec   cl
2857 0000434A 75F8                        jnz   short btdstr1
2858                                  btdstr2:
2859 0000434C C3                          retn
2860
2861                                  ; Calculate free memory pages on M.A.T.
2862                                  ; 06/11/2014
2863                                  ; Temporary Code
2864                                  ;
2865
2866                                  calc_free_mem:
2867 0000434D 31D2                        xor   edx, edx
2868                                      ;xor  ecx, ecx
2869 0000434F 668B0D[D8890100]            mov   cx, [mat_size] ; in pages
2870 00004356 C1E10A                      shl   ecx, 10      ; 1024 dwords per page
2871 00004359 BE00001000                  mov   esi, MEM_ALLOC_TBL
2872                                  cfm0:
2873 0000435E AD                          lodsd
2874 0000435F 51                          push  ecx
2875 00004360 B920000000                  mov   ecx, 32
2876                                  cfm1:
2877 00004365 D1E8                        shr   eax, 1
2878 00004367 7301                        jnc   short cfm2
```

```
2879 00004369 42                        inc    edx
2880                                cfm2:
2881 0000436A E2F9                      loop   cfm1
2882 0000436C 59                        pop    ecx
2883 0000436D E2EF                      loop   cfm0
2884 0000436F C3                        retn
2885
2886                                    %include 'diskio.s'  ; 07/03/2015
   1                                <1> ; ****************************************************************************
   2                                <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.2 - diskio.s
   3                                <1> ; ----------------------------------------------------------------------------
   4                                <1> ; Last Update: 30/08/2020
   5                                <1> ; ----------------------------------------------------------------------------
   6                                <1> ; Beginning: 24/01/2016
   7                                <1> ; ----------------------------------------------------------------------------
   8                                <1> ; Assembler: NASM version 2.11 (trdos386.s)
   9                                <1> ; ----------------------------------------------------------------------------
  10                                <1> ; Turkish Rational DOS
  11                                <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
  12                                <1> ;
  13                                <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
  14                                <1> ; diskio.inc (22/08/2015)
  15                                <1> ;
  16                                <1> ; Derived from 'IBM PC-XT-286' BIOS source code (1986)
  17                                <1> ; ****************************************************************************
  18                                <1>
  19                                <1> ; Retro UNIX 386 v1 Kernel - DISKIO.INC
  20                                <1> ; Last Modification: 22/08/2015
  21                                <1> ;     (Initialized Disk Parameters Data is in 'DISKDATA.INC')
  22                                <1> ;     (Uninitialized Disk Parameters Data is in 'DISKBSS.INC')
  23                                <1>
  24                                <1> ; DISK I/O SYSTEM - Erdogan Tan (Retro UNIX 386 v1 project)
  25                                <1>
  26                                <1> ; ///////// DISK I/O SYSTEM ///////////////
  27                                <1>
  28                                <1> ; 06/02/2015
  29                                <1> diskette_io:
30 00004370 F8                      <1>       clc ; 20/07/2020
31 00004371 9C                      <1>       pushfd
32 00004372 0E                      <1>       push   cs
33 00004373 E809000000              <1>       call   DISKETTE_IO_1
34 00004378 C3                      <1>       retn
35                                  <1>
36                                  <1> ;;;;;; DISKETTE I/O ;;;;;;;;;;;;;;;;;;;; 06/02/2015 ;;;
37                                  <1> ;//////////////////////////////////////////////////////
38                                  <1>
39                                  <1> ; DISKETTE I/O - Erdogan Tan (Retro UNIX 386 v1 project)
40                                  <1> ; 20/02/2015
41                                  <1> ; 06/02/2015 (unix386.s)
42                                  <1> ; 16/12/2014 - 02/01/2015 (dsectrm2.s)
43                                  <1> ;
44                                  <1> ; Code (DELAY) modifications - AWARD BIOS 1999 (ADISK.EQU, COMMON.MAC)
45                                  <1> ;
46                                  <1> ; ADISK.EQU
47                                  <1>
48                                  <1> ;----- Wait control constants
49                                  <1>
50                                  <1> ;amount of time to wait while RESET is active.
51                                  <1>
52                                  <1> WAITCPU_RESET_ON   EQU    21           ;Reset on must last at least 14us
53                                  <1>                                        ;at 250 KBS xfer rate.
54                                  <1>                                        ;see INTEL MCS, 1985, pg. 5-456
55                                  <1>
56                                  <1> WAITCPU_FOR_STATUS EQU    100          ;allow 30 microseconds for
57                                  <1>                                        ;status register to become valid
58                                  <1>                                        ;before re-reading.
59                                  <1>
60                                  <1> ;After sending a byte to NEC, status register may remain
61                                  <1> ;incorrectly set for 24 us.
62                                  <1>
63                                  <1> WAITCPU_RQM_LOW          EQU    24           ;number of loops to check for
64                                  <1>                                        ;RQM low.
65                                  <1>
66                                  <1> ; COMMON.MAC
67                                  <1> ;
68                                  <1> ;     Timing macros
69                                  <1> ;
70                                  <1>
71                                  <1> %macro             SIODELAY 0                ; SHORT IODELAY
72                                  <1>             jmp short $+2
73                                  <1> %endmacro
74                                  <1>
75                                  <1> %macro             IODELAY  0                ; NORMAL IODELAY
76                                  <1>             jmp short $+2
77                                  <1>             jmp short $+2
78                                  <1> %endmacro
79                                  <1>
80                                  <1> %macro             NEWIODELAY 0
81                                  <1>             out    0ebh,al
82                                  <1> %endmacro
83                                  <1>
84                                  <1> ; (According to) AWARD BIOS 1999 - ATORGS.ASM (dw -> equ, db -> equ)
85                                  <1> ;;; WAIT_FOR_MEM
86                                  <1> ;WAIT_FDU_INT_LO   equ    017798       ; 2.5 secs in 30 micro units.
87                                  <1> ;WAIT_FDU_INT_HI   equ    1
88                                  <1> WAIT_FDU_INT_LH           equ    83334        ; 27/02/2015 (2.5 seconds waiting)
89                                  <1> ;;; WAIT_FOR_PORT
90                                  <1> ;WAIT_FDU_SEND_LO  equ    16667        ; .5 secons in 30 us units.
91                                  <1> ;WAIT_FDU_SEND_HI  equ    0
92                                  <1> WAIT_FDU_SEND_LH   equ    16667        ; 27/02/2015
93                                  <1> ;Time to wait while waiting for each byte of NEC results = .5
94                                  <1> ;seconds.  .5 seconds = 500,000 micros.  500,000/30 = 16,667.
95                                  <1> ;WAIT_FDU_RESULTS_LO      equ    16667        ; .5 seconds in 30 micro units.
96                                  <1> ;WAIT_FDU_RESULTS_HI      equ    0
97                                  <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 ***
```

```
308                                     <1> ;
309                                     <1> ;         (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
310                                     <1> ;         (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
311                                     <1> ;                BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
312                                     <1> ;         (DH) - MAXIMUM HEAD NUMBER
313                                     <1> ;         (DL) - NUMBER OF DISKETTE DRIVES INSTALLED
314                                     <1> ;         (BH) - 0
315                                     <1> ;         (BL) - BITS 7 THRU 4 - 0
316                                     <1> ;                BITS 3 THRU 0 - VALID DRIVE TYPE VALUE IN CMOS
317                                     <1> ;         (AX) - 0
318                                     <1> ;      UNDER THE FOLLOWING CIRCUMSTANCES:
319                                     <1> ;         (1) THE DRIVE NUMBER IS INVALID,
320                                     <1> ;         (2) THE DRIVE TYPE IS UNKNOWN AND CMOS IS NOT PRESENT,
321                                     <1> ;         (3) THE DRIVE TYPE IS UNKNOWN AND CMOS IS BAD,
322                                     <1> ;         (4) OR THE DRIVE TYPE IS UNKNOWN AND THE CMOS DRIVE TYPE IS INVALID
323                                     <1> ;         THEN ES,AX,BX,CX,DH,DI=0 ; DL=NUMBER OF DRIVES.
324                                     <1> ;         IF NO DRIVES ARE PRESENT THEN: ES,AX,BX,CX,DX,DI=0.
325                                     <1> ;         @DISKETTE_STATUS = 0 AND CY IS RESET.
326                                     <1> ;--------------------------------------------------------------------------------
327                                     <1> ;     (AH)= 15H  READ DASD TYPE
328                                     <1> ;     OUTPUT REGISTERS
329                                     <1> ;     (AH) - ON RETURN IF CARRY FLAG NOT SET, OTHERWISE ERROR
330                                     <1> ;            00 - DRIVE NOT PRESENT
331                                     <1> ;            01 - DISKETTE, NO CHANGE LINE AVAILABLE
332                                     <1> ;            02 - DISKETTE, CHANGE LINE AVAILABLE
333                                     <1> ;            03 - RESERVED (FIXED DISK)
334                                     <1> ;     (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
335                                     <1> ;--------------------------------------------------------------------------------
336                                     <1> ;     (AH)= 16H  DISK CHANGE LINE STATUS
337                                     <1> ;     OUTPUT REGISTERS
338                                     <1> ;     (AH) - 00 - DISK CHANGE LINE NOT ACTIVE
339                                     <1> ;            06 - DISK CHANGE LINE ACTIVE & CARRY BIT ON
340                                     <1> ;     (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
341                                     <1> ;--------------------------------------------------------------------------------
342                                     <1> ;     (AH)= 17H  SET DASD TYPE FOR FORMAT
343                                     <1> ;     INPUT REGISTERS
344                                     <1> ;     (AL) - 00 - NOT USED
345                                     <1> ;            01 - DISKETTE 320/360K IN 360K DRIVE
346                                     <1> ;            02 - DISKETTE 360K IN 1.2M DRIVE
347                                     <1> ;            03 - DISKETTE 1.2M IN 1.2M DRIVE
348                                     <1> ;            04 - DISKETTE 720K IN 720K DRIVE
349                                     <1> ;     (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED:
350                                     <1> ;            (DO NOT USE WHEN DISKETTE ATTACH CARD USED)
351                                     <1> ;--------------------------------------------------------------------------------
352                                     <1> ;     (AH)= 18H  SET MEDIA TYPE FOR FORMAT
353                                     <1> ;     INPUT REGISTERS
354                                     <1> ;     (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM TRACKS
355                                     <1> ;     (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
356                                     <1> ;            BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
357                                     <1> ;     (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHACKED)
358                                     <1> ;     OUTPUT REGISTERS:
359                                     <1> ;     (ES:DI) - POINTER TO DRIVE PARAMETERS TABLE FOR THIS MEDIA TYPE,
360                                     <1> ;               UNCHANGED IF (AH) IS NON-ZERO
361                                     <1> ;     (AH) - 00H, CY = 0, TRACK AND SECTORS/TRACK COMBINATION IS SUPPORTED
362                                     <1> ;          - 01H, CY = 1, FUNCTION IS NOT AVAILABLE
363                                     <1> ;          - 0CH, CY = 1, TRACK AND SECTORS/TRACK COMBINATION IS NOT SUPPORTED
364                                     <1> ;          - 80H, CY = 1, TIME OUT (DISKETTE NOT PRESENT)
365                                     <1> ;--------------------------------------------------------------------------------
366                                     <1> ;     DISK CHANGE STATUS IS ONLY CHECKED WHEN A MEDIA SPECIFIED IS OTHER
367                                     <1> ;     THAN 360 KB DRIVE. IF THE DISK CHANGE LINE IS FOUND TO BE
368                                     <1> ;     ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:
369                                     <1> ;           ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.
370                                     <1> ;           IF ATTEMPT SUCCEEDS SET DASD TYPE FOR FORMAT AND RETURN DISK
371                                     <1> ;           CHANGE ERROR CODE
372                                     <1> ;           IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASD TYPE
373                                     <1> ;           TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.
374                                     <1> ;     IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASD TYPE FOR FORMAT.
375                                     <1> ;
376                                     <1> ; DATA VARIABLE -- @DISK_POINTER
377                                     <1> ;     DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
378                                     <1> ;--------------------------------------------------------------------------------
379                                     <1> ; OUTPUT FOR ALL FUNCTIONS
380                                     <1> ;     AH = STATUS OF OPERATION
381                                     <1> ;           STATUS BITS ARE DEFINED IN THE EQUATES FOR @DISKETTE_STATUS
382                                     <1> ;           VARIABLE IN THE DATA SEGMENT OF THIS MODULE
383                                     <1> ;     CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASD
384                                     <1> ;           TYPE AH=(15)).
385                                     <1> ;     CY = 1 FAILED OPERATION (AH HAS ERROR REASON)
386                                     <1> ;     FOR READ/WRITE/VERIFY
387                                     <1> ;           DS,BX,DX,CX PRESERVED
388                                     <1> ;     NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
389                                     <1> ;           ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
390                                     <1> ;           ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
391                                     <1> ;           THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
392                                     <1> ;           PROBLEM IS NOT DUE TO MOTOR START-UP.
393                                     <1> ;--------------------------------------------------------------------------------
394                                     <1> ;
395                                     <1> ; DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)
396                                     <1> ;
397                                     <1> ;   ---------------------------------------------------------------
398                                     <1> ;   |       |       |       |       |       |       |       |       |
399                                     <1> ;   |   7   |   6   |   5   |   4   |   3   |   2   |   1   |   0   |
400                                     <1> ;   |       |       |       |       |       |       |       |       |
401                                     <1> ;   ---------------------------------------------------------------
402                                     <1> ;     |       |       |       |       |       |       |
403                                     <1> ;     |       |       |       |       |     ----------------
404                                     <1> ;     |       |       |       |       |           |
405                                     <1> ;     |       |       |       |    RESERVED       |
406                                     <1> ;     |       |       |       |             PRESENT STATE
407                                     <1> ;     |       |       |       |      000: 360K IN 360K DRIVE UNESTABLISHED
408                                     <1> ;     |       |       |       |      001: 360K IN 1.2M DRIVE UNESTABLISHED
409                                     <1> ;     |       |       |       |      010: 1.2M IN 1.2M DRIVE UNESTABLISHED
410                                     <1> ;     |       |       |       |      011: 360K IN 360K DRIVE ESTABLISHED
411                                     <1> ;     |       |       |       |      100: 360K IN 1.2M DRIVE ESTABLISHED
412                                     <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> ; 30/08/2020 - TRDOS 386 v2
                                <1>
                                <1> ;;int13h: ; 16/02/2015
                                <1> ;; 16/02/2015 - 21/02/2015
                                <1> int40h:
00004379 9C                     <1>       pushfd
0000437A 0E                     <1>       push   cs
0000437B E801000000             <1>       call   DISKETTE_IO_1
00004380 C3                     <1>       retn
                                <1>
                                <1> DISKETTE_IO_1:
                                <1>
00004381 FB                     <1>       STI                         ; INTERRUPTS BACK ON
00004382 55                     <1>       PUSH   eBP                  ; USER REGISTER
00004383 57                     <1>       PUSH   eDI                  ; USER REGISTER
00004384 52                     <1>       PUSH   eDX                  ; HEAD #, DRIVE # OR USER REGISTER
00004385 53                     <1>       PUSH   eBX                  ; BUFFER OFFSET PARAMETER OR REGISTER
00004386 51                     <1>       PUSH   eCX                  ; TRACK #-SECTOR # OR USER REGISTER
00004387 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
00004389 06                     <1>       push   es ; 06/02/2015
0000438A 1E                     <1>       PUSH   DS                   ; BUFFER SEGMENT PARM OR USER REGISTER
0000438B 56                     <1>       PUSH   eSI                  ; USER REGISTERS
                                <1>       ;CALL  DDS                  ; SEGMENT OF BIOS DATA AREA TO DS
                                <1>       ;mov   cx, cs
                                <1>       ;mov   ds, cx
0000438C 66B91000               <1>       mov    cx, KDATA
00004390 8ED9                   <1>         mov     ds, cx
00004392 8EC1                   <1>         mov     es, cx
                                <1>
                                <1>       ;CMP   AH,(FNC_TAE-FNC_TAB)/2     ; CHECK FOR > LARGEST FUNCTION
00004394 80FC19                 <1>       cmp    ah,(FNC_TAE-FNC_TAB)/4  ; 18/02/2015
00004397 7202                   <1>       JB     short OK_FUNC        ; FUNCTION OK
00004399 B414                   <1>       MOV    AH,14H               ; REPLACE WITH KNOWN INVALID FUNCTION
                                <1> OK_FUNC:
0000439B 80FC01                 <1>       CMP    AH,1                 ; RESET OR STATUS ?
0000439E 760C                   <1>       JBE    short OK_DRV         ; IF RESET OR STATUS DRIVE ALWAYS OK
000043A0 80FC08                 <1>       CMP    AH,8                 ; READ DRIVE PARMS ?
000043A3 7407                   <1>       JZ     short OK_DRV         ; IF SO DRIVE CHECKED LATER
000043A5 80FA01                 <1>       CMP    DL,1                 ; DRIVES 0 AND 1 OK
000043A8 7602                   <1>       JBE    short OK_DRV         ; IF 0 OR 1 THEN JUMP
000043AA B414                   <1>       MOV    AH,14H               ; REPLACE WITH KNOWN INVALID FUNCTION
                                <1> OK_DRV:
000043AC 31C9                   <1>       xor    ecx, ecx
                                <1>       ;mov   esi, ecx ; 08/02/2015
000043AE 89CF                   <1>       mov    edi, ecx ; 08/02/2015
000043B0 88E1                   <1>       MOV    CL,AH                ; CL = FUNCTION
                                <1>       ;XOR   CH,CH                ; CX = FUNCTION
                                <1>       ;SHL   CL, 1                ; FUNCTION TIMES 2
000043B2 C0E102                 <1>       SHL    CL, 2 ; 20/02/2015   ; FUNCTION TIMES 4 (for 32 bit offset)
000043B5 BB[ED430000]           <1>       MOV    eBX,FNC_TAB          ; LOAD START OF FUNCTION TABLE
000043BA 01CB                   <1>       ADD    eBX,eCX                   ; ADD OFFSET INTO TABLE => ROUTINE
```

```
000043BC 88F4                <1>      MOV    AH,DH               ; AX = HEAD #,# OF SECTORS OR DASD TYPE
000043BE 30F6                <1>      XOR    DH,DH               ; DX = DRIVE #
000043C0 6689C6              <1>      MOV    SI,AX               ; SI = HEAD #,# OF SECTORS OR DASD TYPE
000043C3 6689D7              <1>      MOV     DI,DX                    ; DI = DRIVE #
                             <1>      ;
                             <1>      ; 11/12/2014
000043C6 8815[ED6D0000]      <1>        mov     [cfd], dl               ; current floppy drive (for 'GET_PARM')
                             <1>      ;
000043CC 8A25[488A0100]      <1>      MOV    AH, [DSKETTE_STATUS]      ; LOAD STATUS TO AH FOR STATUS FUNCTION
000043D2 C605[488A0100]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
000043D9 FF13                <1>      CALL   dWORD [eBX]        ; CALL THE REQUESTED FUNCTION
000043DB 5E                  <1>      POP    eSI                ; RESTORE ALL REGISTERS
000043DC 1F                  <1>      POP    DS
000043DD 07                  <1>      pop    es     ; 06/02/2015
000043DE 59                  <1>      POP    eCX
000043DF 5B                  <1>      POP    eBX
000043E0 5A                  <1>      POP    eDX
000043E1 5F                  <1>      POP    eDI
000043E2 89E5                <1>      MOV    eBP, eSP
000043E4 50                  <1>      PUSH   eAX
000043E5 9C                  <1>      PUSHFd
000043E6 58                  <1>      POP    eAX
                             <1>      ;MOV   [BP+6], AX
000043E7 89450C              <1>      mov    [ebp+12], eax  ; 18/02/2015, flags
000043EA 58                  <1>      POP    eAX
000043EB 5D                  <1>      POP    eBP
000043EC CF                  <1>      IRETd
                             <1>
                             <1> ;--------------------------------------------------------------------------
                             <1> ; DW --> dd (06/02/2015)
000043ED [51440000]          <1> FNC_TAB      dd     DSK_RESET           ; AH = 00H; RESET
000043F1 [CA440000]          <1>              dd     DSK_STATUS          ; AH = 01H; STATUS
000043F5 [DB440000]          <1>              dd     DSK_READ            ; AH = 02H; READ
000043F9 [EC440000]          <1>              dd     DSK_WRITE           ; AH = 03H; WRITE
000043FD [FD440000]          <1>              dd     DSK_VERF            ; AH = 04H; VERIFY
00004401 [0E450000]          <1>              dd     DSK_FORMAT          ; AH = 05H; FORMAT
00004405 [93450000]          <1>              dd     FNC_ERR                   ; AH = 06H; INVALID
00004409 [93450000]          <1>              dd     FNC_ERR                   ; AH = 07H; INVALID
0000440D [A0450000]          <1>              dd     DSK_PARMS           ; AH = 08H; READ DRIVE PARAMETERS
00004411 [93450000]          <1>              dd     FNC_ERR                   ; AH = 09H; INVALID
00004415 [93450000]          <1>              dd     FNC_ERR                   ; AH = 0AH; INVALID
00004419 [93450000]          <1>              dd     FNC_ERR                   ; AH = 0BH; INVALID
0000441D [93450000]          <1>              dd     FNC_ERR                   ; AH = 0CH; INVALID
00004421 [93450000]          <1>              dd     FNC_ERR                   ; AH = 0DH; INVALID
00004425 [93450000]          <1>              dd     FNC_ERR                   ; AH = 0EH; INVALID
00004429 [93450000]          <1>              dd     FNC_ERR                   ; AH = 0FH; INVALID
0000442D [93450000]          <1>              dd     FNC_ERR                   ; AH = 10H; INVALID
00004431 [93450000]          <1>              dd     FNC_ERR                   ; AH = 11H; INVALID
00004435 [93450000]          <1>              dd     FNC_ERR                   ; AH = 12H; INVALID
00004439 [93450000]          <1>              dd     FNC_ERR                   ; AH = 13H; INVALID
0000443D [93450000]          <1>              dd     FNC_ERR                   ; AH = 14H; INVALID
00004441 [90460000]          <1>              dd     DSK_TYPE            ; AH = 15H; READ DASD TYPE
00004445 [C0460000]          <1>              dd     DSK_CHANGE          ; AH = 16H; CHANGE STATUS
00004449 [FA460000]          <1>              dd     FORMAT_SET          ; AH = 17H; SET DASD TYPE
0000444D [7D470000]          <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:
00004451 66BAF203            <1>      MOV    DX,03F2H            ; ADAPTER CONTROL PORT
00004455 FA                  <1>      CLI                        ; NO INTERRUPTS
00004456 A0[468A0100]        <1>      MOV    AL,[MOTOR_STATUS]   ; GET DIGITAL OUTPUT REGISTER REFLECTION
0000445B 243F                <1>      AND    AL,00111111B        ; KEEP SELECTED AND MOTOR ON BITS
0000445D C0C004              <1>      ROL    AL,4                ; MOTOR VALUE TO HIGH NIBBLE
                             <1>                                 ; DRIVE SELECT TO LOW NIBBLE
00004460 0C08                <1>      OR     AL,00001000B        ; TURN ON INTERRUPT ENABLE
00004462 EE                  <1>      OUT    DX,AL               ; RESET THE ADAPTER
00004463 C605[458A0100]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
0000446A E6EB                <2>  out 0ebh,al
                             <1>
                             <1>      ; 17/12/2014
                             <1>      ; AWARD BIOS 1999 - RESETDRIVES (ADISK.ASM)
0000446C B915000000          <1>      mov    ecx, WAITCPU_RESET_ON     ; cx = 21 -- Min. 14 micro seconds !?
                             <1> wdw1:
                             <1>      NEWIODELAY   ; 27/02/2015
00004471 E6EB                <2>  out 0ebh,al
```

```
                                <1>       loop   wdw1
                                <1>       ;
00004475 0C04                   <1>       OR     AL,00000100B        ; TURN OFF RESET BIT
00004477 EE                     <1>       OUT    DX,AL               ; RESET THE ADAPTER
                                <1>       ; 16/12/2014
                                <1>       IODELAY
00004478 EB00                   <2>  jmp short $+2
0000447A EB00                   <2>  jmp short $+2
                                <1>       ;
                                <1>       ;STI                       ; ENABLE THE INTERRUPTS
0000447C E8590C0000             <1>       CALL   WAIT_INT            ; WAIT FOR THE INTERRUPT
00004481 723E                   <1>       JC     short DR_ERR        ; IF ERROR, RETURN IT
00004483 66B9C000               <1>       MOV    CX,11000000B        ; CL = EXPECTED @NEC_STATUS
                                <1> NXT_DRV:
00004487 6651                   <1>       PUSH   CX                  ; SAVE FOR CALL
00004489 B8[BF440000]           <1>       MOV    eAX, DR_POP_ERR     ; LOAD NEC_OUTPUT ERROR ADDRESS
0000448E 50                     <1>       PUSH   eAX                 ; "
0000448F B408                   <1>       MOV    AH,08H              ; SENSE INTERRUPT STATUS COMMAND
00004491 E8370B0000             <1>       CALL   NEC_OUTPUT
00004496 58                     <1>       POP    eAX                 ; THROW AWAY ERROR RETURN
00004497 E86E0C0000             <1>       CALL   RESULTS                   ; READ IN THE RESULTS
0000449C 6659                   <1>       POP    CX                  ; RESTORE AFTER CALL
0000449E 7221                   <1>       JC     short DR_ERR        ; ERROR RETURN
000044A0 3A0D[498A0100]         <1>       CMP    CL, [NEC_STATUS]    ; TEST FOR DRIVE READY TRANSITION
000044A6 7519                   <1>       JNZ    short DR_ERR        ; EVERYTHING OK
000044A8 FEC1                   <1>       INC    CL                  ; NEXT EXPECTED @NEC_STATUS
000044AA 80F9C3                 <1>       CMP    CL,11000011B        ; ALL POSSIBLE DRIVES CLEARED
000044AD 76D8                   <1>       JBE    short NXT_DRV       ; FALL THRU IF 11000100B OR >
                                <1>       ;
000044AF E886030000             <1>       CALL   SEND_SPEC           ; SEND SPECIFY COMMAND TO NEC
                                <1> RESBAC:
000044B4 E83A090000             <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
000044B9 6689F3                 <1>       MOV    BX,SI               ; GET SAVED AL TO BL
000044BC 88D8                   <1>       MOV    AL,BL               ; PUT BACK FOR RETURN
000044BE C3                     <1>       RETn
                                <1> DR_POP_ERR:
000044BF 6659                   <1>       POP    CX                  ; CLEAR STACK
                                <1> DR_ERR:
000044C1 800D[488A0100]20       <1>       OR     byte [DSKETTE_STATUS],BAD_NEC ; SET ERROR CODE
000044C8 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:
000044CA 8825[488A0100]         <1>       MOV    [DSKETTE_STATUS],AH ; PUT BACK FOR SETUP END
000044D0 E81E090000             <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
000044D5 6689F3                 <1>       MOV    BX,SI               ; GET SAVED AL TO BL
000044D8 88D8                   <1>       MOV    AL,BL               ; PUT BACK FOR RETURN
000044DA 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:
000044DB 8025[468A0100]7F       <1>       AND    byte [MOTOR_STATUS],01111111B ; INDICATE A READ OPERATION
000044E2 66B846E6               <1>       MOV    AX,0E646H           ; AX = NEC COMMAND, DMA COMMAND
000044E6 E859040000             <1>       CALL   RD_WR_VF            ; COMMON READ/WRITE/VERIFY
000044EB 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:
000044EC 66B84AC5               <1>       MOV    AX,0C54AH           ; AX = NEC COMMAND, DMA COMMAND
000044F0 800D[468A0100]80       <1>         OR     byte [MOTOR_STATUS],10000000B ; INDICATE WRITE OPERATION
000044F7 E848040000             <1>       CALL   RD_WR_VF            ; COMMON READ/WRITE/VERIFY
000044FC 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:
000044FD 8025[468A0100]7F   <1>       AND    byte [MOTOR_STATUS],01111111B ; INDICATE A READ OPERATION
00004504 66B842E6           <1>       MOV    AX,0E642H           ; AX = NEC COMMAND, DMA COMMAND
00004508 E837040000         <1>       CALL   RD_WR_VF            ; COMMON READ/WRITE/VERIFY
0000450D 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:
0000450E E870030000         <1>       CALL   XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
00004513 E86C050000         <1>       CALL   FMT_INIT            ; ESTABLISH STATE IF UNESTABLISHED
00004518 800D[468A0100]80   <1>         OR     byte [MOTOR_STATUS], 10000000B ; INDICATE WRITE OPERATION
0000451F E8B4050000         <1>       CALL   MED_CHANGE          ; CHECK MEDIA CHANGE AND RESET IF SO
00004524 725D               <1>         JC     short FM_DON            ; MEDIA CHANGED, SKIP
00004526 E80F030000         <1>       CALL   SEND_SPEC           ; SEND SPECIFY COMMAND TO NEC
0000452B E81A060000         <1>       CALL   CHK_LASTRATE        ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
00004530 7405               <1>         JZ     short FM_WR             ; YES, SKIP SPECIFY COMMAND
00004532 E8F1050000         <1>       CALL   SEND_RATE           ; SEND DATA RATE TO CONTROLLER
                        <1> FM_WR:
00004537 E8A7060000         <1>       CALL   FMTDMA_SET          ; SET UP THE DMA FOR FORMAT
0000453C 7245               <1>         JC     short FM_DON            ; RETURN WITH ERROR
0000453E B44D               <1>       MOV    AH,04DH                ; ESTABLISH THE FORMAT COMMAND
00004540 E804070000         <1>       CALL   NEC_INIT            ; INITIALIZE THE NEC
00004545 723C               <1>         JC     short FM_DON            ; ERROR - EXIT
00004547 B8[83450000]       <1>         MOV    eAX, FM_DON             ; LOAD ERROR ADDRESS
0000454C 50                 <1>       PUSH   eAX                 ; PUSH NEC_OUT ERROR RETURN
0000454D B203               <1>       MOV    DL,3                ; BYTES/SECTOR VALUE TO NEC
0000454F E873090000         <1>       CALL   GET_PARM
00004554 E8740A0000         <1>       CALL   NEC_OUTPUT
00004559 B204               <1>       MOV    DL,4                ; SECTORS/TRACK VALUE TO NEC
0000455B E867090000         <1>       CALL   GET_PARM
00004560 E8680A0000         <1>       CALL   NEC_OUTPUT
00004565 B207               <1>       MOV    DL,7                ; GAP LENGTH VALUE TO NEC
00004567 E85B090000         <1>       CALL   GET_PARM
0000456C E85C0A0000         <1>       CALL   NEC_OUTPUT
00004571 B208               <1>       MOV    DL,8                ; FILLER BYTE TO NEC
00004573 E84F090000         <1>       CALL   GET_PARM
00004578 E8500A0000         <1>       CALL   NEC_OUTPUT
0000457D 58                 <1>       POP    eAX                 ; THROW AWAY ERROR
0000457E E844070000         <1>       CALL   NEC_TERM            ; TERMINATE, RECEIVE STATUS, ETC,
                        <1> FM_DON:
00004583 E82C030000         <1>       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
00004588 E866080000         <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
0000458D 6689F3             <1>       MOV    BX,SI               ; GET SAVED AL TO BL
00004590 88D8               <1>       MOV    AL,BL               ; PUT BACK FOR RETURN
00004592 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
00004593 6689F0             <1>       MOV    AX,SI               ; RESTORE AL
00004596 B401               <1>       MOV    AH,BAD_CMD          ; SET BAD COMMAND ERROR
00004598 8825[488A0100]     <1>       MOV    [DSKETTE_STATUS],AH ; STORE IN DATA AREA
0000459E F9                 <1>       STC                        ; SET CARRY INDICATING ERROR
0000459F C3                 <1>       RETn
                        <1>
                        <1> ; 30/08/2020
                        <1> ; 29/08/2020
                        <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:
000045A0 E8DE020000     <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
000045A5 29D2           <1>       sub    edx, edx
000045A7 66A1[FE6D0000] <1>       mov     ax, [fd0_type]
000045AD 6621C0         <1>       and     ax, ax
000045B0 0F849B000000   <1>         jz      NON_DRV
000045B6 FEC2           <1>       inc     dl
000045B8 20E4           <1>       and     ah, ah
000045BA 7402           <1>       jz      short STO_DL
000045BC FEC2           <1>       inc     dl
                        <1> STO_DL:
                        <1>       ; 30/08/2020
000045BE 6639FA         <1>       cmp    dx, di
000045C1 0F868A000000   <1>       jna    NON_DRV
                        <1>       ;
                        <1>       ;MOV   [BP+4],DL           ; STORE NUMBER OF DRIVES
000045C7 895508         <1>       mov    [ebp+8], edx ; 20/02/2015
000045CA 6683FF01       <1>       CMP    DI,1                ; CHECK FOR VALID DRIVE
                        <1>       ;JA    short NON_DRV1              ; DRIVE INVALID
000045CE 0F8780000000   <1>       ja     NON_DRV1 ; 29/08/2020
                        <1>       ;MOV   BYTE [BP+5],1       ; MAXIMUM HEAD NUMBER =   1
000045D4 C6450901       <1>       mov    byte [ebp+9], 1  ; 20/02/2015
000045D8 E8E1080000     <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
000045DD 740F           <1>       JZ     short CHK_EST       ; JUMP IF SO
000045DF E82B020000     <1>       CALL   DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
000045E4 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
000045E6 8A4B04         <1>       MOV     CL, [eBX+MD.SEC_TRK]     ; GET SECTOR/TRACK
000045E9 8A6B0B         <1>         MOV     CH, [eBX+MD.MAX_TRK]     ; GET MAX. TRACK NUMBER
000045EC EB36           <1>       JMP    SHORT STO_CX        ; CMOS GOOD, USE CMOS
                        <1> CHK_EST:
000045EE 8AA7[558A0100] <1>       MOV    AH, [DSK_STATE+eDI] ; LOAD STATE FOR THIS DRIVE
000045F4 F6C410         <1>       TEST   AH,MED_DET          ; CHECK FOR ESTABLISHED STATE
000045F7 745B           <1>       JZ     short NON_DRV1              ; CMOS BAD/INVALID OR UNESTABLISHED
                        <1> USE_EST:
000045F9 80E4C0         <1>       AND    AH,RATE_MSK         ; ISOLATE STATE
000045FC 80FC80         <1>       CMP    AH,RATE_250         ; RATE 250 ?
000045FF 757B           <1>       JNE    short USE_EST2              ; NO, GO CHECK OTHER RATE
                        <1>
                        <1> ;-----       DATA RATE IS 250 KBS, TRY 360 KB TABLE FIRST
                        <1>
00004601 B001           <1>       MOV    AL,01               ; DRIVE TYPE 1 (360KB)
00004603 E807020000     <1>       CALL   DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
00004608 8A4B04         <1>         MOV     CL, [eBX+MD.SEC_TRK]     ; GET SECTOR/TRACK
0000460B 8A6B0B         <1>         MOV     CH, [eBX+MD.MAX_TRK]     ; GET MAX. TRACK NUMBER
0000460E F687[558A0100]01 <1>       TEST   byte [DSK_STATE+eDI],TRK_CAPA ; 80 TRACK ?
00004615 740D           <1>       JZ     short STO_CX        ; MUST BE 360KB DRIVE
                        <1>
                        <1> ;-----       IT IS 1.44 MB DRIVE
                        <1>
                        <1> PARM144:
00004617 B004           <1>       MOV    AL,04               ; DRIVE TYPE 4 (1.44MB)
00004619 E8F1010000     <1>       CALL   DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
0000461E 8A4B04         <1>         MOV     CL, [eBX+MD.SEC_TRK]     ; GET SECTOR/TRACK
00004621 8A6B0B         <1>         MOV     CH, [eBX+MD.MAX_TRK]     ; GET MAX. TRACK NUMBER
                        <1> STO_CX:
00004624 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>       ;
00004627 57             <1>       push   edi
00004628 8B7D04         <1>       mov    edi, [ebp+4]              ; ebx (input), user's buffer address
                        <1>       ; 29/08/2020
0000462B 09FF           <1>       or     edi, edi
0000462D 7417           <1>       jz     short no_copy_fdpt
                        <1>       ;
0000462F 0FB6C0         <1>       movzx  eax, al
00004632 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)
00004635 A3[4C960100]   <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>       ;
0000463A 89DE           <1>       mov    esi, ebx           ; floppy disk parameter table (16 bytes)
0000463C B910000000     <1>       mov    ecx, 16 ; 16 bytes
00004641 E8ACD40000     <1>         call    transfer_to_user_buffer ; trdosk6.s (16/05/2016)
                        <1> no_copy_fdpt:
00004646 5F             <1>       pop    edi
                        <1> DP_OUT:
00004647 E868020000     <1>       CALL   XLAT_OLD           ; TRANSLATE STATE TO COMPATIBLE MODE
0000464C 6631C0         <1>       XOR    AX,AX              ; CLEAR
0000464F F8             <1>       CLC
00004650 C3             <1>       RETn
                        <1>
                        <1> ;-----      NO DRIYE PRESENT HANDLER
                        <1>
                        <1> NON_DRV:
                        <1>       ;MOV   BYTE [BP+4],0      ; CLEAR NUMBER OF DRIVES
00004651 895508         <1>       mov    [ebp+8], edx ; 0 ; 20/02/2015
                        <1> NON_DRV1:
00004654 6681FF8000     <1>       CMP    DI,80H             ; CHECK FOR FIXED MEDIA TYPE REQUEST
00004659 720C           <1>       JB     short NON_DRV2            ; CONTINUE IF NOT REQUEST FALL THROUGH
                        <1>
                        <1> ;-----      FIXED DISK REQUEST FALL THROUGH ERROR
                        <1>
0000465B E854020000     <1>       CALL   XLAT_OLD           ; ELSE TRANSLATE TO COMPATIBLE MODE
00004660 6689F0         <1>       MOV    AX,SI              ; RESTORE AL
00004663 B401           <1>       MOV    AH,BAD_CMD         ; SET BAD COMMAND ERROR
00004665 F9             <1>       STC
00004666 C3             <1>       RETn
                        <1>
                        <1> NON_DRV2:
                        <1>       ;XOR   AX,AX              ; CLEAR PARMS IF NO DRIVES OR CMOS BAD
00004667 31C0           <1>       xor    eax, eax
00004669 66894500       <1>       MOV    [eBP],AX           ; TRACKS, SECTORS/TRACK = 0
                        <1>       ;MOV   [BP+5],AH          ; HEAD = 0
0000466D 886509         <1>       mov    [ebp+9], ah ; 06/02/2015
                        <1>       ;MOV   [BP+6],AX          ; OFFSET TO DISK_BASE = 0
00004670 89450C         <1>       mov    [ebp+12], eax
                        <1>       ;;MOV  ES,AX              ; ES IS SEGMENT OF TABLE
                        <1>       ;JMP   SHORT DP_OUT
                        <1>
                        <1>       ; 30/08/2020
00004673 E83C020000     <1>       call   XLAT_OLD
                        <1>       ;mov   ah, NOT_RDY ; drive not ready
00004678 B407           <1>       mov    ah, INIT_FAIL ; DRIVE PARAMETER ACTIVITY FAILED
0000467A F9             <1>       stc    ; cf -> 1, ah = 'drive not ready' error code
0000467B C3             <1>       retn
                        <1>
                        <1> ;-----      DATA RATE IS EITHER 300 KBS OR 500 KBS, TRY 1.2 MB TABLE FIRST
                        <1>
                        <1> USE_EST2:
0000467C B002           <1>       MOV    AL,02              ; DRIVE TYPE 2 (1.2MB)
0000467E E88C010000     <1>       CALL   DR_TYPE_CHECK      ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
00004683 8A4B04         <1>         MOV     CL, [eBX+MD.SEC_TRK]    ; GET SECTOR/TRACK
00004686 8A6B0B         <1>         MOV     CH, [eBX+MD.MAX_TRK]    ; GET MAX. TRACK NUMBER
00004689 80FC40         <1>       CMP    AH,RATE_300        ; RATE 300 ?
0000468C 7496           <1>       JZ     short STO_CX       ; MUST BE 1.2MB DRIVE
0000468E EB87           <1>       JMP    SHORT PARM144      ; ELSE, IT IS 1.44MB DRIVE
                        <1>
                        <1> ; 30/08/2020
                        <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:
00004690 E8EE010000     <1>       CALL   XLAT_NEW           ; TRANSLATE STATE TO PRESENT ARCH.
00004695 8A87[558A0100] <1>       MOV    AL,[DSK_STATE+eDI] ; GET PRESENT STATE INFORMATION
0000469B 08C0           <1>       OR     AL,AL              ; CHECK FOR NO DRIVE
0000469D 7418           <1>       JZ     short NO_DRV
0000469F B401           <1>       MOV    AH,NOCHGLN         ; NO CHANGE LINE FOR 40 TRACK DRIVE
000046A1 A801           <1>       TEST   AL,TRK_CAPA        ; IS THIS DRIVE AN 80 TRACK DRIVE?
000046A3 7402           <1>       JZ     short DT_BACK      ; IF NO JUMP
000046A5 B402           <1>       MOV    AH,CHGLN           ; CHANGE LINE FOR 80 TRACK DRIVE
                        <1> DT_BACK:
000046A7 6650           <1>       PUSH   AX                 ; SAVE RETURN VALUE
000046A9 E806020000     <1>       CALL   XLAT_OLD           ; TRANSLATE STATE TO COMPATIBLE MODE
000046AE 6658           <1>       POP    AX                 ; RESTORE RETURN VALUE
000046B0 F8             <1>       CLC                       ; NO ERROR
000046B1 6689F3         <1>       MOV    BX,SI              ; GET SAVED AL TO BL
000046B4 88D8           <1>       MOV    AL,BL              ; PUT BACK FOR RETURN
000046B6 C3             <1>       RETn
                        <1> NO_DRV:
                        <1>       ;XOR   AH,AH              ; NO DRIVE PRESENT OR UNKNOWN
                        <1>       ;JMP   SHORT DT_BACK
                        <1>
                        <1>       ; 30/08/2020
000046B7 E8F8010000     <1>       call   XLAT_OLD
000046BC 29C0           <1>       sub    eax, eax
000046BE F9             <1>       stc    ; cf = 1 -> drive not ready, ah = 0 (disk type = 0)
000046BF C3             <1>       retn
                        <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:
000046C0 E8BE010000         <1>       CALL   XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
000046C5 8A87[558A0100]     <1>       MOV    AL, [DSK_STATE+eDI] ; GET MEDIA STATE INFORMATION
000046CB 08C0               <1>       OR     AL,AL               ; DRIVE PRESENT ?
000046CD 7422               <1>       JZ     short DC_NON        ; JUMP IF NO DRIVE
000046CF A801               <1>       TEST   AL,TRK_CAPA         ; 80 TRACK DRIVE ?
000046D1 7407               <1>       JZ     short SETIT         ; IF SO , CHECK CHANGE LINE
                            <1> DC0:
000046D3 E88D0A0000         <1>         CALL    READ_DSKCHNG            ; GO CHECK STATE OF DISK CHANGE LINE
000046D8 7407               <1>       JZ     short FINIS         ; CHANGE LINE NOT ACTIVE
                            <1>
000046DA C605[488A0100]06   <1> SETIT:       MOV    byte [DSKETTE_STATUS], MEDIA_CHANGE ; INDICATE MEDIA REMOVED
                            <1>
000046E1 E8CE010000         <1> FINIS:       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
000046E6 E808070000         <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
000046EB 6689F3             <1>       MOV    BX,SI               ; GET SAVED AL TO BL
000046EE 88D8               <1>       MOV    AL,BL               ; PUT BACK FOR RETURN
000046F0 C3                 <1>       RETn
                            <1> DC_NON:
000046F1 800D[488A0100]80   <1>       OR     byte [DSKETTE_STATUS], TIME_OUT ; SET TIMEOUT, NO DRIVE
000046F8 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:
000046FA E884010000         <1>       CALL   XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
000046FF 6656               <1>       PUSH   SI                  ; SAVE DASD TYPE
00004701 6689F0             <1>       MOV    AX,SI               ; AH = ? , AL , DASD TYPE
00004704 30E4               <1>       XOR    AH,AH               ; AH , 0 , AL , DASD TYPE
00004706 6689C6             <1>       MOV    SI,AX               ; SI = DASD TYPE
00004709 80A7[558A0100]0F   <1>       AND    byte [DSK_STATE+eDI], ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR STATE
00004710 664E               <1>       DEC    SI                  ; CHECK FOR 320/360K MEDIA & DRIVE
00004712 7509               <1>       JNZ    short NOT_320       ; BYPASS IF NOT
00004714 808F[558A0100]90   <1>       OR     byte [DSK_STATE+eDI], MED_DET+RATE_250 ; SET TO 320/360
0000471B EB48               <1>       JMP    SHORT S0
                            <1>
                            <1> NOT_320:
0000471D E8B6030000         <1>       CALL   MED_CHANGE          ; CHECK FOR TIME_OUT
00004722 803D[488A0100]80   <1>       CMP    byte [DSKETTE_STATUS], TIME_OUT
00004729 743A               <1>       JZ     short S0            ; IF TIME OUT TELL CALLER
                            <1> S3:
0000472B 664E               <1>       DEC    SI                  ; CHECK FOR 320/360K IN 1.2M DRIVE
0000472D 7509               <1>       JNZ    short NOT_320_12    ; BYPASS IF NOT
0000472F 808F[558A0100]70   <1>       OR     byte [DSK_STATE+eDI], MED_DET+DBL_STEP+RATE_300 ; SET STATE
00004736 EB2D               <1>       JMP    SHORT S0
                            <1>
                            <1> NOT_320_12:
00004738 664E               <1>       DEC    SI                  ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
0000473A 7509               <1>       JNZ    short NOT_12        ; BYPASS IF NOT
0000473C 808F[558A0100]10   <1>       OR     byte [DSK_STATE+eDI], MED_DET+RATE_500 ; SET STATE VARIABLE
00004743 EB20               <1>       JMP    SHORT S0            ; RETURN TO CALLER
                            <1>
                            <1> NOT_12:
00004745 664E               <1>       DEC    SI                  ; CHECK FOR SET DASD TYPE 04
00004747 752B               <1>       JNZ    short FS_ERR        ; BAD COMMAND EXIT IF NOT VALID TYPE
                            <1>
00004749 F687[558A0100]04   <1>       TEST   byte [DSK_STATE+eDI], DRV_DET ; DRIVE DETERMINED ?
00004750 740B               <1>       JZ     short ASSUME        ; IF STILL NOT DETERMINED ASSUME
00004752 B050               <1>       MOV    AL,MED_DET+RATE_300
00004754 F687[558A0100]02   <1>         TEST    byte [DSK_STATE+eDI], FMT_CAPA ; MULTIPLE FORMAT CAPABILITY ?
0000475B 7502               <1>       JNZ    short OR_IT_IN            ; IF 1.2 M THEN DATA RATE 300
                            <1>
                            <1> ASSUME:
0000475D B090               <1>       MOV    AL,MED_DET+RATE_250 ; SET UP
                            <1>
                            <1> OR_IT_IN:
0000475F 0887[558A0100]     <1>       OR     [DSK_STATE+eDI], AL ; OR IN THE CORRECT STATE
                            <1> S0:
00004765 E84A010000         <1>       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
0000476A E884060000         <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
0000476F 665B               <1>       POP    BX                  ; GET SAVED AL TO BL
00004771 88D8               <1>       MOV    AL,BL               ; PUT BACK FOR RETURN
00004773 C3                 <1>       RETn
                            <1>
                            <1> FS_ERR:
00004774 C605[488A0100]01   <1>       MOV    byte [DSKETTE_STATUS], BAD_CMD ; UNKNOWN STATE,BAD COMMAND
0000477B 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
```

```
1144                            <1> ;     IF ERROR:
1145                            <1> ;           AH = @DSKETTE_STATUS
1146                            <1> ;           CY = 1
1147                            <1> ;-------------------------------------------------------------------------------
1148                            <1> SET_MEDIA:
1149 0000477D E801010000       <1>       CALL   XLAT_NEW           ; TRANSLATE STATE TO PRESENT ARCH.
1150 00004782 F687[558A0100]01 <1>         TEST    byte [DSK_STATE+eDI], TRK_CAPA ; CHECK FOR CHANGE LINE AVAILABLE
1151 00004789 7415             <1>       JZ     short SM_CMOS      ; JUMP IF 40 TRACK DRIVE
1152 0000478B E848030000       <1>       CALL   MED_CHANGE         ; RESET CHANGE LINE
1153 00004790 803D[488A0100]80 <1>       CMP    byte [DSKETTE_STATUS], TIME_OUT ; IF TIME OUT TELL CALLER
1154 00004797 746B             <1>       JE     short SM_RTN
1155 00004799 C605[488A0100]00 <1>       MOV    byte [DSKETTE_STATUS], 0 ; CLEAR STATUS
1156                            <1> SM_CMOS:
1157 000047A0 E819070000       <1>       CALL   CMOS_TYPE          ; RETURN DRIVE TYPE IN (AL)
1158                            <1>       ;;20/02/2015
1159                            <1>       ;;JC   short MD_NOT_FND   ; ERROR IN CMOS
1160                            <1>       ;;OR   AL,AL              ; TEST FOR NO DRIVE
1161 000047A5 745D             <1>       JZ     short SM_RTN       ; RETURN IF SO
1162 000047A7 E863000000       <1>       CALL   DR_TYPE_CHECK      ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
1163 000047AC 7231             <1>       JC     short MD_NOT_FND   ; TYPE NOT IN TABLE (BAD CMOS)
1164 000047AE 57               <1>       PUSH   eDI                ; SAVE REG.
1165 000047AF 31DB             <1>       XOR    eBX,eBX                  ; BX = INDEX TO DR. TYPE TABLE
1166 000047B1 B906000000       <1>       MOV    eCX,DR_CNT         ; CX = LOOP COUNT
1167                            <1> DR_SEARCH:
1168 000047B6 8AA3[786D0000]   <1>       MOV    AH, [DR_TYPE+eBX]  ; GET DRIVE TYPE
1169 000047BC 80E47F           <1>       AND    AH,BIT7OFF         ; MASK OUT MSB
1170 000047BF 38E0             <1>       CMP    AL,AH              ; DRIVE TYPE MATCH ?
1171 000047C1 7516             <1>       JNE    short NXT_MD       ; NO, CHECK NEXT DRIVE TYPE
1172                            <1> DR_FND:
1173 000047C3 8BBB[796D0000]   <1>       MOV    eDI, [DR_TYPE+eBX+1]      ; DI = MEDIA/DRIVE PARAM TABLE
1174                            <1> MD_SEARCH:
1175 000047C9 8A6704           <1>         MOV     AH, [eDI+MD.SEC_TRK]    ; GET SECTOR/TRACK
1176 000047CC 386500           <1>       CMP    [eBP],AH           ; MATCH?
1177 000047CF 7508             <1>       JNE    short NXT_MD       ; NO, CHECK NEXT MEDIA
1178 000047D1 8A670B           <1>         MOV     AH, [eDI+MD.MAX_TRK]    ; GET MAX. TRACK #
1179 000047D4 386501           <1>       CMP    [eBP+1],AH         ; MATCH?
1180 000047D7 740F             <1>       JE     short MD_FND       ; YES, GO GET RATE
1181                            <1> NXT_MD:
1182                            <1>       ;ADD   BX,3               ; CHECK NEXT DRIVE TYPE
1183 000047D9 83C305           <1>         add ebx, 5 ; 18/02/2015
1184 000047DC E2D8             <1>       LOOP    DR_SEARCH
1185 000047DE 5F               <1>       POP    eDI                ; RESTORE REG.
1186                            <1> MD_NOT_FND:
1187 000047DF C605[488A0100]0C <1>       MOV    byte [DSKETTE_STATUS], MED_NOT_FND ; ERROR, MEDIA TYPE NOT FOUND
1188 000047E6 EB1C             <1>       JMP    SHORT SM_RTN       ; RETURN
1189                            <1> MD_FND:
1190 000047E8 8A470C           <1>         MOV     AL, [eDI+MD.RATE]       ; GET RATE
1191 000047EB 3C40             <1>       CMP    AL,RATE_300        ; DOUBLE STEP REQUIRED FOR RATE 300
1192 000047ED 7502             <1>       JNE    short MD_SET
1193 000047EF 0C20             <1>       OR     AL,DBL_STEP
1194                            <1> MD_SET:
1195                            <1>       ;MOV   [BP+6],DI          ; SAVE TABLE POINTER IN STACK
1196 000047F1 897D0C           <1>       mov    [ebp+12], edi ; 18/02/2015
1197 000047F4 0C10             <1>       OR     AL,MED_DET         ; SET MEDIA ESTABLISHED
1198 000047F6 5F               <1>       POP    eDI
1199 000047F7 80A7[558A0100]0F <1>       AND    byte [DSK_STATE+eDI], ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR STATE
1200 000047FE 0887[558A0100]   <1>       OR     [DSK_STATE+eDI], AL
1201                            <1>       ;MOV   AX, CS             ; SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
1202                            <1>       ;MOV   ES, AX             ; ES IS SEGMENT OF TABLE
1203                            <1> SM_RTN:
1204 00004804 E8AB000000       <1>       CALL   XLAT_OLD           ; TRANSLATE STATE TO COMPATIBLE MODE
1205 00004809 E8E5050000       <1>       CALL   SETUP_END          ; VARIOUS CLEANUPS
1206 0000480E C3               <1>       RETn
1207                            <1>
1208                            <1> ;--------------------------------------------------------------
1209                            <1> ; DR_TYPE_CHECK                                          :
1210                            <1> ;     CHECK IF THE GIVEN DRIVE TYPE IN REGISTER (AL)           :
1211                            <1> ;     IS SUPPORTED IN BIOS DRIVE TYPE TABLE                :
1212                            <1> ; ON ENTRY:                                        :
1213                            <1> ;     AL = DRIVE TYPE                                      :
1214                            <1> ; ON EXIT:                                         :
1215                            <1> ;     CS = SEGMENT MEDIA/DRIVE PARAMETER TABLE (CODE)          :
1216                            <1> ;     CY = 0       DRIVE TYPE SUPPORTED                        :
1217                            <1> ;          BX = OFFSET TO MEDIA/DRIVE PARAMETER TABLE          :
1218                            <1> ;     CY = 1 DRIVE TYPE NOT SUPPORTED               :
1219                            <1> ; REGISTERS ALTERED: eBX                                 :
1220                            <1> ;--------------------------------------------------------------
1221                            <1> DR_TYPE_CHECK:
1222 0000480F 6650             <1>       PUSH   AX
1223 00004811 51               <1>       PUSH   eCX
1224 00004812 31DB             <1>       XOR    eBX,eBX                  ; BX = INDEX TO DR_TYPE TABLE
1225 00004814 B906000000       <1>       MOV    eCX,DR_CNT         ; CX = LOOP COUNT
1226                            <1> TYPE_CHK:
1227 00004819 8AA3[786D0000]   <1>       MOV    AH,[DR_TYPE+eBX]   ; GET DRIVE TYPE
1228 0000481F 38E0             <1>       CMP    AL,AH              ; DRIVE TYPE MATCH?
1229 00004821 740D             <1>       JE     short DR_TYPE_VALID ; YES, RETURN WITH CARRY RESET
1230                            <1>       ;ADD   BX,3               ; CHECK NEXT DRIVE TYPE
1231 00004823 83C305           <1>         add ebx, 5 ; 16/02/2015 (32 bit address modification)
1232 00004826 E2F1             <1>       LOOP    TYPE_CHK
1233                            <1>       ;
1234 00004828 BB[D76D0000]     <1>       mov    ebx, MD_TBL6       ; 1.44MB fd parameter table
1235                            <1>                                 ; Default for GET_PARM (11/12/2014)
1236                            <1>       ;
1237 0000482D F9               <1>       STC                       ; DRIVE TYPE NOT FOUND IN TABLE
1238 0000482E EB06             <1>       JMP    SHORT TYPE_RTN
1239                            <1> DR_TYPE_VALID:
1240 00004830 8B9B[796D0000]   <1>       MOV    eBX,[DR_TYPE+eBX+1]       ; BX = MEDIA TABLE
1241                            <1> TYPE_RTN:
1242 00004836 59               <1>       POP    eCX
1243 00004837 6658             <1>       POP    AX
1244 00004839 C3               <1>       RETn
1245                            <1>
1246                            <1> ;--------------------------------------------------------------
1247                            <1> ; SEND_SPEC                                              :
1248                            <1> ;     SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM    :
```

```
1249                            <1> ;     THE DRIVE PARAMETER TABLE POINTED BY @DISK_POINTER   :
1250                            <1> ; ON ENTRY:  @DISK_POINTER = DRIVE PARAMETER TABLE         :
1251                            <1> ; ON EXIT:   NONE                                  :
1252                            <1> ; REGISTERS ALTERED: CX, DX                                :
1253                            <1> ;-----------------------------------------------------------------
1254                            <1> SEND_SPEC:
1255 0000483A 50               <1>       PUSH   eAX                ; SAVE AX
1256 0000483B B8[61480000]     <1>       MOV    eAX, SPECBAC       ; LOAD ERROR ADDRESS
1257 00004840 50               <1>       PUSH   eAX                ; PUSH NEC_OUT ERROR RETURN
1258 00004841 B403             <1>       MOV    AH,03H             ; SPECIFY COMMAND
1259 00004843 E885070000       <1>       CALL   NEC_OUTPUT         ; OUTPUT THE COMMAND
1260 00004848 28D2             <1>       SUB    DL,DL              ; FIRST SPECIFY BYTE
1261 0000484A E878060000       <1>       CALL   GET_PARM           ; GET PARAMETER TO AH
1262 0000484F E879070000       <1>       CALL   NEC_OUTPUT         ; OUTPUT THE COMMAND
1263 00004854 B201             <1>       MOV    DL,1               ; SECOND SPECIFY BYTE
1264 00004856 E86C060000       <1>       CALL   GET_PARM           ; GET PARAMETER TO AH
1265 0000485B E86D070000       <1>       CALL   NEC_OUTPUT         ; OUTPUT THE COMMAND
1266 00004860 58               <1>       POP    eAX                ; POP ERROR RETURN
1267                            <1> SPECBAC:
1268 00004861 58               <1>       POP    eAX                ; RESTORE ORIGINAL AX VALUE
1269 00004862 C3               <1>       RETn
1270                            <1>
1271                            <1> ;-----------------------------------------------------------------
1272                            <1> ; SEND_SPEC_MD                                             :
1273                            <1> ;     SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM     :
1274                            <1> ;     THE MEDIA/DRIVE PARAMETER TABLE POINTED BY (CS:BX)   :
1275                            <1> ; ON ENTRY:  CS:BX = MEDIA/DRIVE PARAMETER TABLE           :
1276                            <1> ; ON EXIT:   NONE                                  :
1277                            <1> ; REGISTERS ALTERED: AX                                    :
1278                            <1> ;-----------------------------------------------------------------
1279                            <1> SEND_SPEC_MD:
1280 00004863 50               <1>       PUSH   eAX                ; SAVE RATE DATA
1281 00004864 B8[81480000]     <1>       MOV    eAX, SPEC_ESBAC           ; LOAD ERROR ADDRESS
1282 00004869 50               <1>       PUSH   eAX                ; PUSH NEC_OUT ERROR RETURN
1283 0000486A B403             <1>       MOV    AH,03H             ; SPECIFY COMMAND
1284 0000486C E85C070000       <1>       CALL   NEC_OUTPUT         ; OUTPUT THE COMMAND
1285 00004871 8A23             <1>         MOV     AH, [eBX+MD.SPEC1]      ; GET 1ST SPECIFY BYTE
1286 00004873 E855070000       <1>       CALL   NEC_OUTPUT         ; OUTPUT THE COMMAND
1287 00004878 8A6301           <1>         MOV     AH, [eBX+MD.SPEC2]      ; GET SECOND SPECIFY BYTE
1288 0000487B E84D070000       <1>       CALL   NEC_OUTPUT         ; OUTPUT THE COMMAND
1289 00004880 58               <1>       POP    eAX                ; POP ERROR RETURN
1290                            <1> SPEC_ESBAC:
1291 00004881 58               <1>       POP    eAX                ; RESTORE ORIGINAL AX VALUE
1292 00004882 C3               <1>       RETn
1293                            <1>
1294                            <1> ;-----------------------------------------------------------------------------
1295                            <1> ; XLAT_NEW
1296                            <1> ;     TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE
1297                            <1> ;     MODE TO NEW ARCHITECTURE.
1298                            <1> ;
1299                            <1> ; ON ENTRY:  DI = DRIVE #
1300                            <1> ;-----------------------------------------------------------------------------
1301                            <1> XLAT_NEW:
1302 00004883 83FF01           <1>       CMP    eDI,1                     ; VALID DRIVE
1303 00004886 7725             <1>       JA     short XN_OUT              ; IF INVALID BACK
1304 00004888 80BF[558A0100]00 <1>       CMP    byte [DSK_STATE+eDI], 0         ; NO DRIVE ?
1305 0000488F 741D             <1>       JZ     short DO_DET              ; IF NO DRIVE ATTEMPT DETERMINE
1306 00004891 6689F9           <1>       MOV    CX,DI                     ; CX = DRIVE NUMBER
1307 00004894 C0E102           <1>       SHL    CL,2                      ; CL = SHIFT COUNT, A=0, B=4
1308 00004897 A0[548A0100]     <1>       MOV    AL, [HF_CNTRL]                  ; DRIVE INFORMATION
1309 0000489C D2C8             <1>       ROR    AL,CL                     ; TO LOW NIBBLE
1310 0000489E 2407             <1>       AND    AL,DRV_DET+FMT_CAPA+TRK_CAPA     ; KEEP DRIVE BITS
1311 000048A0 80A7[558A0100]F8 <1>         AND     byte [DSK_STATE+eDI], ~(DRV_DET+FMT_CAPA+TRK_CAPA)
1312 000048A7 0887[558A0100]   <1>       OR     [DSK_STATE+eDI], AL       ; UPDATE DRIVE STATE
1313                            <1> XN_OUT:
1314 000048AD C3               <1>       RETn
1315                            <1> DO_DET:
1316 000048AE E8BF080000       <1>       CALL   DRIVE_DET                 ; TRY TO DETERMINE
1317 000048B3 C3               <1>       RETn
1318                            <1>
1319                            <1> ;-----------------------------------------------------------------------------
1320                            <1> ; XLAT_OLD
1321                            <1> ;     TRANSLATES DISKETTE STATE LOCATIONS FROM NEW
1322                            <1> ;     ARCHITECTURE TO COMPATIBLE MODE.
1323                            <1> ;
1324                            <1> ; ON ENTRY:  DI = DRIVE
1325                            <1> ;-----------------------------------------------------------------------------
1326                            <1> XLAT_OLD:
1327 000048B4 83FF01           <1>       CMP    eDI,1               ; VALID DRIVE ?
1328                            <1>         ;JA     short XO_OUT            ; IF INVALID BACK
1329 000048B7 0F8786000000     <1>         ja      XO_OUT
1330 000048BD 80BF[558A0100]00 <1>         CMP  byte [DSK_STATE+eDI],0    ; NO DRIVE ?
1331 000048C4 747D             <1>       JZ     short XO_OUT        ; IF NO DRIVE TRANSLATE DONE
1332                            <1>
1333                            <1> ;-----      TEST FOR SAVED DRIVE INFORMATION ALREADY SET
1334                            <1>
1335 000048C6 6689F9           <1>       MOV    CX,DI               ; CX = DRIVE NUMBER
1336 000048C9 C0E102           <1>       SHL    CL,2                ; CL = SHIFT COUNT, A=0, B=4
1337 000048CC B402             <1>       MOV    AH,FMT_CAPA         ; LOAD MULTIPLE DATA RATE BIT MASK
1338 000048CE D2CC             <1>       ROR    AH,CL               ; ROTATE BY MASK
1339 000048D0 8425[548A0100]   <1>       TEST   [HF_CNTRL], AH            ; MULTIPLE-DATA RATE DETERMINED ?
1340 000048D6 751C             <1>       JNZ    short SAVE_SET            ; IF SO, NO NEED TO RE-SAVE
1341                            <1>
1342                            <1> ;-----      ERASE DRIVE BITS IN @HF_CNTRL FOR THIS DRIVE
1343                            <1>
1344 000048D8 B407             <1>       MOV    AH,DRV_DET+FMT_CAPA+TRK_CAPA ; MASK TO KEEP
1345 000048DA D2CC             <1>       ROR    AH,CL               ; FIX MASK TO KEEP
1346 000048DC F6D4             <1>       NOT    AH                  ; TRANSLATE MASK
1347 000048DE 2025[548A0100]   <1>       AND    [HF_CNTRL], AH            ; KEEP BITS FROM OTHER DRIVE INTACT
1348                            <1>
1349                            <1> ;-----      ACCESS CURRENT DRIVE BITS AND STORE IN @HF_CNTRL
1350                            <1>
1351 000048E4 8A87[558A0100]   <1>       MOV    AL, [DSK_STATE+eDI] ; ACCESS STATE
1352 000048EA 2407             <1>       AND    AL,DRV_DET+FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
1353 000048EC D2C8             <1>       ROR    AL,CL               ; FIX FOR THIS DRIVE
```

```
1354 000048EE 0805[548A0100]      <1>       OR    [HF_CNTRL], AL            ; UPDATE SAVED DRIVE STATE
1355                              <1>
1356                              <1> ;-----      TRANSLATE TO COMPATIBILITY MODE
1357                              <1>
1358                              <1> SAVE_SET:
1359 000048F4 8AA7[558A0100]      <1>       MOV   AH, [DSK_STATE+eDI] ; ACCESS STATE
1360 000048FA 88E7                <1>       MOV   BH,AH               ; TO BH FOR LATER
1361 000048FC 80E4C0              <1>       AND   AH,RATE_MSK         ; KEEP ONLY RATE
1362 000048FF 80FC00              <1>       CMP   AH,RATE_500         ; RATE 500 ?
1363 00004902 7410                <1>       JZ    short CHK_144       ; YES 1.2/1.2 OR 1.44/1.44
1364 00004904 B001                <1>       MOV   AL,M3D1U            ; AL = 360 IN 1.2 UNESTABLISHED
1365 00004906 80FC40              <1>       CMP   AH,RATE_300         ; RATE 300 ?
1366 00004909 7518                <1>       JNZ   short CHK_250       ; NO, 360/360, 720/720 OR 720/1.44
1367 0000490B F6C720              <1>       TEST  BH,DBL_STEP         ; CHECK FOR DOUBLE STEP
1368 0000490E 751F                <1>       JNZ   short TST_DET       ; MUST BE 360 IN 1.2
1369                              <1> UNKNO:
1370 00004910 B007                <1>       MOV   AL,MED_UNK          ; NONE OF THE ABOVE
1371 00004912 EB22                <1>       JMP   SHORT AL_SET        ; PROCESS COMPLETE
1372                              <1> CHK_144:
1373 00004914 E8A5050000          <1>       CALL  CMOS_TYPE           ; RETURN DRIVE TYPE IN (AL)
1374                              <1>       ;;20/02/2015
1375                              <1>       ;;JC  short UNKNO         ; ERROR, SET 'NONE OF ABOVE'
1376 00004919 74F5                <1>       jz    short UNKNO ;; 20/02/2015
1377 0000491B 3C02                <1>       CMP   AL,2                ; 1.2MB DRIVE ?
1378 0000491D 75F1                <1>       JNE   short UNKNO         ; NO, GO SET 'NONE OF ABOVE'
1379 0000491F B002                <1>       MOV   AL,M1D1U            ; AL = 1.2 IN 1.2 UNESTABLISHED
1380 00004921 EB0C                <1>       JMP   SHORT TST_DET
1381                              <1> CHK_250:
1382 00004923 B000                <1>       MOV   AL,M3D3U            ; AL = 360 IN 360 UNESTABLISHED
1383 00004925 80FC80              <1>       CMP   AH,RATE_250         ; RATE 250 ?
1384 00004928 75E6                <1>       JNZ   short UNKNO         ; IF SO FALL IHRU
1385 0000492A F6C701              <1>       TEST  BH,TRK_CAPA         ; 80 TRACK CAPABILITY ?
1386 0000492D 75E1                <1>       JNZ   short UNKNO         ; IF SO JUMP, FALL THRU TEST_DET
1387                              <1> TST_DET:
1388 0000492F F6C710              <1>       TEST  BH,MED_DET          ; DETERMINED ?
1389 00004932 7402                <1>       JZ    short AL_SET        ; IF NOT THEN SET
1390 00004934 0403                <1>       ADD   AL,3                ; MAKE DETERMINED/ESTABLISHED
1391                              <1> AL_SET:
1392 00004936 80A7[558A0100]F8    <1>       AND   byte [DSK_STATE+eDI], ~(DRV_DET+FMT_CAPA+TRK_CAPA) ; CLEAR DRIVE
1393 0000493D 0887[558A0100]      <1>       OR    [DSK_STATE+eDI], AL ; REPLACE WITH COMPATIBLE MODE
1394                              <1> XO_OUT:
1395 00004943 C3                  <1>       RETn
1396                              <1>
1397                              <1> ;--------------------------------------------------------------------------------
1398                              <1> ; RD_WR_VF
1399                              <1> ;     COMMON READ, WRITE AND VERIFY:
1400                              <1> ;     MAIN LOOP FOR STATE RETRIES.
1401                              <1> ;
1402                              <1> ; ON ENTRY:  AH = READ/WRITE/VERIFY NEC PARAMETER
1403                              <1> ;            AL = READ/WRITE/VERIFY DMA PARAMETER
1404                              <1> ;
1405                              <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1406                              <1> ;--------------------------------------------------------------------------------
1407                              <1> RD_WR_VF:
1408 00004944 6650                <1>       PUSH  AX                  ; SAVE DMA, NEC PARAMETERS
1409 00004946 E838FFFFFF          <1>       CALL  XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
1410 0000494B E8F3000000          <1>       CALL  SETUP_STATE         ; INITIALIZE START AND END RATE
1411 00004950 6658                <1>       POP   AX                  ; RESTORE READ/WRITE/VERIFY
1412                              <1> DO_AGAIN:
1413 00004952 6650                <1>       PUSH  AX                  ; SAVE READ/WRITE/VERIFY PARAMETER
1414 00004954 E87F010000          <1>       CALL  MED_CHANGE          ; MEDIA CHANGE AND RESET IF CHANGED
1415 00004959 6658                <1>       POP   AX                  ; RESTORE READ/WRITE/VERIFY
1416 0000495B 0F82C9000000        <1>         JC      RWV_END                 ; MEDIA CHANGE ERROR OR TIME-OUT
1417                              <1> RWV:
1418 00004961 6650                <1>       PUSH  AX                  ; SAVE READ/WRITE/VERIFY PARAMETER
1419 00004963 8AB7[558A0100]      <1>       MOV   DH, [DSK_STATE+eDI] ; GET RATE STATE OF THIS DRIVE
1420 00004969 80E6C0              <1>       AND   DH,RATE_MSK         ; KEEP ONLY RATE
1421 0000496C E84D050000          <1>       CALL  CMOS_TYPE           ; RETURN DRIVE TYPE IN AL (AL)
1422                              <1>       ;;20/02/2015
1423                              <1>       ;;JC  short RWV_ASSUME    ; ERROR IN CMOS
1424 00004971 7451                <1>       jz    short RWV_ASSUME ; 20/02/2015
1425 00004973 3C01                <1>       CMP   AL,1                ; 40 TRACK DRIVE?
1426 00004975 750D                <1>       JNE   short RWV_1         ; NO, BYPASS CMOS VALIDITY CHECK
1427 00004977 F687[558A0100]01    <1>       TEST  byte [DSK_STATE+eDI], TRK_CAPA ; CHECK FOR 40 TRACK DRIVE
1428 0000497E 7413                <1>       JZ    short RWV_2         ; YES, CMOS IS CORRECT
1429 00004980 B002                <1>       MOV   AL,2                ; CHANGE TO 1.2M
1430 00004982 EB0F                <1>       JMP   SHORT RWV_2
1431                              <1> RWV_1:
1432 00004984 720D                <1>       JB    short RWV_2         ; NO DRIVE SPECIFIED, CONTINUE
1433 00004986 F687[558A0100]01    <1>       TEST    byte [DSK_STATE+eDI], TRK_CAPA ; IS IT REALLY 40 TRACK?
1434 0000498D 7504                <1>       JNZ   short RWV_2         ; NO, 80 TRACK
1435 0000498F B001                <1>       MOV   AL,1                ; IT IS 40 TRACK, FIX CMOS VALUE
1436 00004991 EB04                <1>       jmp   short rwv_3
1437                              <1> RWV_2:
1438 00004993 08C0                <1>       OR    AL,AL               ; TEST FOR NO DRIVE
1439 00004995 742D                <1>       JZ    short RWV_ASSUME    ; ASSUME TYPE, USE MAX TRACK
1440                              <1> rwv_3:
1441 00004997 E873FEFFFF          <1>       CALL  DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL.
1442 0000499C 7226                <1>       JC    short RWV_ASSUME    ; TYPE NOT IN TABLE (BAD CMOS)
1443                              <1>
1444                              <1> ;-----      SEARCH FOR MEDIA/DRIVE PARAMETER TABLE
1445                              <1>
1446 0000499E 57                  <1>       PUSH  eDI                 ; SAVE DRIVE #
1447 0000499F 31DB                <1>       XOR   eBX,eBX                   ; BX = INDEX TO DR_TYPE TABLE
1448 000049A1 B906000000          <1>       MOV   eCX,DR_CNT          ; CX = LOOP COUNT
1449                              <1> RWV_DR_SEARCH:
1450 000049A6 8AA3[786D0000]      <1>       MOV   AH, [DR_TYPE+eBX]   ; GET DRIVE TYPE
1451 000049AC 80E47F              <1>       AND   AH,BIT7OFF          ; MASK OUT MSB
1452 000049AF 38E0                <1>       CMP   AL,AH               ; DRIVE TYPE MATCH?
1453 000049B1 750B                <1>       JNE   short RWV_NXT_MD    ; NO, CHECK NEXT DRIVE TYPE
1454                              <1> RWV_DR_FND:
1455 000049B3 8BBB[796D0000]      <1>       MOV   eDI, [DR_TYPE+eBX+1]      ; DI = MEDIA/DRIVE PARAMETER TABLE
1456                              <1> RWV_MD_SEARH:
1457 000049B9 3A770C              <1>         CMP     DH, [eDI+MD.RATE]       ; MATCH?
1458 000049BC 741B                <1>       JE    short RWV_MD_FND    ; YES, GO GET 1ST SPECIFY BYTE
```

```
1459                            <1> RWV_NXT_MD:
1460                            <1>       ;ADD   BX,3               ; CHECK NEXT DRIVE TYPE
1461 000049BE 83C305           <1>       add    eBX, 5
1462 000049C1 E2E3             <1>       LOOP   RWV_DR_SEARCH
1463 000049C3 5F               <1>       POP    eDI                ; RESTORE DRIVE #
1464                            <1>
1465                            <1> ;-----      ASSUME PRIMARY DRIVE IS INSTALLED AS SHIPPED
1466                            <1>
1467                            <1> RWV_ASSUME:
1468 000049C4 BB[966D0000]      <1>       MOV    eBX, MD_TBL1        ; POINT TO 40 TRACK 250 KBS
1469 000049C9 F687[558A0100]01  <1>       TEST   byte [DSK_STATE+eDI], TRK_CAPA ; TEST FOR 80 TRACK
1470 000049D0 740A             <1>       JZ     short RWV_MD_FND1   ; MUST BE 40 TRACK
1471 000049D2 BB[B06D0000]      <1>       MOV    eBX, MD_TBL3        ; POINT TO 80 TRACK 500 KBS
1472 000049D7 EB03             <1>       JMP    short RWV_MD_FND1   ; GO SPECIFY PARAMTERS
1473                            <1>
1474                            <1> ;-----      CS:BX POINTS TO MEDIA/DRIVE PARAMETER TABLE
1475                            <1>
1476                            <1> RWV_MD_FND:
1477 000049D9 89FB             <1>       MOV    eBX,eDI                   ; BX = MEDIA/DRIVE PARAMETER TABLE
1478 000049DB 5F               <1>       POP    eDI                ; RESTORE DRIVE #
1479                            <1>
1480                            <1> ;-----      SEND THE SPECIFY COMMAND TO THE CONTROLLER
1481                            <1>
1482                            <1> RWV_MD_FND1:
1483 000049DC E882FEFFFF       <1>       CALL   SEND_SPEC_MD
1484 000049E1 E864010000       <1>       CALL   CHK_LASTRATE        ; ZF=1 ATTEMP RATE IS SAME AS LAST RATE
1485 000049E6 7405             <1>       JZ     short RWV_DBL       ; YES,SKIP SEND RATE COMMAND
1486 000049E8 E83B010000       <1>       CALL   SEND_RATE           ; SEND DATA RATE TO NEC
1487                            <1> RWV_DBL:
1488 000049ED 53               <1>       PUSH   eBX                 ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
1489 000049EE E822040000       <1>       CALL   SETUP_DBL           ; CHECK FOR DOUBLE STEP
1490 000049F3 5B               <1>       POP    eBX                 ; RESTORE ADDRESS
1491 000049F4 7226             <1>       JC     short CHK_RET       ; ERROR FROM READ ID, POSSIBLE RETRY
1492 000049F6 6658             <1>       POP    AX                  ; RESTORE NEC, DMA COMMAND
1493 000049F8 6650             <1>       PUSH   AX                  ; SAVE NEC COMMAND
1494 000049FA 53               <1>       PUSH   eBX                 ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
1495 000049FB E861010000       <1>       CALL   DMA_SETUP           ; SET UP THE DMA
1496 00004A00 5B               <1>       POP    eBX
1497 00004A01 6658             <1>       POP    AX                  ; RESTORE NEC COMMAND
1498 00004A03 722F             <1>       JC     short RWV_BAC       ; CHECK FOR DMA BOUNDARY ERROR
1499 00004A05 6650             <1>       PUSH   AX                  ; SAVE NEC COMMAND
1500 00004A07 53               <1>       PUSH   eBX                 ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
1501 00004A08 E83C020000       <1>       CALL   NEC_INIT            ; INITIALIZE NEC
1502 00004A0D 5B               <1>       POP    eBX                 ; RESTORE ADDRESS
1503 00004A0E 720C             <1>       JC     short CHK_RET       ; ERROR - EXIT
1504 00004A10 E866020000       <1>       CALL   RWV_COM                   ; OP CODE COMMON TO READ/WRITE/VERIFY
1505 00004A15 7205             <1>       JC     short CHK_RET       ; ERROR - EXIT
1506 00004A17 E8AB020000       <1>       CALL   NEC_TERM            ; TERMINATE, GET STATUS, ETC.
1507                            <1> CHK_RET:
1508 00004A1C E84A030000       <1>       CALL   RETRY               ; CHECK FOR, SETUP RETRY
1509 00004A21 6658             <1>       POP    AX                  ; RESTORE READ/WRITE/VERIFY PARAMETER
1510 00004A23 7305             <1>       JNC    short RWV_END       ; CY = 0 NO RETRY
1511 00004A25 E928FFFFFF       <1>         JMP     DO_AGAIN                ; CY = 1 MEANS RETRY
1512                            <1> RWV_END:
1513 00004A2A E8F4020000       <1>       CALL   DSTATE              ; ESTABLISH STATE IF SUCCESSFUL
1514 00004A2F E887030000       <1>       CALL   NUM_TRANS           ; AL = NUMBER TRANSFERRED
1515                            <1> RWV_BAC:                       ; BAD DMA ERROR ENTRY
1516 00004A34 6650             <1>       PUSH   AX                  ; SAVE NUMBER TRANSFERRED
1517 00004A36 E879FEFFFF       <1>       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
1518 00004A3B 6658             <1>       POP    AX                  ; RESTORE NUMBER TRANSFERRED
1519 00004A3D E8B1030000       <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
1520 00004A42 C3               <1>       RETn
1521                            <1>
1522                            <1> ;--------------------------------------------------------------------------
1523                            <1> ; SETUP_STATE:     INITIALIZES START AND END RATES.
1524                            <1> ;--------------------------------------------------------------------------
1525                            <1> SETUP_STATE:
1526 00004A43 F687[558A0100]10  <1>       TEST   byte [DSK_STATE+eDI], MED_DET ; MEDIA DETERMINED ?
1527 00004A4A 7537             <1>       JNZ    short J1C           ; NO STATES IF DETERMINED
1528 00004A4C 66B84000         <1>         MOV     AX,(RATE_500*256)+RATE_300  ; AH = START RATE, AL = END RATE
1529 00004A50 F687[558A0100]04  <1>       TEST   byte [DSK_STATE+eDI],DRV_DET ; DRIVE ?
1530 00004A57 740D             <1>       JZ     short AX_SET        ; DO NOT KNOW DRIVE
1531 00004A59 F687[558A0100]02  <1>       TEST   byte [DSK_STATE+eDI], FMT_CAPA ; MULTI-RATE?
1532 00004A60 7504             <1>       JNZ    short AX_SET        ; JUMP IF YES
1533 00004A62 66B88080         <1>         MOV     AX,RATE_250*257         ; START A END RATE 250 FOR 360 DRIVE
1534                            <1> AX_SET:
1535 00004A66 80A7[558A0100]1F  <1>       AND    byte [DSK_STATE+eDI], ~(RATE_MSK+DBL_STEP) ; TURN OFF THE RATE
1536 00004A6D 08A7[558A0100]    <1>       OR     [DSK_STATE+eDI], AH ; RATE FIRST TO TRY
1537 00004A73 8025[508A0100]F3  <1>       AND    byte [LASTRATE], ~STRT_MSK ; ERASE LAST TO TRY RATE BITS
1538 00004A7A C0C804           <1>       ROR    AL,4                ; TO OPERATION LAST RATE LOCATION
1539 00004A7D 0805[508A0100]    <1>       OR     [LASTRATE], AL             ; LAST RATE
1540                            <1> J1C:
1541 00004A83 C3               <1>       RETn
1542                            <1>
1543                            <1> ;--------------------------------------------------------------------------
1544                            <1> ;  FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.
1545                            <1> ;--------------------------------------------------------------------------
1546                            <1> FMT_INIT:
1547 00004A84 F687[558A0100]10  <1>       TEST   byte [DSK_STATE+eDI], MED_DET ; IS MEDIA ESTABLISHED
1548 00004A8B 7546             <1>       JNZ    short F1_OUT        ; IF SO RETURN
1549 00004A8D E82C040000       <1>       CALL   CMOS_TYPE           ; RETURN DRIVE TYPE IN AL
1550                            <1>       ;; 20/02/2015
1551                            <1>       ;;JC   short CL_DRV        ; ERROR IN CMOS ASSUME NO DRIVE
1552 00004A92 7440             <1>       jz     short CL_DRV ;; 20/02/2015
1553 00004A94 FEC8             <1>       DEC    AL                  ; MAKE ZERO ORIGIN
1554                            <1>       ;;JS   short CL_DRV        ; NO DRIVE IF AL 0
1555 00004A96 8AA7[558A0100]    <1>       MOV    AH, [DSK_STATE+eDI] ; AH = CURRENT STATE
1556 00004A9C 80E40F           <1>       AND    AH, ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR
1557 00004A9F 08C0             <1>       OR     AL,AL               ; CHECK FOR 360
1558 00004AA1 7505             <1>       JNZ    short N_360         ; IF 360 WILL BE 0
1559 00004AA3 80CC90           <1>       OR     AH,MED_DET+RATE_250 ; ESTABLISH MEDIA
1560 00004AA6 EB25             <1>       JMP    SHORT SKP_STATE           ; SKIP OTHER STATE PROCESSING
1561                            <1> N_360:
1562 00004AA8 FEC8             <1>       DEC    AL                  ; 1.2 M DRIVE
1563 00004AAA 7505             <1>       JNZ    short N_12          ; JUMP IF NOT
```

```
                        <1> F1_RATE:
00004AAC 80CC10         <1>         OR      AH,MED_DET+RATE_500 ; SET FORMAT RATE
00004AAF EB1C           <1>         JMP     SHORT SKP_STATE           ; SKIP OTHER STATE PROCESSING
                        <1> N_12:
00004AB1 FEC8           <1>         DEC     AL                  ; CHECK FOR TYPE 3
00004AB3 750F           <1>         JNZ     short N_720         ; JUMP IF NOT
00004AB5 F6C404         <1>         TEST    AH,DRV_DET          ; IS DRIVE DETERMINED
00004AB8 7410           <1>         JZ      short ISNT_12       ; TREAT AS NON 1.2 DRIVE
00004ABA F6C402         <1>         TEST    AH,FMT_CAPA         ; IS 1.2M
00004ABD 740B           <1>         JZ      short ISNT_12       ; JUMP IF NOT
00004ABF 80CC50         <1>         OR      AH,MED_DET+RATE_300 ; RATE 300
00004AC2 EB09           <1>         JMP     SHORT SKP_STATE           ; CONTINUE
                        <1> N_720:
00004AC4 FEC8           <1>         DEC     AL                  ; CHECK FOR TYPE 4
00004AC6 750C           <1>         JNZ     short CL_DRV        ; NO DRIVE, CMOS BAD
00004AC8 EBE2           <1>         JMP     SHORT F1_RATE
                        <1> ISNT_12:
00004ACA 80CC90         <1>         OR      AH,MED_DET+RATE_250 ; MUST BE RATE 250
                        <1>
                        <1> SKP_STATE:
00004ACD 88A7[558A0100] <1>         MOV     [DSK_STATE+eDI], AH ; STORE AWAY
                        <1> F1_OUT:
00004AD3 C3             <1>         RETn
                        <1> CL_DRV:
00004AD4 30E4           <1>         XOR     AH,AH               ; CLEAR STATE
00004AD6 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:
00004AD8 E888060000     <1>         CALL    READ_DSKCHNG        ; READ DISK CHANCE LINE STATE
00004ADD 7447           <1>         JZ      short MC_OUT        ; BYPASS HANDLING DISK CHANGE LINE
00004ADF 80A7[558A0100]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>
00004AE6 6689F9         <1>         MOV     CX,DI               ; CL = DRIVE 0
00004AE9 B001           <1>         MOV     AL,1                ; MOTOR ON BIT MASK
00004AEB D2E0           <1>         SHL     AL,CL               ; TO APPROPRIATE POSITION
00004AED F6D0           <1>         NOT     AL                  ; KEEP ALL BUT MOTOR ON
00004AEF FA             <1>         CLI                         ; NO INTERRUPTS
00004AF0 2005[468A0100] <1>         AND     [MOTOR_STATUS], AL  ; TURN MOTOR OFF INDICATOR
00004AF6 FB             <1>         STI                         ; INTERRUPTS ENABLED
00004AF7 E810040000     <1>         CALL    MOTOR_ON            ; TURN MOTOR ON
                        <1>
                        <1> ;-----      THIS SEQUENCE OF SEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL
                        <1>
00004AFC E850F9FFFF     <1>         CALL    DSK_RESET           ; RESET NEC
00004B01 B501           <1>         MOV     CH,01H              ; MOVE TO CYLINDER 1
00004B03 E8FF040000     <1>         CALL    SEEK                ; ISSUE SEEK
00004B08 30ED           <1>         XOR     CH,CH               ; MOVE TO CYLINDER 0
00004B0A E8F8040000     <1>         CALL    SEEK                ; ISSUE SEEK
00004B0F C605[488A0100]06 <1>       MOV     byte [DSKETTE_STATUS], MEDIA_CHANGE ; STORE IN STATUS
                        <1> OK1:
00004B16 E84A060000     <1>         CALL    READ_DSKCHNG        ; CHECK MEDIA CHANGED AGAIN
00004B1B 7407           <1>         JZ      short OK2           ; IF ACTIVE, NO DISKETTE, TIMEOUT
                        <1> OK4:
00004B1D C605[488A0100]80 <1>       MOV     byte [DSKETTE_STATUS], TIME_OUT ; TIMEOUT IF DRIVE EMPTY
                        <1> OK2:
00004B24 F9             <1>         STC                         ; MEDIA CHANGED, SET CY
00004B25 C3             <1>         RETn
                        <1> MC_OUT:
00004B26 F8             <1>         CLC                         ; NO MEDIA CHANGED, CLEAR CY
00004B27 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:
00004B28 6650           <1>         PUSH    AX                  ; SAVE REG.
00004B2A 8025[508A0100]3F <1>       AND     byte [LASTRATE], ~SEND_MSK ; ELSE CLEAR LAST RATE ATTEMPTED
00004B31 8A87[558A0100] <1>         MOV     AL, [DSK_STATE+eDI] ; GET RATE STATE OF THIS DRIVE
00004B37 24C0           <1>         AND     AL,SEND_MSK         ; KEEP ONLY RATE BITS
00004B39 0805[508A0100] <1>         OR      [LASTRATE], AL            ; SAVE NEW RATE FOR NEXT CHECK
00004B3F C0C002         <1>         ROL     AL,2                ; MOVE TO BIT OUTPUT POSITIONS
00004B42 66BAF703       <1>         MOV     DX,03F7H            ; OUTPUT NEW DATA RATE
00004B46 EE             <1>         OUT     DX,AL
00004B47 6658           <1>         POP     AX                  ; RESTORE REG.
00004B49 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:
00004B4A 6650           <1>         PUSH    AX                  ; SAVE REG
00004B4C 2225[508A0100] <1>         AND     AH, [LASTRATE]            ; GET LAST DATA RATE SELECTED
```

```
1669 00004B52 8A87[558A0100]      <1>       MOV    AL, [DSK_STATE+eDI] ; GET RATE STATE OF THIS DRIVE
1670 00004B58 6625C0C0            <1>         AND     AX, SEND_MSK*257        ; KEEP ONLY RATE BITS OF BOTH
1671 00004B5C 38E0                <1>       CMP    AL, AH              ; COMPARE TO PREVIOUSLY TRIED
1672                              <1>                                   ; ZF = 1 RATE IS THE SAME
1673 00004B5E 6658                <1>       POP    AX                  ; RESTORE REG.
1674 00004B60 C3                  <1>       RETn
1675                              <1>
1676                              <1> ;--------------------------------------------------------------------------------
1677                              <1> ; DMA_SETUP
1678                              <1> ;     THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY OPERATIONS.
1679                              <1> ;
1680                              <1> ; ON ENTRY:  AL = DMA COMMAND
1681                              <1> ;
1682                              <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1683                              <1> ;--------------------------------------------------------------------------------
1684                              <1>
1685                              <1> ; SI = Head #, # of Sectors or DASD Type
1686                              <1>
1687                              <1> ; 22/08/2015
1688                              <1> ; 08/02/2015 - Protected Mode Modification
1689                              <1> ; 06/02/2015 - 07/02/2015
1690                              <1> ; NOTE: Buffer address must be in 1st 16MB of Physical Memory (24 bit limit).
1691                              <1> ; (DMA Addres = Physical Address)
1692                              <1> ; (Retro UNIX 386 v1 Kernel/System Mode Virtual Address = Physical Address)
1693                              <1> ;
1694                              <1>
1695                              <1>
1696                              <1> ; 04/02/2016 (clc)
1697                              <1> ; 20/02/2015 modification (source: AWARD BIOS 1999, DMA_SETUP)
1698                              <1> ; 16/12/2014 (IODELAY)
1699                              <1>
1700                              <1> DMA_SETUP:
1701                              <1>
1702                              <1> ;; 20/02/2015
1703 00004B61 8B5504              <1>       mov    edx, [ebp+4]        ; Buffer address
1704 00004B64 F7C2000000FF        <1>       test   edx, 0FF000000h           ; 16 MB limit (22/08/2015, bugfix)
1705 00004B6A 756E                <1>       jnz    short dma_bnd_err_stc
1706                              <1>       ;
1707 00004B6C 6650                <1>       push   ax                  ; DMA command
1708 00004B6E 52                  <1>       push   edx                 ; *
1709 00004B6F B203                <1>       mov    dl, 3               ; GET BYTES/SECTOR PARAMETER
1710 00004B71 E851030000          <1>       call   GET_PARM            ;
1711 00004B76 88E1                <1>       mov    cl, ah                   ; SHIFT COUNT (0=128, 1=256, 2=512 ETC)
1712 00004B78 6689F0              <1>       mov    ax, si              ; Sector count
1713 00004B7B 88C4                <1>       mov    ah, al              ; AH =  # OF SECTORS
1714 00004B7D 28C0                <1>       sub    al, al              ; AL = 0, AX = # SECTORS * 256
1715 00004B7F 66D1E8              <1>       shr    ax, 1               ; AX = # SECTORS * 128
1716 00004B82 66D3E0              <1>       shl    ax, cl              ; SHIFT BY PARAMETER VALUE
1717 00004B85 6648                <1>       dec    ax                  ; -1 FOR DMA VALUE
1718 00004B87 6689C1              <1>       mov    cx, ax
1719 00004B8A 5A                  <1>       pop    edx                 ; *
1720 00004B8B 6658                <1>       pop    ax
1721 00004B8D 3C42                <1>       cmp    al, 42h
1722 00004B8F 7507                <1>         jne     short NOT_VERF
1723 00004B91 BA0000FF00          <1>       mov    edx, 0FF0000h
1724 00004B96 EB08                <1>       jmp    short J33
1725                              <1> NOT_VERF:
1726 00004B98 6601CA              <1>       add    dx, cx              ; check for overflow
1727 00004B9B 723E                <1>       jc     short dma_bnd_err
1728                              <1>       ;
1729 00004B9D 6629CA              <1>       sub    dx, cx              ; Restore start address
1730                              <1> J33:
1731 00004BA0 FA                  <1>       CLI                        ; DISABLE INTERRUPTS DURING DMA SET-UP
1732 00004BA1 E60C                <1>       OUT    DMA+12,AL           ; SET THE FIRST/LA5T F/F
1733                              <1>       IODELAY                          ; WAIT FOR I/O
1733 00004BA3 EB00                <2>  jmp short $+2
1733 00004BA5 EB00                <2>  jmp short $+2
1734 00004BA7 E60B                <1>       OUT    DMA+11,AL           ; OUTPUT THE MODE BYTE
1735 00004BA9 89D0                <1>       mov    eax, edx            ; Buffer address
1736 00004BAB E604                <1>       OUT    DMA+4,AL            ; OUTPUT LOW ADDRESS
1737                              <1>       IODELAY                          ; WAIT FOR I/O
1737 00004BAD EB00                <2>  jmp short $+2
1737 00004BAF EB00                <2>  jmp short $+2
1738 00004BB1 88E0                <1>       MOV    AL,AH
1739 00004BB3 E604                <1>       OUT    DMA+4,AL            ; OUTPUT HIGH ADDRESS
1740 00004BB5 C1E810              <1>       shr    eax, 16
1741                              <1>       IODELAY                          ; I/O WAIT STATE
1741 00004BB8 EB00                <2>  jmp short $+2
1741 00004BBA EB00                <2>  jmp short $+2
1742 00004BBC E681                <1>       OUT    081H,AL                  ; OUTPUT highest BITS TO PAGE REGISTER
1743                              <1>       IODELAY
1743 00004BBE EB00                <2>  jmp short $+2
1743 00004BC0 EB00                <2>  jmp short $+2
1744 00004BC2 6689C8              <1>       mov    ax, cx              ; Byte count - 1
1745 00004BC5 E605                <1>       OUT    DMA+5,AL            ; LOW BYTE OF COUNT
1746                              <1>       IODELAY                          ; WAIT FOR I/O
1746 00004BC7 EB00                <2>  jmp short $+2
1746 00004BC9 EB00                <2>  jmp short $+2
1747 00004BCB 88E0                <1>       MOV    AL, AH
1748 00004BCD E605                <1>       OUT    DMA+5,AL            ; HIGH BYTE OF COUNT
1749                              <1>       IODELAY
1749 00004BCF EB00                <2>  jmp short $+2
1749 00004BD1 EB00                <2>  jmp short $+2
1750 00004BD3 FB                  <1>       STI                        ; RE-ENABLE INTERRUPTS
1751 00004BD4 B002                <1>       MOV    AL, 2               ; MODE FOR 8237
1752 00004BD6 E60A                <1>       OUT    DMA+10, AL          ; INITIALIZE THE DISKETTE CHANNEL
1753                              <1>
1754 00004BD8 F8                  <1>       clc    ; 04/02/2016
1755 00004BD9 C3                  <1>       retn
1756                              <1>
1757                              <1> dma_bnd_err_stc:
1758 00004BDA F9                  <1>       stc
1759                              <1> dma_bnd_err:
1760 00004BDB C605[488A0100]09    <1>       MOV    byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
1761 00004BE2 C3                  <1>       RETn                       ; CY SET BY ABOVE IF ERROR
```

```
1762                            <1>
1763                            <1> ;; 16/12/2014
1764                            <1> ;;    CLI                       ; DISABLE INTERRUPTS DURING DMA SET-UP
1765                            <1> ;;    OUT    DMA+12,AL          ; SET THE FIRST/LA5T F/F
1766                            <1> ;;    ;JMP   $+2                ; WAIT FOR I/O
1767                            <1> ;;    IODELAY
1768                            <1> ;;    OUT    DMA+11,AL          ; OUTPUT THE MODE BYTE
1769                            <1> ;;    ;SIODELAY
1770                            <1> ;;      ;CMP AL, 42H                  ; DMA VERIFY COMMAND
1771                            <1> ;;      ;JNE short NOT_VERF           ; NO
1772                            <1> ;;      ;XOR AX, AX             ; START ADDRESS
1773                            <1> ;;      ;JMP SHORT J33
1774                            <1> ;;;NOT_VERF:
1775                            <1> ;;    ;MOV   AX,ES              ; GET THE ES VALUE
1776                            <1> ;;    ;ROL   AX,4               ; ROTATE LEFT
1777                            <1> ;;    ;MOV   CH,AL              ; GET HIGHEST NIBBLE OF ES TO CH
1778                            <1> ;;    ;AND   AL,11110000B       ; ZERO THE LOW NIBBLE FROM SEGMENT
1779                            <1> ;;    ;ADD   AX,[BP+2]          ; TEST FOR CARRY FROM ADDITION
1780                            <1> ;;    mov    eax, [ebp+4] ; 06/02/2015
1781                            <1> ;;    ;JNC   short J33
1782                            <1> ;;    ;INC   CH                 ; CARRY MEANS HIGH 4 BITS MUST BE INC
1783                            <1> ;;;J33:
1784                            <1> ;;    PUSH   eAX                ; SAVE START ADDRESS
1785                            <1> ;;    OUT    DMA+4,AL           ; OUTPUT LOW ADDRESS
1786                            <1> ;;    ;JMP   $+2                ; WAIT FOR I/O
1787                            <1> ;;    IODELAY
1788                            <1> ;;    MOV    AL,AH
1789                            <1> ;;    OUT    DMA+4,AL           ; OUTPUT HIGH ADDRESS
1790                            <1> ;;    shr    eax, 16           ; 07/02/2015
1791                            <1> ;;    ;MOV   AL,CH              ; GET HIGH 4 BITS
1792                            <1> ;;    ;JMP   $+2                ; I/O WAIT STATE
1793                            <1> ;;    IODELAY
1794                            <1> ;;    ;AND   AL,00001111B
1795                            <1> ;;    OUT    081H,AL                  ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
1796                            <1> ;;    ;SIODELAY
1797                            <1> ;;
1798                            <1> ;;;----- DETERMINE COUNT
1799                            <1> ;;    sub    eax, eax ; 08/02/2015
1800                            <1> ;;    MOV    AX, SI             ; AL =  # OF SECTORS
1801                            <1> ;;    XCHG   AL, AH             ; AH =  # OF SECTORS
1802                            <1> ;;    SUB    AL, AL             ; AL = 0, AX = # SECTORS * 256
1803                            <1> ;;    SHR    AX, 1              ; AX = # SECTORS * 128
1804                            <1> ;;    PUSH   AX                 ; SAVE # OF SECTORS * 128
1805                            <1> ;;    MOV    DL, 3              ; GET BYTES/SECTOR PARAMETER
1806                            <1> ;;    CALL   GET_PARM           ; "
1807                            <1> ;;    MOV    CL,AH              ; SHIFT COUNT (0=128, 1=256, 2=512 ETC)
1808                            <1> ;;    POP    AX                 ; AX = # SECTORS * 128
1809                            <1> ;;    SHL    AX,CL              ; SHIFT BY PARAMETER VALUE
1810                            <1> ;;    DEC    AX                 ; -1 FOR DMA VALUE
1811                            <1> ;;    PUSH   eAX  ; 08/02/2015  ; SAVE COUNT VALUE
1812                            <1> ;;    OUT    DMA+5,AL           ; LOW BYTE OF COUNT
1813                            <1> ;;    ;JMP   $+2                ; WAIT FOR I/O
1814                            <1> ;;    IODELAY
1815                            <1> ;;    MOV    AL, AH
1816                            <1> ;;    OUT    DMA+5,AL           ; HIGH BYTE OF COUNT
1817                            <1> ;;    ;IODELAY
1818                            <1> ;;    STI                       ; RE-ENABLE INTERRUPTS
1819                            <1> ;;    POP    eCX  ; 08/02/2015  ; RECOVER COUNT VALUE
1820                            <1> ;;    POP    eAX  ; 08/02/2015  ; RECOVER ADDRESS VALUE
1821                            <1> ;;    ;ADD   AX, CX             ; ADD, TEST FOR 64K OVERFLOW
1822                            <1> ;;    add    ecx, eax ; 08/02/2015
1823                            <1> ;;    MOV    AL, 2              ; MODE FOR 8237
1824                            <1> ;;    ;JMP   $+2                ; WAIT FOR I/O
1825                            <1> ;;    SIODELAY
1826                            <1> ;;    OUT    DMA+10, AL         ; INITIALIZE THE DISKETTE CHANNEL
1827                            <1> ;;    ;JNC   short NO_BAD       ; CHECK FOR ERROR
1828                            <1> ;;    jc     short dma_bnd_err ; 08/02/2015
1829                            <1> ;;    and    ecx, 0FFF00000h ; 16 MB limit
1830                            <1> ;;    jz     short NO_BAD
1831                            <1> ;;dma_bnd_err:
1832                            <1> ;;    MOV    byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
1833                            <1> ;;NO_BAD:
1834                            <1> ;;    RETn                      ; CY SET BY ABOVE IF ERROR
1835                            <1>
1836                            <1> ;--------------------------------------------------------------------------
1837                            <1> ; FMTDMA_SET
1838                            <1> ;     THIS ROUTINE SETS UP THE DMA CONTROLLER FOR A FORMAT OPERATION.
1839                            <1> ;
1840                            <1> ; ON ENTRY:  NOTHING REQUIRED
1841                            <1> ;
1842                            <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1843                            <1> ;--------------------------------------------------------------------------
1844                            <1>
1845                            <1> FMTDMA_SET:
1846                            <1> ;; 20/02/2015 modification
1847 00004BE3 8B5504           <1>       mov    edx, [ebp+4]       ; Buffer address
1848 00004BE6 F7C20000F0FF     <1>       test   edx, 0FFF00000h          ; 16 MB limit
1849 00004BEC 75EC             <1>       jnz    short dma_bnd_err_stc
1850                            <1>       ;
1851 00004BEE 6652             <1>       push   dx                 ; *
1852 00004BF0 B204             <1>       mov    DL, 4              ; SECTORS/TRACK VALUE IN PARM TABLE
1853 00004BF2 E8D0020000       <1>       call   GET_PARM           ; "
1854 00004BF7 88E0             <1>       mov    al, ah             ; AL = SECTORS/TRACK VALUE
1855 00004BF9 28E4             <1>       sub    ah, ah             ; AX = SECTORS/TRACK VALUE
1856 00004BFB 66C1E002         <1>       shl    ax, 2              ; AX = SEC/TRK * 4 (OFFSET C,H,R,N)
1857 00004BFF 6648             <1>       dec    ax                 ; -1 FOR DMA VALUE
1858 00004C01 6689C1           <1>       mov    cx, ax
1859 00004C04 665A             <1>       pop    dx                 ; *
1860 00004C06 6601CA           <1>       add    dx, cx             ; check for overflow
1861 00004C09 72D0             <1>       jc     short dma_bnd_err
1862                            <1>       ;
1863 00004C0B 6629CA           <1>       sub    dx, cx             ; Restore start address
1864                            <1>       ;
1865 00004C0E B04A             <1>       MOV    AL, 04AH           ; WILL WRITE TO THE DISKETTE
1866 00004C10 FA               <1>       CLI                       ; DISABLE INTERRUPTS DURING DMA SET-UP
```

```
1867 00004C11 E60C                      <1>       OUT    DMA+12,AL          ; SET THE FIRST/LA5T F/F
1868                                    <1>       IODELAY                         ; WAIT FOR I/O
1868 00004C13 EB00                      <2>  jmp short $+2
1868 00004C15 EB00                      <2>  jmp short $+2
1869 00004C17 E60B                      <1>       OUT    DMA+11,AL          ; OUTPUT THE MODE BYTE
1870 00004C19 89D0                      <1>       mov    eax, edx           ; Buffer address
1871 00004C1B E604                      <1>       OUT    DMA+4,AL           ; OUTPUT LOW ADDRESS
1872                                    <1>       IODELAY                         ; WAIT FOR I/O
1872 00004C1D EB00                      <2>  jmp short $+2
1872 00004C1F EB00                      <2>  jmp short $+2
1873 00004C21 88E0                      <1>       MOV    AL,AH
1874 00004C23 E604                      <1>       OUT    DMA+4,AL           ; OUTPUT HIGH ADDRESS
1875 00004C25 C1E810                    <1>       shr    eax, 16
1876                                    <1>       IODELAY                         ; I/O WAIT STATE
1876 00004C28 EB00                      <2>  jmp short $+2
1876 00004C2A EB00                      <2>  jmp short $+2
1877 00004C2C E681                      <1>       OUT    081H,AL                  ; OUTPUT highest BITS TO PAGE REGISTER
1878                                    <1>       IODELAY
1878 00004C2E EB00                      <2>  jmp short $+2
1878 00004C30 EB00                      <2>  jmp short $+2
1879 00004C32 6689C8                    <1>       mov    ax, cx              ; Byte count - 1
1880 00004C35 E605                      <1>       OUT    DMA+5,AL            ; LOW BYTE OF COUNT
1881                                    <1>       IODELAY                         ; WAIT FOR I/O
1881 00004C37 EB00                      <2>  jmp short $+2
1881 00004C39 EB00                      <2>  jmp short $+2
1882 00004C3B 88E0                      <1>       MOV    AL, AH
1883 00004C3D E605                      <1>       OUT    DMA+5,AL            ; HIGH BYTE OF COUNT
1884                                    <1>       IODELAY
1884 00004C3F EB00                      <2>  jmp short $+2
1884 00004C41 EB00                      <2>  jmp short $+2
1885 00004C43 FB                        <1>       STI                         ; RE-ENABLE INTERRUPTS
1886 00004C44 B002                      <1>       MOV    AL, 2                ; MODE FOR 8237
1887 00004C46 E60A                      <1>       OUT    DMA+10, AL          ; INITIALIZE THE DISKETTE CHANNEL
1888 00004C48 C3                        <1>       retn
1889                                    <1>
1890                                    <1> ;; 08/02/2015 - Protected Mode Modification
1891                                    <1> ;;    MOV    AL, 04AH             ; WILL WRITE TO THE DISKETTE
1892                                    <1> ;;    CLI                         ; DISABLE INTERRUPTS DURING DMA SET-UP
1893                                    <1> ;;    OUT    DMA+12,AL            ; SET THE FIRST/LA5T F/F
1894                                    <1> ;;    ;JMP   $+2                  ; WAIT FOR I/O
1895                                    <1> ;;    IODELAY
1896                                    <1> ;;    OUT    DMA+11,AL            ; OUTPUT THE MODE BYTE
1897                                    <1> ;;    ;MOV   AX,ES                ; GET THE ES VALUE
1898                                    <1> ;;    ;ROL   AX,4                 ; ROTATE LEFT
1899                                    <1> ;;    ;MOV   CH,AL                ; GET HIGHEST NIBBLE OF ES TO CH
1900                                    <1> ;;    ;AND   AL,11110000B         ; ZERO THE LOW NIBBLE FROM SEGMENT
1901                                    <1> ;;    ;ADD   AX,[BP+2]            ; TEST FOR CARRY FROM ADDITION
1902                                    <1> ;;    ;JNC   short J33A
1903                                    <1> ;;    ;INC   CH                   ; CARRY MEANS HIGH 4 BITS MUST BE INC
1904                                    <1> ;;    mov    eax, [ebp+4] ; 08/02/2015
1905                                    <1> ;;;J33A:
1906                                    <1> ;;    PUSH   eAX ; 08/02/2015    ; SAVE START ADDRESS
1907                                    <1> ;;    OUT    DMA+4,AL             ; OUTPUT LOW ADDRESS
1908                                    <1> ;;    ;JMP   $+2                  ; WAIT FOR I/O
1909                                    <1> ;;    IODELAY
1910                                    <1> ;;    MOV    AL,AH
1911                                    <1> ;;    OUT    DMA+4,AL             ; OUTPUT HIGH ADDRESS
1912                                    <1> ;;    shr    eax, 16 ; 08/02/2015
1913                                    <1> ;;    ;MOV   AL,CH                ; GET HIGH 4 BITS
1914                                    <1> ;;    ;JMP   $+2                  ; I/O WAIT STATE
1915                                    <1> ;;    IODELAY
1916                                    <1> ;;    ;AND   AL,00001111B
1917                                    <1> ;;    OUT    081H,AL                   ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
1918                                    <1> ;;
1919                                    <1> ;;;----- DETERMINE COUNT
1920                                    <1> ;;    sub    eax, eax ; 08/02/2015
1921                                    <1> ;;    MOV    DL, 4                ; SECTORS/TRACK VALUE IN PARM TABLE
1922                                    <1> ;;    CALL   GET_PARM             ; "
1923                                    <1> ;;    XCHG   AL, AH               ; AL = SECTORS/TRACK VALUE
1924                                    <1> ;;    SUB    AH, AH               ; AX = SECTORS/TRACK VALUE
1925                                    <1> ;;    SHL    AX, 2                ; AX = SEC/TRK * 4 (OFFSET C,H,R,N)
1926                                    <1> ;;    DEC    AX                   ; -1 FOR DMA VALUE
1927                                    <1> ;;    PUSH   eAX    ; 08/02/2015 ; SAVE # OF BYTES TO BE TRANSFERED
1928                                    <1> ;;    OUT    DMA+5,AL             ; LOW BYTE OF COUNT
1929                                    <1> ;;    ;JMP   $+2                  ; WAIT FOR I/O
1930                                    <1> ;;    IODELAY
1931                                    <1> ;;    MOV    AL, AH
1932                                    <1> ;;    OUT    DMA+5,AL             ; HIGH BYTE OF COUNT
1933                                    <1> ;;    STI                         ; RE-ENABLE INTERRUPTS
1934                                    <1> ;;    POP    eCX    ; 08/02/2015 ; RECOVER COUNT VALUE
1935                                    <1> ;;    POP    eAX    ; 08/02/2015 ; RECOVER ADDRESS VALUE
1936                                    <1> ;;    ;ADD   AX, CX               ; ADD, TEST FOR 64K OVERFLOW
1937                                    <1> ;;    add    ecx, eax ; 08/02/2015
1938                                    <1> ;;    MOV    AL, 2                ; MODE FOR 8237
1939                                    <1> ;;    ;JMP   $+2                  ; WAIT FOR I/O
1940                                    <1> ;;    SIODELAY
1941                                    <1> ;;    OUT    DMA+10, AL           ; INITIALIZE THE DISKETTE CHANNEL
1942                                    <1> ;;    ;JNC   short FMTDMA_OK            ; CHECK FOR ERROR
1943                                    <1> ;;    jc     short fmtdma_bnd_err ; 08/02/2015
1944                                    <1> ;;    and    ecx, 0FFF00000h  ; 16 MB limit
1945                                    <1> ;;    jz     short FMTDMA_OK
1946                                    <1> ;;    stc    ; 20/02/2015
1947                                    <1> ;;fmtdma_bnd_err:
1948                                    <1> ;;    MOV    byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
1949                                    <1> ;;FMTDMA_OK:
1950                                    <1> ;;    RETn                        ; CY SET BY ABOVE IF ERROR
1951                                    <1>
1952                                    <1> ;--------------------------------------------------------------------------------
1953                                    <1> ; NEC_INIT
1954                                    <1> ;     THIS ROUTINE SEEKS TO THE REQUESTED TRACK AND INITIALIZES
1955                                    <1> ;     THE NEC FOR THE READ/WRITE/VERIFY/FORMAT OPERATION.
1956                                    <1> ;
1957                                    <1> ; ON ENTRY:  AH = NEC COMMAND TO BE PERFORMED
1958                                    <1> ;
1959                                    <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
```

```
1960                            <1> ;--------------------------------------------------------------------------------
1961                            <1> NEC_INIT:
1962 00004C49 6650             <1>       PUSH   AX                  ; SAVE NEC COMMAND
1963 00004C4B E8BC020000       <1>       CALL   MOTOR_ON            ; TURN MOTOR ON FOR SPECIFIC DRIVE
1964                            <1>
1965                            <1> ;-----      DO THE SEEK OPERATION
1966                            <1>
1967 00004C50 8A6D01           <1>       MOV    CH,[eBP+1]          ; CH = TRACK #
1968 00004C53 E8AF030000       <1>       CALL   SEEK                ; MOVE TO CORRECT TRACK
1969 00004C58 6658             <1>       POP    AX                  ; RECOVER COMMAND
1970 00004C5A 721E             <1>       JC     short ER_1          ; ERROR ON SEEK
1971 00004C5C BB[7A4C0000]     <1>       MOV    eBX, ER_1           ; LOAD ERROR ADDRESS
1972 00004C61 53               <1>       PUSH   eBX                 ; PUSH NEC_OUT ERROR RETURN
1973                            <1>
1974                            <1> ;-----      SEND OUT THE PARAMETERS TO THE CONTROLLER
1975                            <1>
1976 00004C62 E866030000       <1>       CALL   NEC_OUTPUT          ; OUTPUT THE OPERATION COMMAND
1977 00004C67 6689F0           <1>       MOV    AX,SI               ; AH = HEAD #
1978 00004C6A 89FB             <1>       MOV    eBX,eDI                   ; BL = DRIVE #
1979 00004C6C C0E402           <1>       SAL    AH,2                ; MOVE IT TO BIT 2
1980 00004C6F 80E404           <1>       AND    AH,00000100B        ; ISOLATE THAT BIT
1981 00004C72 08DC             <1>       OR     AH,BL               ; OR IN THE DRIVE NUMBER
1982 00004C74 E854030000       <1>       CALL   NEC_OUTPUT          ; FALL THRU CY SET IF ERROR
1983 00004C79 5B               <1>       POP    eBX                 ; THROW AWAY ERROR RETURN
1984                            <1> ER_1:
1985 00004C7A C3               <1>       RETn
1986                            <1>
1987                            <1> ;--------------------------------------------------------------------------------
1988                            <1> ; RWV_COM
1989                            <1> ;     THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC TO THE
1990                            <1> ;     READ/WRITE/VERIFY OPERATIONS.
1991                            <1> ;
1992                            <1> ; ON ENTRY:  CS:BX = ADDRESS OF MEDIA/DRIVE PARAMETER TABLE
1993                            <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1994                            <1> ;--------------------------------------------------------------------------------
1995                            <1> RWV_COM:
1996 00004C7B B8[C64C0000]     <1>       MOV    eAX, ER_2           ; LOAD ERROR ADDRESS
1997 00004C80 50               <1>       PUSH   eAX                 ; PUSH NEC_OUT ERROR RETURN
1998 00004C81 8A6501           <1>       MOV    AH,[eBP+1]          ; OUTPUT TRACK #
1999 00004C84 E844030000       <1>       CALL   NEC_OUTPUT
2000 00004C89 6689F0           <1>       MOV    AX,SI               ; OUTPUT HEAD #
2001 00004C8C E83C030000       <1>       CALL   NEC_OUTPUT
2002 00004C91 8A6500           <1>         MOV     AH,[eBP]                ; OUTPUT SECTOR #
2003 00004C94 E834030000       <1>       CALL   NEC_OUTPUT
2004 00004C99 B203             <1>       MOV    DL,3                ; BYTES/SECTOR PARAMETER FROM BLOCK
2005 00004C9B E827020000       <1>       CALL   GET_PARM            ; ... TO THE NEC
2006 00004CA0 E828030000       <1>       CALL   NEC_OUTPUT          ; OUTPUT TO CONTROLLER
2007 00004CA5 B204             <1>       MOV    DL,4                ; EOT PARAMETER FROM BLOCK
2008 00004CA7 E81B020000       <1>       CALL   GET_PARM            ; ... TO THE NEC
2009 00004CAC E81C030000       <1>       CALL   NEC_OUTPUT          ; OUTPUT TO CONTROLLER
2010 00004CB1 8A6305           <1>         MOV     AH, [eBX+MD.GAP]        ; GET GAP LENGTH
2011                            <1> _R15:
2012 00004CB4 E814030000       <1>       CALL   NEC_OUTPUT
2013 00004CB9 B206             <1>       MOV    DL,6                ; DTL PARAMETER PROM BLOCK
2014 00004CBB E807020000       <1>       CALL   GET_PARM            ;  TO THE NEC
2015 00004CC0 E808030000       <1>       CALL   NEC_OUTPUT          ; OUTPUT TO CONTROLLER
2016 00004CC5 58               <1>       POP    eAX                 ; THROW AWAY ERROR EXIT
2017                            <1> ER_2:
2018 00004CC6 C3               <1>       RETn
2019                            <1>
2020                            <1> ;--------------------------------------------------------------------------------
2021                            <1> ; NEC_TERM
2022                            <1> ;     THIS ROUTINE WAITS FOR THE OPERATION THEN ACCEPTS THE STATUS
2023                            <1> ;     FROM THE NEC FOR THE READ/WRITE/VERIFY/FORWAT OPERATION.
2024                            <1> ;
2025                            <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
2026                            <1> ;--------------------------------------------------------------------------------
2027                            <1> NEC_TERM:
2028                            <1>
2029                            <1> ;-----      LET THE OPERATION HAPPEN
2030                            <1>
2031 00004CC7 56               <1>       PUSH   eSI                 ; SAVE HEAD #, # OF SECTORS
2032 00004CC8 E80D040000       <1>       CALL   WAIT_INT            ; WAIT FOR THE INTERRUPT
2033 00004CCD 9C               <1>       PUSHF
2034 00004CCE E837040000       <1>       CALL   RESULTS                   ; GET THE NEC STATUS
2035 00004CD3 724B             <1>       JC     short SET_END_POP
2036 00004CD5 9D               <1>       POPF
2037 00004CD6 723E             <1>       JC     short SET_END       ; LOOK FOR ERROR
2038                            <1>
2039                            <1> ;-----      CHECK THE RESULTS RETURNED BY THE CONTROLLER
2040                            <1>
2041 00004CD8 FC               <1>       CLD                        ; SET THE CORRECT DIRECTION
2042 00004CD9 BE[498A0100]     <1>       MOV    eSI, NEC_STATUS           ; POINT TO STATUS FIELD
2043 00004CDE AC               <1>       lodsb                      ; GET ST0
2044 00004CDF 24C0             <1>       AND    AL,11000000B        ; TEST FOR NORMAL TERMINATION
2045 00004CE1 7433             <1>       JZ     short SET_END
2046 00004CE3 3C40             <1>       CMP    AL,01000000B        ; TEST FOR ABNORMAL TERMINATION
2047 00004CE5 7527             <1>       JNZ    short J18           ; NOT ABNORMAL, BAD NEC
2048                            <1>
2049                            <1> ;-----      ABNORMAL TERMINATION, FIND OUT WHY
2050                            <1>
2051 00004CE7 AC               <1>       lodsb                      ; GET ST1
2052 00004CE8 D0E0             <1>       SAL    AL,1                ; TEST FOR EDT FOUND
2053 00004CEA B404             <1>       MOV    AH,RECORD_NOT_FND
2054 00004CEC 7222             <1>       JC     short J19
2055 00004CEE C0E002           <1>       SAL    AL,2
2056 00004CF1 B410             <1>       MOV    AH,BAD_CRC
2057 00004CF3 721B             <1>       JC     short J19
2058 00004CF5 D0E0             <1>       SAL    AL,1                ; TEST FOR DMA OVERRUN
2059 00004CF7 B408             <1>       MOV    AH,BAD_DMA
2060 00004CF9 7215             <1>       JC     short J19
2061 00004CFB C0E002           <1>       SAL    AL,2                ; TEST FOR RECORD NOT FOUND
2062 00004CFE B404             <1>       MOV    AH,RECORD_NOT_FND
2063 00004D00 720E             <1>       JC     short J19
2064 00004D02 D0E0             <1>       SAL    AL,1
```

```
2065 00004D04 B403                <1>       MOV    AH,WRITE_PROTECT    ; TEST FOR WRITE_PROTECT
2066 00004D06 7208                <1>       JC     short J19
2067 00004D08 D0E0                <1>       SAL    AL,1                ; TEST MISSING ADDRESS MARK
2068 00004D0A B402                <1>       MOV    AH,BAD_ADDR_MARK
2069 00004D0C 7202                <1>       JC     short J19
2070                              <1>
2071                              <1> ;-----      NEC MUST HAVE FAILED
2072                              <1> J18:
2073 00004D0E B420                <1>       MOV    AH,BAD_NEC
2074                              <1> J19:
2075 00004D10 0825[488A0100]      <1>       OR     [DSKETTE_STATUS], AH
2076                              <1> SET_END:
2077 00004D16 803D[488A0100]01    <1>       CMP    byte [DSKETTE_STATUS], 1 ; SET ERROR CONDITION
2078 00004D1D F5                  <1>       CMC
2079 00004D1E 5E                  <1>       POP    eSI
2080 00004D1F C3                  <1>       RETn                       ; RESTORE HEAD #, # OF SECTORS
2081                              <1>
2082                              <1> SET_END_POP:
2083 00004D20 9D                  <1>       POPF
2084 00004D21 EBF3                <1>       JMP    SHORT SET_END
2085                              <1>
2086                              <1> ;--------------------------------------------------------------------------
2087                              <1> ; DSTATE:   ESTABLISH STATE UPON SUCCESSFUL OPERATION.
2088                              <1> ;--------------------------------------------------------------------------
2089                              <1> DSTATE:
2090 00004D23 803D[488A0100]00    <1>       CMP    byte [DSKETTE_STATUS],0   ; CHECK FOR ERROR
2091 00004D2A 753E                <1>       JNZ    short SETBAC            ; IF ERROR JUMP
2092 00004D2C 808F[558A0100]10    <1>       OR     byte [DSK_STATE+eDI],MED_DET ; NO ERROR, MARK MEDIA AS DETERMINED
2093 00004D33 F687[558A0100]04    <1>       TEST   byte [DSK_STATE+eDI],DRV_DET ; DRIVE DETERMINED ?
2094 00004D3A 752E                <1>       JNZ    short SETBAC        ; IF DETERMINED NO TRY TO DETERMINE
2095 00004D3C 8A87[558A0100]      <1>       MOV    AL,[DSK_STATE+eDI]  ; LOAD STATE
2096 00004D42 24C0                <1>       AND    AL,RATE_MSK         ; KEEP ONLY RATE
2097 00004D44 3C80                <1>       CMP    AL,RATE_250         ; RATE 250 ?
2098 00004D46 751B                <1>       JNE    short M_12          ; NO, MUST BE 1.2M OR 1.44M DRIVE
2099                              <1>
2100                              <1> ;-----      CHECK IF IT IS 1.44M
2101                              <1>
2102 00004D48 E871010000          <1>       CALL   CMOS_TYPE           ; RETURN DRIVE TYPE IN (AL)
2103                              <1>       ;;20/02/2015
2104                              <1>       ;;JC   short M_12          ; CMOS BAD
2105 00004D4D 7414                <1>       jz     short M_12 ;; 20/02/2015
2106 00004D4F 3C04                <1>       CMP    AL, 4               ; 1.44MB DRIVE ?
2107 00004D51 7410                <1>       JE     short M_12          ; YES
2108                              <1> M_720:
2109 00004D53 80A7[558A0100]FD    <1>       AND    byte [DSK_STATE+eDI], ~FMT_CAPA ; TURN OFF FORMAT CAPABILITY
2110 00004D5A 808F[558A0100]04    <1>       OR     byte [DSK_STATE+eDI],DRV_DET  ; MARK DRIVE DETERMINED
2111 00004D61 EB07                <1>       JMP    SHORT SETBAC        ; BACK
2112                              <1> M_12:
2113 00004D63 808F[558A0100]06    <1>       OR     byte [DSK_STATE+eDI],DRV_DET+FMT_CAPA
2114                              <1>                                  ; TURN ON DETERMINED & FMT CAPA
2115                              <1> SETBAC:
2116 00004D6A C3                  <1>       RETn
2117                              <1>
2118                              <1> ;--------------------------------------------------------------------------
2119                              <1> ; RETRY
2120                              <1> ;     DETERMINES WHETHER A RETRY IS NECESSARY.
2121                              <1> ;     IF RETRY IS REQUIRED THEN STATE INFORMATION IS UPDATED FOR RETRY.
2122                              <1> ;
2123                              <1> ; ON EXIT:  CY = 1 FOR RETRY, CY = 0 FOR NO RETRY
2124                              <1> ;--------------------------------------------------------------------------
2125                              <1> RETRY:
2126 00004D6B 803D[488A0100]00    <1>       CMP    byte [DSKETTE_STATUS],0   ; GET STATUS OF OPERATION
2127 00004D72 7445                <1>       JZ     short NO_RETRY            ; SUCCESSFUL OPERATION
2128 00004D74 803D[488A0100]80    <1>       CMP    byte [DSKETTE_STATUS],TIME_OUT ; IF TIME OUT NO RETRY
2129 00004D7B 743C                <1>       JZ     short NO_RETRY
2130 00004D7D 8AA7[558A0100]      <1>       MOV    AH,[DSK_STATE+eDI]  ; GET MEDIA STATE OF DRIVE
2131 00004D83 F6C410              <1>       TEST   AH,MED_DET          ; ESTABLISHED/DETERMINED ?
2132 00004D86 7531                <1>       JNZ    short NO_RETRY            ; IF ESTABLISHED STATE THEN TRUE ERROR
2133 00004D88 80E4C0              <1>       AND    AH,RATE_MSK         ; ISOLATE RATE
2134 00004D8B 8A2D[508A0100]      <1>       MOV    CH,[LASTRATE]       ; GET START OPERATION STATE
2135 00004D91 C0C504              <1>       ROL    CH,4                ; TO CORRESPONDING BITS
2136 00004D94 80E5C0              <1>       AND    CH,RATE_MSK         ; ISOLATE RATE BITS
2137 00004D97 38E5                <1>       CMP    CH,AH               ; ALL RATES TRIED
2138 00004D99 741E                <1>       JE     short NO_RETRY            ; IF YES, THEN TRUE ERROR
2139                              <1>
2140                              <1> ;     SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE
2141                              <1> ;      00000000B (500) -> 10000000B    (250)
2142                              <1> ;      10000000B (250) -> 01000000B    (300)
2143                              <1> ;      01000000B (300) -> 00000000B    (500)
2144                              <1>
2145 00004D9B 80FC01              <1>       CMP    AH,RATE_500+1       ; SET CY FOR RATE 500
2146 00004D9E D0DC                <1>       RCR    AH,1                ; TO NEXT STATE
2147 00004DA0 80E4C0              <1>       AND    AH,RATE_MSK         ; KEEP ONLY RATE BITS
2148 00004DA3 80A7[558A0100]1F    <1>       AND    byte [DSK_STATE+eDI], ~(RATE_MSK+DBL_STEP)
2149                              <1>                                  ; RATE, DBL STEP OFF
2150 00004DAA 08A7[558A0100]      <1>       OR     [DSK_STATE+eDI],AH  ; TURN ON NEW RATE
2151 00004DB0 C605[488A0100]00    <1>       MOV    byte [DSKETTE_STATUS],0   ; RESET STATUS FOR RETRY
2152 00004DB7 F9                  <1>       STC                        ; SET CARRY FOR RETRY
2153 00004DB8 C3                  <1>       RETn                       ; RETRY RETURN
2154                              <1>
2155                              <1> NO_RETRY:
2156 00004DB9 F8                  <1>       CLC                        ; CLEAR CARRY NO RETRY
2157 00004DBA C3                  <1>       RETn                       ; NO RETRY RETURN
2158                              <1>
2159                              <1> ;--------------------------------------------------------------------------
2160                              <1> ; NUM_TRANS
2161                              <1> ;     THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT WERE
2162                              <1> ;     ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.
2163                              <1> ;
2164                              <1> ; ON ENTRY:  [BP+1] = TRACK
2165                              <1> ;            SI-HI  = HEAD
2166                              <1> ;            [BP]   = START SECTOR
2167                              <1> ;
2168                              <1> ; ON EXIT:   AL = NUMBER ACTUALLY TRANSFERRED
2169                              <1> ;--------------------------------------------------------------------------
```

```
                        <1> NUM_TRANS:
00004DBB 30C0           <1>         XOR     AL,AL               ; CLEAR FOR ERROR
00004DBD 803D[488A0100]00 <1>       CMP     byte [DSKETTE_STATUS],0    ; CHECK FOR ERROR
00004DC4 752C           <1>         JNZ     NT_OUT              ; IF ERROR 0 TRANSFERRED
00004DC6 B204           <1>         MOV     DL,4                ; SECTORS/TRACK OFFSET TO DL
00004DC8 E8FA000000     <1>         CALL    GET_PARM            ; AH = SECTORS/TRACK
00004DCD 8A1D[4E8A0100] <1>         MOV     BL, [NEC_STATUS+5]  ; GET ENDING SECTOR
00004DD3 6689F1         <1>         MOV     CX,SI               ; CH = HEAD # STARTED
00004DD6 3A2D[4D8A0100] <1>         CMP     CH, [NEC_STATUS+4]  ; GET HEAD ENDED UP ON
00004DDC 750D           <1>         JNZ     DIF_HD              ; IF ON SAME HEAD, THEN NO ADJUST
00004DDE 8A2D[4C8A0100] <1>         MOV     CH, [NEC_STATUS+3]  ; GET TRACK ENDED UP ON
00004DE4 3A6D01         <1>         CMP     CH,[eBP+1]          ; IS IT ASKED FOR TRACK
00004DE7 7404           <1>         JZ      short SAME_TRK            ; IF SAME TRACK NO INCREASE
00004DE9 00E3           <1>         ADD     BL,AH               ; ADD SECTORS/TRACK
                        <1> DIF_HD:
00004DEB 00E3           <1>         ADD     BL,AH               ; ADD SECTORS/TRACK
                        <1> SAME_TRK:
00004DED 2A5D00         <1>         SUB     BL,[eBP]            ; SUBTRACT START FROM END
00004DF0 88D8           <1>         MOV     AL,BL               ; TO AL
                        <1> NT_OUT:
00004DF2 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:
00004DF3 B202           <1>         MOV     DL,2                ; GET THE MOTOR WAIT PARAMETER
00004DF5 6650           <1>         PUSH    AX                  ; SAVE NUMBER TRANSFERRED
00004DF7 E8CB000000     <1>         CALL    GET_PARM
00004DFC 8825[478A0100] <1>         MOV     [MOTOR_COUNT],AH    ; STORE UPON RETURN
00004E02 6658           <1>         POP     AX                  ; RESTORE NUMBER TRANSFERRED
00004E04 8A25[488A0100] <1>         MOV     AH, [DSKETTE_STATUS]      ; GET STATUS OF OPERATION
00004E0A 08E4           <1>         OR      AH,AH               ; CHECK FOR ERROR
00004E0C 7402           <1>         JZ      short NUN_ERR       ; NO ERROR
00004E0E 30C0           <1>         XOR     AL,AL               ; CLEAR NUMBER RETURNED
                        <1> NUN_ERR:
00004E10 80FC01         <1>         CMP     AH,1                ; SET THE CARRY FLAG TO INDICATE
00004E13 F5             <1>         CMC                         ; SUCCESS OR FAILURE
00004E14 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:
00004E15 8AA7[558A0100] <1>         MOV     AH, [DSK_STATE+eDI] ; ACCESS STATE
00004E1B F6C410         <1>         TEST    AH,MED_DET          ; ESTABLISHED STATE ?
00004E1E 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>
00004E20 C605[458A0100]00 <1>       MOV     byte [SEEK_STATUS],0      ; SET RECALIBRATE REQUIRED ON ALL DRIVES
00004E27 E8E0000000     <1>         CALL    MOTOR_ON            ; ENSURE MOTOR STAY ON
00004E2C B500           <1>         MOV     CH,0                ; LOAD TRACK 0
00004E2E E8D4010000     <1>         CALL    SEEK                ; SEEK TO TRACK 0
00004E33 E868000000     <1>         CALL    READ_ID                   ; READ ID FUNCTION
00004E38 7249           <1>         JC      short SD_ERR        ; IF ERROR NO TRACK 0
                        <1>
                        <1> ;-----      INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)
                        <1>
00004E3A 66B95004       <1>         MOV     CX,0450H            ; START, MAX TRACKS
00004E3E F687[558A0100]01 <1>       TEST    byte [DSK_STATE+eDI],TRK_CAPA ; TEST FOR 80 TRACK CAPABILITY
00004E45 7402           <1>         JZ      short CNT_OK        ; IF NOT COUNT IS SETUP
00004E47 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:
00004E49 C605[478A0100]FF <1>         MOV     byte [MOTOR_COUNT], 0FFH ; ENSURE MOTOR STAYS ON FOR OPERATION
00004E50 6651           <1>         PUSH    CX                  ; SAVE TRACK, COUNT
00004E52 C605[488A0100]00 <1>       MOV     byte [DSKETTE_STATUS],0   ; CLEAR STATUS, EXPECT ERRORS
00004E59 6631C0         <1>         XOR     AX,AX               ; CLEAR AX
00004E5C D0ED           <1>         SHR     CH,1                ; HALVE TRACK, CY = HEAD
00004E5E C0D003         <1>         RCL     AL,3                ; AX = HEAD IN CORRECT BIT
00004E61 6650           <1>         PUSH    AX                  ; SAVE HEAD
00004E63 E89F010000     <1>         CALL    SEEK                ; SEEK TO TRACK
00004E68 6658           <1>         POP     AX                  ; RESTORE HEAD
00004E6A 6609C7         <1>         OR      DI,AX               ; DI = HEAD OR'ED DRIVE
00004E6D E82E000000     <1>         CALL    READ_ID                   ; READ ID HEAD 0
00004E72 9C             <1>         PUSHF                       ; SAVE RETURN FROM READ_ID
00004E73 6681E7FB00     <1>         AND     DI,11111011B        ; TURN OFF HEAD 1 BIT
00004E78 9D             <1>         POPF                        ; RESTORE ERROR RETURN
00004E79 6659           <1>         POP     CX                  ; RESTORE COUNT
00004E7B 7308           <1>         JNC     short DO_CHK        ; IF OK, ASKED = RETURNED TRACK ?
00004E7D FEC5           <1>         INC     CH                  ; INC FOR NEXT TRACK
00004E7F 38CD           <1>         CMP     CH,CL               ; REACHED MAXIMUM YET
00004E81 75C6           <1>         JNZ     short CNT_OK        ; CONTINUE TILL ALL TRIED
                        <1>
                        <1> ;-----      FALL THRU, READ ID FAILED FOR ALL TRACKS
                        <1>
                        <1> SD_ERR:
00004E83 F9             <1>         STC                         ; SET CARRY FOR ERROR
00004E84 C3             <1>         RETn                        ; SETUP_DBL ERROR EXIT
                        <1>
```

```
                            <1> DO_CHK:
00004E85 8A0D[4C8A0100]     <1>       MOV    CL, [NEC_STATUS+3]  ; LOAD RETURNED TRACK
00004E8B 888F[598A0100]     <1>       MOV    [DSK_TRK+eDI], CL   ; STORE TRACK NUMBER
00004E91 D0ED               <1>       SHR    CH,1                ; HALVE TRACK
00004E93 38CD               <1>       CMP    CH,CL               ; IS IT THE SAME AS ASKED FOR TRACK
00004E95 7407               <1>       JZ     short NO_DBL        ; IF SAME THEN NO DOUBLE STEP
00004E97 808F[558A0100]20   <1>       OR     byte [DSK_STATE+eDI],DBL_STEP ; TURN ON DOUBLE STEP REQUIRED
                            <1> NO_DBL:
00004E9E F8                 <1>       CLC                        ; CLEAR ERROR FLAG
00004E9F 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:
00004EA0 B8[BD4E0000]       <1>       MOV    eAX, ER_3           ; MOVE NEC OUTPUT ERROR ADDRESS
00004EA5 50                 <1>       PUSH   eAX
00004EA6 B44A               <1>       MOV    AH,4AH              ; READ ID COMMAND
00004EA8 E820010000         <1>       CALL   NEC_OUTPUT          ; TO CONTROLLER
00004EAD 6689F8             <1>       MOV    AX,DI               ; DRIVE # TO AH, HEAD 0
00004EB0 88C4               <1>       MOV    AH,AL
00004EB2 E816010000         <1>       CALL   NEC_OUTPUT          ; TO CONTROLLER
00004EB7 E80BFEFFFF         <1>       CALL   NEC_TERM            ; WAIT FOR OPERATION, GET STATUS
00004EBC 58                 <1>       POP    eAX                 ; THROW AWAY ERROR ADDRESS
                            <1> ER_3:
00004EBD 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
00004EBE 8A87[FE6D0000]     <1> mov   al, [eDI+fd0_type]
00004EC4 20C0               <1> and   al, al ; 18/12/2014
00004EC6 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
00004EC7 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)
00004EC8 87D3               <1>       XCHG   eDX,eBX                   ; BL = INDEX
                            <1>       ;SUB   BH,BH               ; BX = INDEX
00004ECA 81E3FF000000       <1>       and    ebx, 0FFh
                            <1>       ;LDS   SI, [DISK_POINTER]  ; POINT TO BLOCK
                            <1>       ;
                            <1>       ; 17/12/2014
00004ED0 66A1[ED6D0000]     <1>       mov    ax, [cfd] ; current (AL) and previous fd (AH)
00004ED6 38E0               <1>       cmp    al, ah
00004ED8 7425               <1>       je     short gpndc
00004EDA A2[EE6D0000]       <1>       mov    [pfd], al ; current drive -> previous drive
00004EDF 53                 <1>       push   ebx ; 08/02/2015
00004EE0 88C3               <1>       mov    bl, al
                            <1>       ; 11/12/2014
00004EE2 8A83[FE6D0000]     <1>       mov    al, [eBX+fd0_type]  ; Drive type (0,1,2,3,4)
                            <1>       ; 18/12/2014
00004EE8 20C0               <1>       and    al, al
00004EEA 7507               <1>       jnz    short gpdtc
00004EEC BB[D76D0000]       <1>       mov    ebx, MD_TBL6        ; 1.44 MB param. tbl. (default)
00004EF1 EB05               <1>         jmp     short gpdpu
                            <1> gpdtc:
00004EF3 E817F9FFFF         <1>       call   DR_TYPE_CHECK
                            <1>       ; cf = 1 -> eBX points to 1.44MB fd parameter table (default)
```

```
                        <1> gpdpu:
00004EF8 891D[746D0000] <1>       mov    [DISK_POINTER], ebx
00004EFE 5B             <1>       pop    ebx
                        <1> gpndc:
00004EFF 8B35[746D0000] <1>       mov    esi, [DISK_POINTER] ; 08/02/2015, si -> esi
00004F05 8A241E         <1>       MOV    AH, [eSI+eBX]       ; GET THE WORD
00004F08 87D3           <1>       XCHG   eDX,eBX                   ; RESTORE BX
00004F0A 5E             <1>       POP    eSI
                        <1>       ;POP   DS
00004F0B 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:
00004F0C 53             <1>       PUSH   eBX                ; SAVE REG.
00004F0D E82A000000     <1>       CALL   TURN_ON                   ; TURN ON MOTOR
00004F12 7226           <1>       JC     short MOT_IS_ON           ; IF CY=1 NO WAIT
00004F14 E89BF9FFFF     <1>       CALL   XLAT_OLD           ; TRANSLATE STATE TO COMPATIBLE MODE
00004F19 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:
00004F1E B20A           <1>       MOV    DL,10              ; GET THE MOTOR WAIT PARAMETER
00004F20 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
00004F25 80FC08         <1>       cmp    ah, 8
                        <1>       ;JAE   short GP2          ; IF YES, CONTINUE
00004F28 7702           <1>       ja     short J13
                        <1>       ;MOV   AL,8               ; ONE SECOND WAIT FOR MOTOR START UP
00004F2A 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)
00004F2C B95E200000     <1>       MOV    eCX,8286           ; COUNT FOR 1/8 SECOND AT 15.085737 US
00004F31 E810D5FFFF     <1>       CALL   WAITF              ; GO TO FIXED WAIT ROUTINE
                        <1>       ;DEC   AL                 ; DECREMENT TIME VALUE
00004F36 FECC           <1>       dec    ah
00004F38 75F2           <1>       JNZ    short J13          ; ARE WE DONE YET
                        <1> MOT_IS_ON:
00004F3A 5B             <1>       POP    eBX                ; RESTORE REG.
00004F3B 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:
00004F3C 89FB           <1>       MOV    eBX,eDI                   ; BX = DRIVE #
00004F3E 88D9           <1>       MOV    CL,BL              ; CL = DRIVE #
00004F40 C0C304         <1>       ROL    BL,4               ; BL = DRIVE SELECT
00004F43 FA             <1>       CLI                       ; NO INTERRUPTS WHILE DETERMINING STATUS
00004F44 C605[478A0100]FF <1>     MOV    byte [MOTOR_COUNT],0FFH   ; ENSURE MOTOR STAYS ON FOR OPERATION
00004F4B A0[468A0100]   <1>       MOV    AL, [MOTOR_STATUS] ; GET DIGITAL OUTPUT REGISTER REFLECTION
00004F50 2430           <1>       AND    AL,00110000B       ; KEEP ONLY DRIVE SELECT BITS
00004F52 B401           <1>       MOV    AH,1               ; MASK FOR DETERMINING MOTOR BIT
00004F54 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>
00004F56 38D8           <1>       CMP    AL,BL              ; REQUESTED DRIVE ALREADY SELECTED ?
00004F58 7508           <1>       JNZ    short TURN_IT_ON   ; IF NOT SELECTED JUMP
00004F5A 8425[468A0100] <1>       TEST   AH, [MOTOR_STATUS] ; TEST MOTOR ON BIT
00004F60 7535           <1>       JNZ    short NO_MOT_WAIT  ; JUMP IF MOTOR ON AND SELECTED
                        <1>
                        <1> TURN_IT_ON:
00004F62 08DC           <1>       OR     AH,BL              ; AH = DRIVE SELECT AND MOTOR ON
00004F64 8A3D[468A0100] <1>       MOV    BH,[MOTOR_STATUS]  ; SAVE COPY OF @MOTOR_STATUS BEFORE
00004F6A 80E70F         <1>       AND    BH,00001111B       ; KEEP ONLY MOTOR BITS
00004F6D 8025[468A0100]CF <1>     AND    byte [MOTOR_STATUS],11001111B ; CLEAR OUT DRIVE SELECT
00004F74 0825[468A0100] <1>       OR     [MOTOR_STATUS],AH  ; OR IN DRIVE SELECTED AND MOTOR ON
00004F7A A0[468A0100]   <1>       MOV    AL,[MOTOR_STATUS]  ; GET DIGITAL OUTPUT REGISTER REFLECTION
00004F7F 88C3           <1>       MOV    BL,AL              ; BL=@MOTOR_STATUS AFTER, BH=BEFORE
00004F81 80E30F         <1>       AND    BL,00001111B       ; KEEP ONLY MOTOR BITS
00004F84 FB             <1>       STI                       ; ENABLE INTERRUPTS AGAIN
00004F85 243F           <1>       AND    AL,00111111B       ; STRIP AWAY UNWANTED BITS
00004F87 C0C004         <1>       ROL    AL,4               ; PUT BITS IN DESIRED POSITIONS
```

```
2485 00004F8A 0C0C              <1>       OR     AL,00001100B        ; NO RESET, ENABLE DMA/INTERRUPT
2486 00004F8C 66BAF203          <1>       MOV    DX,03F2H            ; SELECT DRIVE AND TURN ON MOTOR
2487 00004F90 EE                <1>       OUT    DX,AL
2488 00004F91 38FB              <1>       CMP    BL,BH               ; NEW MOTOR TURNED ON ?
2489                            <1>       ;JZ    short NO_MOT_WAIT   ; NO WAIT REQUIRED IF JUST SELECT
2490 00004F93 7403              <1>       je     short no_mot_w1 ; 27/02/2015
2491 00004F95 F8                <1>       CLC                        ; (re)SET CARRY MEANING WAIT
2492 00004F96 C3                <1>       RETn
2493                            <1>
2494                            <1> NO_MOT_WAIT:
2495 00004F97 FB                <1>       sti
2496                            <1> no_mot_w1: ; 27/02/2015
2497 00004F98 F9                <1>       STC                        ; SET NO WAIT REQUIRED
2498                            <1>       ;STI                       ; INTERRUPTS BACK ON
2499 00004F99 C3                <1>       RETn
2500                            <1>
2501                            <1> ;--------------------------------------------------------------------------
2502                            <1> ; HD_WAIT
2503                            <1> ;     WAIT FOR HEAD SETTLE TIME.
2504                            <1> ;
2505                            <1> ; ON ENTRY:  DI = DRIVE #
2506                            <1> ;
2507                            <1> ; ON EXIT:   AX,BX,CX,DX DESTROYED
2508                            <1> ;--------------------------------------------------------------------------
2509                            <1> HD_WAIT:
2510 00004F9A B209              <1>       MOV    DL,9                ; GET HEAD SETTLE PARAMETER
2511 00004F9C E826FFFFFF        <1>       CALL   GET_PARM
2512 00004FA1 08E4              <1>       or     ah, ah ; 17/12/2014
2513 00004FA3 7519              <1>       jnz    short DO_WAT
2514 00004FA5 F605[468A0100]80  <1>         TEST    byte [MOTOR_STATUS],10000000B ; SEE IF A WRITE OPERATION
2515                            <1>       ;JZ    short ISNT_WRITE    ; IF NOT, DO NOT ENFORCE ANY VALUES
2516                            <1>       ;OR    AH,AH               ; CHECK FOR ANY WAIT?
2517                            <1>       ;JNZ   short DO_WAT        ; IF THERE DO NOT ENFORCE
2518 00004FAC 741E              <1>       jz     short HW_DONE
2519 00004FAE B40F              <1>       MOV    AH,HD12_SETTLE            ; LOAD 1.2M HEAD SETTLE MINIMUM
2520 00004FB0 8A87[558A0100]    <1>       MOV    AL,[DSK_STATE+eDI]  ; LOAD STATE
2521 00004FB6 24C0              <1>       AND    AL,RATE_MSK         ; KEEP ONLY RATE
2522 00004FB8 3C80              <1>       CMP    AL,RATE_250         ; 1.2 M DRIVE ?
2523 00004FBA 7502              <1>       JNZ    short DO_WAT        ; DEFAULT HEAD SETTLE LOADED
2524                            <1> ;GP3:
2525 00004FBC B414              <1>       MOV    AH,HD320_SETTLE           ; USE 320/360 HEAD SETTLE
2526                            <1> ;     JMP    SHORT DO_WAT
2527                            <1>
2528                            <1> ;ISNT_WRITE:
2529                            <1> ;     OR     AH,AH               ; CHECK FOR NO WAIT
2530                            <1> ;     JZ     short HW_DONE       ; IF NOT WRITE AND 0 ITS OK
2531                            <1>
2532                            <1> ;-----      AH CONTAINS NUMBER OF MILLISECONDS TO WAIT
2533                            <1> DO_WAT:
2534                            <1> ;     MOV    AL,AH               ; AL = # MILLISECONDS
2535                            <1> ;     ;XOR   AH,AH               ; AX = # MILLISECONDS
2536                            <1> J29:                              ;      1 MILLISECOND LOOP
2537                            <1>       ;mov   cx, WAIT_FDU_HEAD_SETTLE ; 33 ; 1 ms in 30 micro units.
2538 00004FBE B942000000        <1>       MOV    eCX,66              ; COUNT AT 15.085737 US PER COUNT
2539 00004FC3 E87ED4FFFF        <1>       CALL   WAITF               ; DELAY FOR 1 MILLISECOND
2540                            <1>       ;DEC   AL                  ; DECREMENT THE COUNT
2541 00004FC8 FECC              <1>       dec    ah
2542 00004FCA 75F2              <1>       JNZ    short J29           ; DO AL MILLISECOND # OF TIMES
2543                            <1> HW_DONE:
2544 00004FCC C3                <1>       RETn
2545                            <1>
2546                            <1> ;--------------------------------------------------------------------------
2547                            <1> ; NEC_OUTPUT
2548                            <1> ;     THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER TESTING
2549                            <1> ;     FOR CORRECT DIRECTION AND CONTROLLER READY THIS ROUTINE WILL
2550                            <1> ;     TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN A REASONABLE AMOUNT
2551                            <1> ;     OF TIME, SETTING THE DISKETTE STATUS ON COMPLETION.
2552                            <1> ;
2553                            <1> ; ON ENTRY:  AH = BYTE TO BE OUTPUT
2554                            <1> ;
2555                            <1> ; ON EXIT:   CY = 0  SUCCESS
2556                            <1> ;            CY = 1  FAILURE -- DISKETTE STATUS UPDATED
2557                            <1> ;                    IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL
2558                            <1> ;                    HIGHER THAN THE CALLER OF NEC OUTPUT. THIS REMOVES THE
2559                            <1> ;                    REQUIREMENT OF TESTING AFTER EVERY CALL OF NEC_OUTPUT.
2560                            <1> ;            AX,CX,DX DESTROYED
2561                            <1> ;--------------------------------------------------------------------------
2562                            <1>
2563                            <1> ; 09/12/2014 [Erdogan Tan]
2564                            <1> ;     (from 'PS2 Hardware Interface Tech. Ref. May 88', Page 09-05.)
2565                            <1> ; Diskette Drive Controller Status Register (3F4h)
2566                            <1> ;     This read only register facilitates the transfer of data between
2567                            <1> ;     the system microprocessor and the controller.
2568                            <1> ; Bit 7 - When set to 1, the Data register is ready to transfer data
2569                            <1> ;       with the system micrprocessor.
2570                            <1> ; Bit 6 - The direction of data transfer. If this bit is set to 0,
2571                            <1> ;       the transfer is to the controller.
2572                            <1> ; Bit 5 - When this bit is set to 1, the controller is in the non-DMA mode.
2573                            <1> ; Bit 4 - When this bit is set to 1, a Read or Write command is being executed.
2574                            <1> ; Bit 3 - Reserved.
2575                            <1> ; Bit 2 - Reserved.
2576                            <1> ; Bit 1 - When this bit is set to 1, dskette drive 1 is in the seek mode.
2577                            <1> ; Bit 0 - When this bit is set to 1, dskette drive 1 is in the seek mode.
2578                            <1>
2579                            <1> ; Data Register (3F5h)
2580                            <1> ; This read/write register passes data, commands and parameters, and provides
2581                            <1> ; diskette status information.
2582                            <1>
2583                            <1> NEC_OUTPUT:
2584                            <1>       ;PUSH  BX                  ; SAVE REG.
2585 00004FCD 66BAF403          <1>       MOV    DX,03F4H            ; STATUS PORT
2586                            <1>       ;MOV   BL,2                ; HIGH ORDER COUNTER
2587                            <1>       ;XOR   CX,CX               ; COUNT FOR TIME OUT
2588                            <1>       ; 16/12/2014
2589                            <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
00004FD1 B91B410000     <1>       mov    ecx, WAIT_FDU_SEND_LH   ; 16667 (27/02/2015)
                        <1> ;
                        <1> ;WFPS_OUTER_LP:
                        <1> ;     ;
                        <1> ;WFPS_CHECK_PORT:
                        <1> J23:
00004FD6 EC             <1>       IN     AL,DX               ; GET STATUS
00004FD7 24C0           <1>       AND    AL,11000000B        ; KEEP STATUS AND DIRECTION
00004FD9 3C80           <1>       CMP    AL,10000000B        ; STATUS 1 AND DIRECTION 0 ?
00004FDB 7418           <1>       JZ     short J27           ; STATUS AND DIRECTION OK
                        <1> WFPS_HI:
00004FDD E461           <1>       IN     AL, PORT_B   ;061h  ; SYS1 ; wait for hi to lo
00004FDF A810           <1>       TEST   AL,010H                    ; transition on memory
00004FE1 75FA           <1>       JNZ    SHORT WFPS_HI       ; refresh.
                        <1> WFPS_LO:
00004FE3 E461           <1>       IN     AL, PORT_B          ; SYS1
00004FE5 A810           <1>       TEST   AL,010H
00004FE7 74FA           <1>       JZ     SHORT WFPS_LO
                        <1>       ;LOOP  SHORT WFPS_CHECK_PORT
00004FE9 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>
00004FEB 800D[488A0100]80 <1>       OR     byte [DSKETTE_STATUS],TIME_OUT
                        <1>       ;POP   BX                  ; RESTORE REG.
00004FF2 58             <1>       POP    eAX ; 08/02/2015    ; DISCARD THE RETURN ADDRESS
00004FF3 F9             <1>       STC                        ; INDICATE ERROR TO CALLER
00004FF4 C3             <1>       RETn
                        <1>
                        <1> ;-----      DIRECTION AND STATUS OK; OUTPUT BYTE
                        <1>
                        <1> J27:
00004FF5 88E0           <1>       MOV    AL,AH               ; GET BYTE TO OUTPUT
00004FF7 6642           <1>       INC    DX                  ; DATA PORT = STATUS PORT + 1
00004FF9 EE             <1>       OUT    DX,AL               ; OUTPUT THE BYTE
                        <1>       ;;NEWIODELAY  ;; 27/02/2015
                        <1>       ; 27/02/2015
00004FFA 9C             <1>       PUSHF                      ; SAVE FLAGS
00004FFB B903000000     <1>       MOV    eCX, 3              ; 30 TO 45 MICROSECONDS WAIT FOR
00005000 E841D4FFFF     <1>       CALL   WAITF               ; NEC FLAGS UPDATE CYCLE
00005005 9D             <1>       POPF                       ; RESTORE FLAGS FOR EXIT
                        <1>       ;POP   BX                  ; RESTORE REG
00005006 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:
00005007 89FB           <1>       MOV    eBX,eDI                  ; BX = DRIVE #
00005009 B001           <1>       MOV    AL,1                ; ESTABLISH MASK FOR RECALIBRATE TEST
0000500B 86CB           <1>       XCHG   CL,BL               ; SET DRIVE VALULE INTO CL
0000500D D2C0           <1>       ROL    AL,CL               ; SHIFT MASK BY THE DRIVE VALUE
0000500F 86CB           <1>       XCHG   CL,BL               ; RECOVER TRACK VALUE
00005011 8405[458A0100] <1>       TEST   AL,[SEEK_STATUS]    ; TEST FOR RECALIBRATE REQUIRED
00005017 7526           <1>       JNZ    short J28A          ; JUMP IF RECALIBRATE NOT REQUIRED
                        <1>
00005019 0805[458A0100] <1>       OR     [SEEK_STATUS],AL    ; TURN ON THE NO RECALIBRATE BIT IN FLAG
```

```
2695 0000501F E862000000          <1>       CALL   RECAL               ; RECALIBRATE DRIVE
2696 00005024 730E                <1>       JNC    short AFT_RECAL           ; RECALIBRATE DONE
2697                              <1>
2698                              <1> ;-----      ISSUE RECALIBRATE FOR 80 TRACK DISKETTES
2699                              <1>
2700 00005026 C605[488A0100]00    <1>       MOV    byte [DSKETTE_STATUS],0    ; CLEAR OUT INVALID STATUS
2701 0000502D E854000000          <1>       CALL   RECAL               ; RECALIBRATE DRIVE
2702 00005032 7251                <1>       JC     short RB            ; IF RECALIBRATE FAILS TWICE THEN ERROR
2703                              <1>
2704                              <1> AFT_RECAL:
2705 00005034 C687[598A0100]00    <1>         MOV     byte [DSK_TRK+eDI],0    ; SAVE NEW CYLINDER AS PRESENT POSITION
2706 0000503B 08ED                <1>       OR     CH,CH               ; CHECK FOR SEEK TO TRACK 0
2707 0000503D 743F                <1>       JZ     short DO_WAIT       ; HEAD SETTLE, CY = 0 IF JUMP
2708                              <1>
2709                              <1> ;-----      DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK
2710                              <1>
2711 0000503F F687[558A0100]20    <1> J28A: TEST   byte [DSK_STATE+eDI],DBL_STEP ; CHECK FOR DOUBLE STEP REQUIRED
2712 00005046 7402                <1>       JZ     short _R7           ; SINGLE STEP REQUIRED BYPASS DOUBLE
2713 00005048 D0E5                <1>       SHL    CH,1                ; DOUBLE NUMBER OF STEP TO TAKE
2714                              <1>
2715 0000504A 3AAF[598A0100]      <1> _R7:  CMP    CH, [DSK_TRK+eDI]   ; SEE IF ALREADY AT THE DESIRED TRACK
2716 00005050 7433                <1>       JE     short RB            ; IF YES, DO NOT NEED TO SEEK
2717                              <1>
2718 00005052 BA[85500000]        <1>       MOV    eDX, NEC_ERR        ; LOAD RETURN ADDRESS
2719 00005057 52                  <1>       PUSH   eDX ; (*)           ; ON STACK FOR NEC OUTPUT ERROR
2720 00005058 88AF[598A0100]      <1>       MOV    [DSK_TRK+eDI],CH    ; SAVE NEW CYLINDER AS PRESENT POSITION
2721 0000505E B40F                <1>       MOV    AH,0FH              ; SEEK COMMAND TO NEC
2722 00005060 E868FFFFFF          <1>       CALL   NEC_OUTPUT
2723 00005065 89FB                <1>       MOV    eBX,eDI                  ; BX = DRIVE #
2724 00005067 88DC                <1>       MOV    AH,BL               ; OUTPUT DRIVE NUMBER
2725 00005069 E85FFFFFFF          <1>       CALL   NEC_OUTPUT
2726 0000506E 8AA7[598A0100]      <1>       MOV    AH, [DSK_TRK+eDI]   ; GET CYLINDER NUMBER
2727 00005074 E854FFFFFF          <1>       CALL   NEC_OUTPUT
2728 00005079 E829000000          <1>       CALL   CHK_STAT_2          ; ENDING INTERRUPT AND SENSE STATUS
2729                              <1>
2730                              <1> ;-----      WAIT FOR HEAD SETTLE
2731                              <1>
2732                              <1> DO_WAIT:
2733 0000507E 9C                  <1>       PUSHF                      ; SAVE STATUS
2734 0000507F E816FFFFFF          <1>       CALL   HD_WAIT                  ; WAIT FOR HEAD SETTLE TIME
2735 00005084 9D                  <1>       POPF                       ; RESTORE STATUS
2736                              <1> RB:
2737                              <1> NEC_ERR:
2738                              <1>       ; 08/02/2015 (code trick here from original IBM PC/AT DISKETTE.ASM)
2739                              <1>       ; (*) nec_err -> retn (push edx -> pop edx) -> nec_err -> retn
2740 00005085 C3                  <1>       RETn                       ; RETURN TO CALLER
2741                              <1>
2742                              <1> ;--------------------------------------------------------------------------------
2743                              <1> ; RECAL
2744                              <1> ;     RECALIBRATE DRIVE
2745                              <1> ;
2746                              <1> ; ON ENTRY:  DI = DRIVE #
2747                              <1> ;
2748                              <1> ; ON EXIT:   CY REFLECTS STATUS OF OPERATION.
2749                              <1> ;--------------------------------------------------------------------------------
2750                              <1> RECAL:
2751 00005086 6651                <1>       PUSH   CX
2752 00005088 B8[A4500000]        <1>       MOV    eAX, RC_BACK        ; LOAD NEC_OUTPUT ERROR
2753 0000508D 50                  <1>       PUSH   eAX
2754 0000508E B407                <1>       MOV    AH,07H              ; RECALIBRATE COMMAND
2755 00005090 E838FFFFFF          <1>       CALL   NEC_OUTPUT
2756 00005095 89FB                <1>       MOV    eBX,eDI                  ; BX = DRIVE #
2757 00005097 88DC                <1>       MOV    AH,BL
2758 00005099 E82FFFFFFF          <1>       CALL   NEC_OUTPUT          ; OUTPUT THE DRIVE NUMBER
2759 0000509E E804000000          <1>       CALL   CHK_STAT_2          ; GET THE INTERRUPT AND SENSE INT STATUS
2760 000050A3 58                  <1>       POP    eAX                 ; THROW AWAY ERROR
2761                              <1> RC_BACK:
2762 000050A4 6659                <1>       POP    CX
2763 000050A6 C3                  <1>       RETn
2764                              <1>
2765                              <1> ;--------------------------------------------------------------------------------
2766                              <1> ; CHK_STAT_2
2767                              <1> ;     THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER RECALIBRATE,
2768                              <1> ;     OR SEEK TO THE ADAPTER. THE INTERRUPT IS WAITED FOR, THE
2769                              <1> ;     INTERRUPT STATUS SENSED, AND THE RESULT RETURNED TO THE CALLER.
2770                              <1> ;
2771                              <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
2772                              <1> ;--------------------------------------------------------------------------------
2773                              <1> CHK_STAT_2:
2774 000050A7 B8[CF500000]        <1>         MOV     eAX, CS_BACK            ; LOAD NEC_OUTPUT ERROR ADDRESS
2775 000050AC 50                  <1>       PUSH   eAX
2776 000050AD E828000000          <1>       CALL   WAIT_INT            ; WAIT FOR THE INTERRUPT
2777 000050B2 721A                <1>       JC     short J34           ; IF ERROR, RETURN IT
2778 000050B4 B408                <1>       MOV    AH,08H              ; SENSE INTERRUPT STATUS COMMAND
2779 000050B6 E812FFFFFF          <1>       CALL   NEC_OUTPUT
2780 000050BB E84A000000          <1>       CALL   RESULTS                  ; READ IN THE RESULTS
2781 000050C0 720C                <1>       JC     short J34
2782 000050C2 A0[498A0100]        <1>       MOV    AL,[NEC_STATUS]          ; GET THE FIRST STATUS BYTE
2783 000050C7 2460                <1>       AND    AL,01100000B        ; ISOLATE THE BITS
2784 000050C9 3C60                <1>       CMP    AL,01100000B        ; TEST FOR CORRECT VALUE
2785 000050CB 7403                <1>       JZ     short J35           ; IF ERROR, GO MARK IT
2786 000050CD F8                  <1>       CLC                        ; GOOD RETURN
2787                              <1> J34:
2788 000050CE 58                  <1>       POP    eAX                 ; THROW AWAY ERROR RETURN
2789                              <1> CS_BACK:
2790 000050CF C3                  <1>       RETn
2791                              <1> J35:
2792 000050D0 800D[488A0100]40    <1>       OR     byte [DSKETTE_STATUS], BAD_SEEK
2793 000050D7 F9                  <1>       STC                        ; ERROR RETURN CODE
2794 000050D8 EBF4                <1>       JMP    SHORT J34
2795                              <1>
2796                              <1> ;--------------------------------------------------------------------------------
2797                              <1> ; WAIT_INT
2798                              <1> ;     THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR A TIME OUT ROUTINE
2799                              <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:
000050DA FB             <1>       STI                     ; TURN ON INTERRUPTS, JUST IN CASE
000050DB 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
000050DC B986450100     <1>       mov   ecx, WAIT_FDU_INT_LH      ; 83334 (2.5 seconds)
                        <1> WFMS_CHECK_MEM:
000050E1 F605[458A0100]80 <1>       test  byte [SEEK_STATUS],INT_FLAG ; TEST FOR INTERRUPT OCCURRING
000050E8 7516           <1>         jnz     short J37
                        <1> WFMS_HI:
000050EA E461           <1>       IN    AL,PORT_B  ; 061h   ; SYS1, wait for lo to hi
000050EC A810           <1>       TEST  AL,010H                   ; transition on memory
000050EE 75FA           <1>       JNZ   SHORT WFMS_HI       ; refresh.
                        <1> WFMS_LO:
000050F0 E461           <1>       IN    AL,PORT_B           ;SYS1
000050F2 A810           <1>       TEST  AL,010H
000050F4 74FA           <1>       JZ    SHORT WFMS_LO
000050F6 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:
000050F8 800D[488A0100]80 <1>       OR    byte [DSKETTE_STATUS], TIME_OUT ; NOTHING HAPPENED
000050FF F9             <1>       STC                     ; ERROR RETURN
                        <1> J37:
00005100 9C             <1>       PUSHF                   ; SAVE CURRENT CARRY
00005101 8025[458A0100]7F <1>       AND   byte [SEEK_STATUS], ~INT_FLAG ; TURN OFF INTERRUPT FLAG
00005108 9D             <1>       POPF                    ; RECOVER CARRY
00005109 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:
0000510A 57             <1>       PUSH  eDI
0000510B BF[498A0100]   <1>       MOV   eDI, NEC_STATUS           ; POINTER TO DATA AREA
00005110 B307           <1>       MOV   BL,7              ; MAX STATUS BYTES
00005112 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
00005116 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
0000511B EC             <1>       IN    AL,DX               ; GET STATUS
0000511C 24C0           <1>       AND   AL,11000000B        ; KEEP ONLY STATUS AND DIRECTION
0000511E 3CC0           <1>       CMP   AL,11000000B        ; STATUS 1 AND DIRECTION 1 ?
00005120 7418           <1>       JZ    short J42           ; STATUS AND DIRECTION OK
                        <1> WFPSR_HI:
00005122 E461           <1>       IN    AL, PORT_B   ;061h  ; SYS1 ; wait for hi to lo
00005124 A810           <1>       TEST  AL,010H                   ; transition on memory
00005126 75FA           <1>       JNZ   SHORT WFPSR_HI            ; refresh.
                        <1> WFPSR_LO:
00005128 E461           <1>       IN    AL, PORT_B          ; SYS1
0000512A A810           <1>       TEST  AL,010H
0000512C 74FA           <1>       JZ    SHORT WFPSR_LO
0000512E 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:
00005130 800D[488A0100]80 <1>     OR    byte [DSKETTE_STATUS],TIME_OUT
00005137 F9             <1>       STC                       ; SET ERROR RETURN
00005138 EB29           <1>       JMP   SHORT POPRES        ; POP REGISTERS AND RETURN
                        <1>
                        <1> ;-----      READ IN THE STATUS
                        <1>
                        <1> J42:
0000513A EB00           <1>       JMP   $+2                 ; I/O DELAY
0000513C 6642           <1>       INC   DX                  ; POINT AT DATA PORT
0000513E EC             <1>       IN    AL,DX               ; GET THE DATA
                        <1>       ; 16/12/2014
                        <1>       NEWIODELAY
0000513F E6EB           <2>  out 0ebh,al
00005141 8807           <1>         MOV     [eDI],AL              ; STORE THE BYTE
00005143 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>
00005144 B903000000     <1>       MOV   eCX,3               ; MINIMUM 24 MICROSECONDS FOR NEC
00005149 E8F8D2FFFF     <1>       CALL  WAITF               ; WAIT 30 TO 45 MICROSECONDS
0000514E 664A           <1>       DEC   DX                  ; POINT AT STATUS PORT
00005150 EC             <1>       IN    AL,DX               ; GET STATUS
                        <1>       ; 16/12/2014
                        <1>       NEWIODELAY
00005151 E6EB           <2>  out 0ebh,al
                        <1>       ;
00005153 A810           <1>       TEST  AL,00010000B        ; TEST FOR NEC STILL BUSY
00005155 740C           <1>       JZ    short POPRES        ; RESULTS DONE ?
                        <1>
00005157 FECB           <1>       DEC   BL                  ; DECREMENT THE STATUS COUNTER
00005159 75BB           <1>         JNZ     short  R10              ; GO BACK FOR MORE
0000515B 800D[488A0100]20 <1>     OR    byte [DSKETTE_STATUS],BAD_NEC ; TOO MANY STATUS BYTES
00005162 F9             <1>       STC                       ; SET ERROR FLAG
                        <1>
                        <1> ;-----      RESULT OPERATION IS DONE
                        <1> POPRES:
00005163 5F             <1>       POP   eDI
00005164 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:
00005165 E8A2FDFFFF     <1>       CALL  MOTOR_ON            ; TURN ON THE MOTOR IF OFF
0000516A 66BAF703       <1>       MOV   DX,03F7H            ; ADDRESS DIGITAL INPUT REGISTER
0000516E EC             <1>       IN    AL,DX               ; INPUT DIGITAL INPUT REGISTER
0000516F A880           <1>       TEST  AL,DSK_CHG          ; CHECK FOR DISK CHANGE LINE ACTIVE
00005171 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:
```

```
3008 00005172 E895FDFFFF          <1>       CALL   MOTOR_ON           ; TURN ON MOTOR IF NOT ALREADY ON
3009 00005177 E80AFFFFFF          <1>       CALL   RECAL              ; RECALIBRATE DRIVE
3010 0000517C 7251                <1>       JC     short DD_BAC       ; ASSUME NO DRIVE PRESENT
3011 0000517E B530                <1>       MOV    CH,TRK_SLAP        ; SEEK TO TRACK 48
3012 00005180 E882FEFFFF          <1>       CALL   SEEK
3013 00005185 7248                <1>       JC     short DD_BAC       ; ERROR NO DRIVE
3014 00005187 B50B                <1>       MOV    CH,QUIET_SEEK+1          ; SEEK TO TRACK 10
3015                              <1> SK_GIN:
3016 00005189 FECD                <1>       DEC    CH                 ; DECREMENT TO NEXT TRACK
3017 0000518B 6651                <1>       PUSH   CX                 ; SAVE TRACK
3018 0000518D E875FEFFFF          <1>       CALL   SEEK
3019 00005192 723C                <1>       JC     short POP_BAC      ; POP AND RETURN
3020 00005194 B8[D0510000]        <1>       MOV    eAX, POP_BAC       ; LOAD NEC OUTPUT ERROR ADDRESS
3021 00005199 50                  <1>       PUSH   eAX
3022 0000519A B404                <1>       MOV    AH,SENSE_DRV_ST          ; SENSE DRIVE STATUS COMMAND BYTE
3023 0000519C E82CFEFFFF          <1>       CALL   NEC_OUTPUT         ; OUTPUT TO NEC
3024 000051A1 6689F8              <1>       MOV    AX,DI              ; AL = DRIVE
3025 000051A4 88C4                <1>       MOV    AH,AL              ; AH = DRIVE
3026 000051A6 E822FEFFFF          <1>       CALL   NEC_OUTPUT         ; OUTPUT TO NEC
3027 000051AB E85AFFFFFF          <1>       CALL   RESULTS                 ; GO GET STATUS
3028 000051B0 58                  <1>       POP    eAX                ; THROW AWAY ERROR ADDRESS
3029 000051B1 6659                <1>       POP    CX                 ; RESTORE TRACK
3030 000051B3 F605[498A0100]10    <1>       TEST   byte [NEC_STATUS], HOME    ; TRACK 0 ?
3031 000051BA 74CD                <1>       JZ     short SK_GIN       ; GO TILL TRACK 0
3032 000051BC 08ED                <1>       OR     CH,CH              ; IS HOME AT TRACK 0
3033 000051BE 7408                <1>       JZ     short IS_80        ; MUST BE 80 TRACK DRIVE
3034                              <1>
3035                              <1> ;     DRIVE IS A 360; SET DRIVE TO DETERMINED;
3036                              <1> ;     SET MEDIA TO DETERMINED AT RATE 250.
3037                              <1>
3038 000051C0 808F[558A0100]94    <1>       OR     byte [DSK_STATE+eDI], DRV_DET+MED_DET+RATE_250
3039 000051C7 C3                  <1>       RETn                      ; ALL INFORMATION SET
3040                              <1> IS_80:
3041 000051C8 808F[558A0100]01    <1>       OR     byte [DSK_STATE+eDI], TRK_CAPA ; SETUP 80 TRACK CAPABILITY
3042                              <1> DD_BAC:
3043 000051CF C3                  <1>       RETn
3044                              <1> POP_BAC:
3045 000051D0 6659                <1>       POP    CX                 ; THROW AWAY
3046 000051D2 C3                  <1>       RETn
3047                              <1>
3048                              <1> fdc_int:
3049                              <1>         ; 30/07/2015
3050                              <1>         ; 16/02/2015
3051                              <1> ;int_0Eh: ; 11/12/2014
3052                              <1>
3053                              <1> ;--- HARDWARE INT 0EH -- ( IRQ LEVEL  6 ) -------------------------------------
3054                              <1> ; DISK_INT
3055                              <1> ;     THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.
3056                              <1> ;
3057                              <1> ; ON EXIT:  THE INTERRUPT FLAG IS SET IN @SEEK_STATUS.
3058                              <1> ;--------------------------------------------------------------------------------
3059                              <1> DISK_INT_1:
3060                              <1>
3061 000051D3 6650                <1>       PUSH   AX                 ; SAVE WORK REGISTER
3062 000051D5 1E                  <1>       push   ds
3063 000051D6 66B81000            <1>       mov    ax, KDATA
3064 000051DA 8ED8                <1>       mov    ds, ax
3065 000051DC 800D[458A0100]80    <1>         OR     byte [SEEK_STATUS], INT_FLAG ; TURN ON INTERRUPT OCCURRED
3066 000051E3 B020                <1>       MOV     AL,EOI                  ; END OF INTERRUPT MARKER
3067 000051E5 E620                <1>       OUT    INTA00,AL          ; INTERRUPT CONTROL PORT
3068 000051E7 1F                  <1>       pop    ds
3069 000051E8 6658                <1>       POP    AX                 ; RECOVER REGISTER
3070 000051EA CF                  <1>       IRETd                     ; RETURN FROM INTERRUPT
3071                              <1>
3072                              <1> ;--------------------------------------------------------------------------------
3073                              <1> ; DSKETTE_SETUP
3074                              <1> ;     THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE OF
3075                              <1> ;     DISKETTE DRIVES ARE ATTACH TO THE SYSTEM.
3076                              <1> ;--------------------------------------------------------------------------------
3077                              <1>
3078                              <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
3079                              <1>
3080                              <1> DSKETTE_SETUP:
3081                              <1>       ;PUSH  AX                 ; SAVE REGISTERS
3082                              <1>       ;PUSH  BX
3083                              <1>       ;PUSH  CX
3084 000051EB 52                  <1>       PUSH   eDX
3085                              <1>       ;PUSH  DI
3086                              <1>       ;;PUSH DS
3087                              <1>       ; 14/12/2014
3088                              <1>       ;mov   word [DISK_POINTER], MD_TBL6
3089                              <1>       ;mov   [DISK_POINTER+2], cs
3090                              <1>       ;
3091                              <1>       ;OR    byte [RTC_WAIT_FLAG], 1   ; NO RTC WAIT, FORCE USE OF LOOP
3092 000051EC 31FF                <1>       XOR    eDI,eDI                   ; INITIALIZE DRIVE POINTER
3093 000051EE 66C705[558A0100]00- <1>       MOV    WORD [DSK_STATE],0  ; INITIALIZE STATES
3093 000051F6 00                  <1>
3094 000051F7 8025[508A0100]33    <1>       AND    byte [LASTRATE],~(STRT_MSK+SEND_MSK) ; CLEAR START & SEND
3095 000051FE 800D[508A0100]C0    <1>       OR     byte [LASTRATE],SEND_MSK ; INITIALIZE SENT TO IMPOSSIBLE
3096 00005205 C605[458A0100]00    <1>       MOV    byte [SEEK_STATUS],0      ; INDICATE RECALIBRATE NEEDED
3097 0000520C C605[478A0100]00    <1>       MOV    byte [MOTOR_COUNT],0      ; INITIALIZE MOTOR COUNT
3098 00005213 C605[468A0100]00    <1>       MOV    byte [MOTOR_STATUS],0     ; INITIALIZE DRIVES TO OFF STATE
3099 0000521A C605[488A0100]00    <1>       MOV    byte [DSKETTE_STATUS],0   ; NO ERRORS
3100                              <1>       ;
3101                              <1>       ; 28/02/2015
3102                              <1>       ;mov   word [cfd], 100h
3103 00005221 E82BF2FFFF          <1>       call   DSK_RESET
3104 00005226 5A                  <1>       pop    edx
3105 00005227 F8                  <1>       clc    ; 29/05/2016
3106 00005228 C3                  <1>       retn
3107                              <1>
3108                              <1> ;SUP0:
3109                              <1> ;     CALL   DRIVE_DET          ; DETERMINE DRIVE
3110                              <1> ;     CALL   XLAT_OLD           ; TRANSLATE STATE TO COMPATIBLE MODE
3111                              <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
 00005229 F8            <1>       clc ; 20/07/2020
 0000522A 9C            <1>       pushfd
 0000522B 0E            <1>       push   cs
 0000522C E848010000    <1>       call   DISK_IO
 00005231 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> ; 30/08/2020 - TRDOS 386 v2
                        <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                                                  :
```

```
3217                                    <1> ;     (AH)= 08H  RETURN THE CURRENT DRIVE PARAMETERS                        :
3218                                    <1> ;     (AH)= 09H  INITIALIZE DRIVE PAIR CHARACTERISTICS                  :
3219                                    <1> ;                INTERRUPT 41 POINTS TO DATA BLOCK FOR DRIVE 0          :
3220                                    <1> ;                INTERRUPT 46 POINTS TO DATA BLOCK FOR DRIVE 1          :
3221                                    <1> ;     (AH)= 0AH  READ LONG                                              :
3222                                    <1> ;     (AH)= 0BH  WRITE LONG  (READ & WRITE LONG ENCOMPASS 512 + 4 BYTES ECC) :
3223                                    <1> ;     (AH)= 0CH  SEEK                                                   :
3224                                    <1> ;     (AH)= 0DH  ALTERNATE DISK RESET (SEE DL)                          :
3225                                    <1> ;     (AH)= 0EH  UNUSED                                                 :
3226                                    <1> ;     (AH)= 0FH  UNUSED                                                 :
3227                                    <1> ;     (AH)= 10H  TEST DRIVE READY                                       :
3228                                    <1> ;     (AH)= 11H  RECALIBRATE                                            :
3229                                    <1> ;     (AH)= 12H  UNUSED                                                 :
3230                                    <1> ;     (AH)= 13H  UNUSED                                                 :
3231                                    <1> ;     (AH)= 14H  CONTROLLER INTERNAL DIAGNOSTIC                         :
3232                                    <1> ;     (AH)= 15H  READ DASD TYPE                                         :
3233                                    <1> ;                                                                       :
3234                                    <1> ;-------------------------------------------------------------------------
3235                                    <1> ;                                                                       :
3236                                    <1> ;     REGISTERS USED FOR FIXED DISK OPERATIONS                          :
3237                                    <1> ;                                                                       :
3238                                    <1> ;           (DL)   -  DRIVE NUMBER     (80H-81H FOR DISK. VALUE CHECKED)  :
3239                                    <1> ;           (DH)   -  HEAD NUMBER          (0-15 ALLOWED, NOT VALUE CHECKED)  :
3240                                    <1> ;           (CH)   -  CYLINDER NUMBER  (0-1023, NOT VALUE CHECKED)(SEE CL):
3241                                    <1> ;           (CL)   -  SECTOR NUMBER    (1-17, NOT VALUE CHECKED)        :
3242                                    <1> ;                                                                       :
3243                                    <1> ;                     NOTE: HIGH 2 BITS OF CYLINDER NUMBER ARE PLACED     :
3244                                    <1> ;                          IN THE HIGH 2 BITS OF THE CL REGISTER        :
3245                                    <1> ;                          (10 BITS TOTAL)                              :
3246                                    <1> ;                                                                       :
3247                                    <1> ;           (AL)   -  NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H,    :
3248                                    <1> ;                                       FOR READ/WRITE LONG 1-79H)      :
3249                                    <1> ;                                                                       :
3250                                    <1> ;           (ES:BX) -  ADDRESS OF BUFFER FOR READS AND WRITES,          :
3251                                    <1> ;                      (NOT REQUIRED FOR VERIFY)                        :
3252                                    <1> ;                                                                       :
3253                                    <1> ;           FORMAT (AH=5) ES:BX POINTS TO A 512 BYTE BUFFER. THE FIRST     :
3254                                    <1> ;                      2*(SECTORS/TRACK) BYTES CONTAIN F,N FOR EACH SECTOR.:
3255                                    <1> ;                      F = 00H FOR A GOOD SECTOR                        :
3256                                    <1> ;                          80H FOR A BAD SECTOR                         :
3257                                    <1> ;                      N = SECTOR NUMBER                                :
3258                                    <1> ;                      FOR AN INTERLEAVE OF 2 AND 17 SECTORS/TRACK             :
3259                                    <1> ;                      THE TABLE SHOULD BE:                             :
3260                                    <1> ;                                                                       :
3261                                    <1> ;              DB     00H,01H,00H,0AH,00H,02H,00H,0BH,00H,03H,00H,0CH     :
3262                                    <1> ;              DB     00H,04H,00H,0DH,00H,05H,00H,0EH,00H,06H,00H,0FH     :
3263                                    <1> ;              DB     00H,07H,00H,10H,00H,08H,00H,11H,00H,09H          :
3264                                    <1> ;                                                                       :
3265                                    <1> ;-------------------------------------------------------------------------
3266                                    <1>
3267                                    <1> ;-------------------------------------------------------------------------
3268                                    <1> ; OUTPUT                                                                :
3269                                    <1> ;     AH = STATUS OF CURRENT OPERATION                                  :
3270                                    <1> ;          STATUS BITS ARE DEFINED IN THE EQUATES BELOW                 :
3271                                    <1> ;     CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN)                      :
3272                                    <1> ;     CY = 1 FAILED OPERATION (AH HAS ERROR REASON)                     :
3273                                    <1> ;                                                                       :
3274                                    <1> ;     NOTE:  ERROR 11H  INDICATES THAT THE DATA READ HAD A RECOVERABLE      :
3275                                    <1> ;            ERROR WHICH WAS CORRECTED BY THE ECC ALGORITHM.  THE DATA      :
3276                                    <1> ;            IS PROBABLY GOOD,   HOWEVER THE BIOS ROUTINE INDICATES AN      :
3277                                    <1> ;            ERROR TO ALLOW THE CONTROLLING PROGRAM A CHANCE TO DECIDE      :
3278                                    <1> ;            FOR ITSELF.  THE  ERROR  MAY  NOT  RECUR  IF  THE DATA IS      :
3279                                    <1> ;            REWRITTEN.                                                     :
3280                                    <1> ;                                                                           :
3281                                    <1> ;     IF DRIVE PARAMETERS WERE REQUESTED (DL >= 80H),                   :
3282                                    <1> ;        INPUT:                                                         :
3283                                    <1> ;          (DL) = DRIVE NUMBER                                          :
3284                                    <1> ;          ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)                                :

3285                                    <1> ;          EBX = Buffer address for fixed disk parameters table (32 bytes)   :
3286                                    <1> ;        OUTPUT:                                                        :
3287                                    <1> ;          (DL) = NUMBER OF CONSECUTIVE ACKNOWLEDGING DRIVES ATTACHED (1-2)  :
3288                                    <1> ;                 (CONTROLLER CARD ZERO TALLY ONLY)                       :
3289                                    <1> ;          (DH) = MAXIMUM USEABLE VALUE FOR HEAD NUMBER                   :
3290                                    <1> ;          (CH) = MAXIMUM USEABLE VALUE FOR CYLINDER NUMBER                  :
3291                                    <1> ;          (CL) = MAXIMUM USEABLE VALUE FOR SECTOR NUMBER                 :
3292                                    <1> ;                 AND CYLINDER NUMBER HIGH BITS                           :
3293                                    <1> ;                                                                       :
3294                                    <1> ;     IF READ DASD TYPE WAS REQUESTED,                                  :
3295                                    <1> ;                                                                       :
3296                                    <1> ;     AH = 0 - NOT PRESENT                                              :
3297                                    <1> ;          1 - DISKETTE - NO CHANGE LINE AVAILABLE                      :
3298                                    <1> ;          2 - DISKETTE - CHANGE LINE AVAILABLE                         :
3299                                    <1> ;          3 - FIXED DISK                                               :
3300                                    <1> ;                                                                       :
3301                                    <1> ;     CX,DX = NUMBER OF 512 BYTE BLOCKS WHEN AH = 3                     :
3302                                    <1> ;                                                                       :
3303                                    <1> ;     REGISTERS WILL BE PRESERVED EXCEPT WHEN THEY ARE USED TO RETURN       :
3304                                    <1> ;     INFORMATION.                                                      :
3305                                    <1> ;                                                                       :
3306                                    <1> ;     NOTE: IF AN ERROR IS REPORTED BY THE DISK CODE, THE APPROPRIATE       :
3307                                    <1> ;           ACTION IS TO RESET THE DISK, THEN RETRY THE OPERATION.          :
3308                                    <1> ;                                                                       :
3309                                    <1> ;-------------------------------------------------------------------------
3310                                    <1>
3311                                    <1> SENSE_FAIL   EQU    0FFH          ; NOT IMPLEMENTED
3312                                    <1> NO_ERR       EQU    0E0H          ; STATUS ERROR/ERROR REGISTER=0
3313                                    <1> WRITE_FAULT  EQU    0CCH          ; WRITE FAULT ON SELECTED DRIVE
3314                                    <1> UNDEF_ERR    EQU    0BBH          ; UNDEFINED ERROR OCCURRED
3315                                    <1> NOT_RDY      EQU    0AAH          ; DRIVE NOT READY
3316                                    <1> TIME_OUT     EQU    80H           ; ATTACHMENT FAILED TO RESPOND
3317                                    <1> BAD_SEEK     EQU    40H           ; SEEK OPERATION FAILED
3318                                    <1> BAD_CNTLR    EQU    20H           ; CONTROLLER HAS FAILED
3319                                    <1> DATA_CORRECTED     EQU    11H          ; ECC CORRECTED DATA ERROR
3320                                    <1> BAD_ECC      EQU    10H           ; BAD ECC ON DISK READ
```

```
3321                    <1> BAD_TRACK    EQU    0BH          ; NOT IMPLEMENTED
3322                    <1> BAD_SECTOR   EQU    0AH          ; BAD SECTOR FLAG DETECTED
3323                    <1> ;DMA_BOUNDARY      EQU    09H          ; DATA EXTENDS TOO FAR
3324                    <1> INIT_FAIL    EQU    07H          ; DRIVE PARAMETER ACTIVITY FAILED
3325                    <1> BAD_RESET    EQU    05H          ; RESET FAILED
3326                    <1> ;RECORD_NOT_FND    EQU    04H          ; REQUESTED SECTOR NOT FOUND
3327                    <1> ;BAD_ADDR_MARK     EQU    02H          ; ADDRESS MARK NOT FOUND
3328                    <1> ;BAD_CMD     EQU    01H          ; BAD COMMAND PASSED TO DISK I/O
3329                    <1>
3330                    <1> ;--------------------------------------------------------
3331                    <1> ;                                                      :
3332                    <1> ; FIXED DISK PARAMETER TABLE                           :
3333                    <1> ;  -  THE TABLE IS COMPOSED OF A BLOCK DEFINED AS:     :
3334                    <1> ;                                                      :
3335                    <1> ;  +0 (1 WORD) - MAXIMUM NUMBER OF CYLINDERS           :
3336                    <1> ;  +2 (1 BYTE) - MAXIMUM NUMBER OF HEADS               :
3337                    <1> ;  +3 (1 WORD) - NOT USED/SEE PC-XT                    :
3338                    <1> ;  +5 (1 WORD) - STARTING WRITE PRECOMPENSATION CYL    :
3339                    <1> ;  +7 (1 BYTE) - MAXIMUM ECC DATA BURST LENGTH         :
3340                    <1> ;  +8 (1 BYTE) - CONTROL BYTE                          :
3341                    <1> ;                BIT   7 DISABLE RETRIES -OR-          :
3342                    <1> ;                BIT   6 DISABLE RETRIES               :
3343                    <1> ;                BIT   3 MORE THAN 8 HEADS             :
3344                    <1> ;  +9 (3 BYTES)- NOT USED/SEE PC-XT                    :
3345                    <1> ; +12 (1 WORD) - LANDING ZONE                          :
3346                    <1> ; +14 (1 BYTE) - NUMBER OF SECTORS/TRACK               :
3347                    <1> ; +15 (1 BYTE) - RESERVED FOR FUTURE USE               :
3348                    <1> ;                                                      :
3349                    <1> ;      - TO DYNAMICALLY DEFINE A SET OF PARAMETERS     :
3350                    <1> ;        BUILD A TABLE FOR UP TO 15 TYPES AND PLACE    :
3351                    <1> ;        THE CORRESPONDING VECTOR INTO INTERRUPT 41     :
3352                    <1> ;        FOR DRIVE 0 AND INTERRUPT 46 FOR DRIVE 1.     :
3353                    <1> ;                                                      :
3354                    <1> ;--------------------------------------------------------
3355                    <1>
3356                    <1> ;--------------------------------------------------------
3357                    <1> ;                                                      :
3358                    <1> ; HARDWARE SPECIFIC VALUES                             :
3359                    <1> ;                                                      :
3360                    <1> ;  -  CONTROLLER I/O PORT                              :
3361                    <1> ;                                                      :
3362                    <1> ;     > WHEN READ FROM:                                :
3363                    <1> ;     HF_PORT+0 - READ DATA (FROM CONTROLLER TO CPU)       :
3364                    <1> ;     HF_PORT+1 - GET ERROR REGISTER                   :
3365                    <1> ;     HF_PORT+2 - GET SECTOR COUNT                     :
3366                    <1> ;     HF_PORT+3 - GET SECTOR NUMBER                    :
3367                    <1> ;     HF_PORT+4 - GET CYLINDER LOW                     :
3368                    <1> ;     HF_PORT+5 - GET CYLINDER HIGH (2 BITS)           :
3369                    <1> ;     HF_PORT+6 - GET SIZE/DRIVE/HEAD                  :
3370                    <1> ;     HF_PORT+7 - GET STATUS REGISTER                  :
3371                    <1> ;                                                      :
3372                    <1> ;     > WHEN WRITTEN TO:                               :
3373                    <1> ;     HF_PORT+0 - WRITE DATA (FROM CPU TO CONTROLLER)  :
3374                    <1> ;     HF_PORT+1 - SET PRECOMPENSATION CYLINDER         :
3375                    <1> ;     HF_PORT+2 - SET SECTOR COUNT                     :
3376                    <1> ;     HF_PORT+3 - SET SECTOR NUMBER                    :
3377                    <1> ;     HF_PORT+4 - SET CYLINDER LOW                     :
3378                    <1> ;     HF_PORT+5 - SET CYLINDER HIGH (2 BITS)           :
3379                    <1> ;     HF_PORT+6 - SET SIZE/DRIVE/HEAD                  :
3380                    <1> ;     HF_PORT+7 - SET COMMAND REGISTER                 :
3381                    <1> ;                                                      :
3382                    <1> ;--------------------------------------------------------
3383                    <1>
3384                    <1> ;HF_PORT     EQU    01F0H  ; DISK PORT
3385                    <1> ;HF1_PORT    equ    0170h
3386                    <1> ;HF_REG_PORT EQU    03F6H
3387                    <1> ;HF1_REG_PORT      equ    0376h
3388                    <1>
3389                    <1> HDC1_BASEPORT      equ    1F0h
3390                    <1> HDC2_BASEPORT      equ    170h
3391                    <1>
3392                    <1> align 2
3393                    <1>
3394                    <1> ;-----             STATUS REGISTER
3395                    <1>
3396                    <1> ST_ERROR     EQU    00000001B    ;
3397                    <1> ST_INDEX     EQU    00000010B    ;
3398                    <1> ST_CORRCTD   EQU    00000100B    ; ECC CORRECTION SUCCESSFUL
3399                    <1> ST_DRQ             EQU    00001000B    ;
3400                    <1> ST_SEEK_COMPL      EQU    00010000B    ; SEEK COMPLETE
3401                    <1> ST_WRT_FLT   EQU    00100000B    ; WRITE FAULT
3402                    <1> ST_READY     EQU    01000000B    ;
3403                    <1> ST_BUSY      EQU    10000000B    ;
3404                    <1>
3405                    <1> ;-----             ERROR REGISTER
3406                    <1>
3407                    <1> ERR_DAM      EQU    00000001B    ; DATA ADDRESS MARK NOT FOUND
3408                    <1> ERR_TRK_0    EQU    00000010B    ; TRACK 0 NOT FOUND ON RECAL
3409                    <1> ERR_ABORT    EQU    00000100B    ; ABORTED COMMAND
3410                    <1> ;            EQU    00001000B    ; NOT USED
3411                    <1> ERR_ID             EQU    00010000B    ; ID NOT FOUND
3412                    <1> ;            EQU    00100000B    ; NOT USED
3413                    <1> ERR_DATA_ECC EQU    01000000B
3414                    <1> ERR_BAD_BLOCK      EQU    10000000B
3415                    <1>
3416                    <1>
3417                    <1> RECAL_CMD    EQU    00010000B    ; DRIVE RECAL(10H)
3418                    <1> READ_CMD     EQU    00100000B    ;      READ   (20H)
3419                    <1> WRITE_CMD    EQU    00110000B    ;      WRITE  (30H)
3420                    <1> VERIFY_CMD   EQU    01000000B    ;      VERIFY (40H)
3421                    <1> FMTTRK_CMD   EQU    01010000B    ; FORMAT TRACK      (50H)
3422                    <1> INIT_CMD     EQU    01100000B    ;   INITIALIZE      (60H)
3423                    <1> SEEK_CMD     EQU    01110000B    ;      SEEK   (70H)
3424                    <1> DIAG_CMD     EQU    10010000B    ; DIAGNOSTIC (90H)
3425                    <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
3496 00005232 C705[608A0100]0000- <1>       mov    dword [HDPM_TBL_VEC], (DPT_SEGM*16)+HD0_DPT
3496 0000523A 0900              <1>
                                <1>       ;mov   word [HDPS_TBL_VEC],HD1_DPT      ; PARM TABLE DRIVE 81h
                                <1>       ;mov   word [HDPS_TBL_VEC+2],DPT_SEGM
3499 0000523C C705[648A0100]2000- <1>       mov    dword [HDPS_TBL_VEC], (DPT_SEGM*16)+HD1_DPT
3499 00005244 0900              <1>
                                <1>       ;mov   word [HDSM_TBL_VEC],HD2_DPT      ; PARM TABLE DRIVE 82h
                                <1>       ;mov   word [HDSM_TBL_VEC+2],DPT_SEGM
3502 00005246 C705[688A0100]4000- <1>       mov    dword [HDSM_TBL_VEC], (DPT_SEGM*16)+HD2_DPT
3502 0000524E 0900              <1>
                                <1>       ;mov   word [HDSS_TBL_VEC],HD3_DPT      ; PARM TABLE DRIVE 83h
                                <1>       ;mov   word [HDSS_TBL_VEC+2],DPT_SEGM
3505 00005250 C705[6C8A0100]6000- <1>       mov    dword [HDSS_TBL_VEC], (DPT_SEGM*16)+HD3_DPT
3505 00005258 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
0000525A BE[006E0000]           <1>       mov    esi, hd0_type
                                <1>       ;mov   cx, 4
0000525F B904000000             <1>       mov    ecx, 4
                                <1> hde_l:
00005264 AC                     <1>       lodsb
00005265 3C80                   <1>       cmp    al, 80h                  ; 8?h = existing
00005267 7206                   <1>       jb     short _L4
00005269 FE05[5C8A0100]         <1>       inc    byte [HF_NUM]       ; + 1 hard (fixed) disk drives
                                <1> _L4: ; 26/02/2015
0000526F 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
00005271 8A0D[5C8A0100]         <1>       mov    cl, [HF_NUM]
00005277 20C9                   <1>       and    cl, cl
00005279 740E                   <1>       jz     short POD_DONE
                                <1>       ;
0000527B B27F                   <1>       mov    dl, 7Fh
                                <1> hdc_reset1:
0000527D 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
0000527F B40D                   <1>       mov    ah, 0Dh ; ALTERNATE RESET
                                <1>       ;int   13h
00005281 E8A3FFFFFF             <1>       call   int13h
00005286 E2F5                   <1>       loop   hdc_reset1
00005288 F8                     <1>       clc    ; 29/05/2016
                                <1> POD_DONE:
00005289 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> ; 30/08/2020
                                <1> ; 29/08/2020
                                <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
0000528A 80642408FE             <1>       and   byte [esp+8], 11111110b  ; clear carry bit of eflags register
                                <1>       ; 16/05/2016
0000528F 1E                     <1>       push  ds
00005290 53                     <1>       push  ebx ; user's buffer address (virtual)
00005291 66BB1000               <1>       mov   bx, KDATA ; System (Kernel's) data segment
00005295 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>
00005297 8F05[4C960100]         <1>       pop    dword [user_buffer] ; 01/06/2016
                                <1>
0000529D C605[868F0100]00       <1>       mov    byte [scount], 0 ; sector count for transfer
000052A4 80FC03                 <1>       cmp    ah, 03h ; chs write
000052A7 7744                   <1>       ja     short int33h_2
000052A9 7407                   <1>       je     short int33h_0
000052AB 80FC02                 <1>       cmp    ah, 02h ; chs read
000052AE 726F                   <1>       jb     short int33h_5
000052B0 EB68                   <1>       jmp    short int33h_4
                                <1> int33h_0:
                                <1>       ; transfer user's buffer content to sector buffer
000052B2 51                     <1>       push   ecx
000052B3 0FB6C8                 <1>       movzx  ecx, al
                                <1> int33h_1:
000052B6 56                     <1>       push   esi
000052B7 8B35[4C960100]         <1>       mov    esi, [user_buffer]
                                <1>       ; esi = user's buffer address (virtual, ebx)
000052BD 57                     <1>       push   edi
000052BE 06                     <1>       push   es
000052BF 50                     <1>       push   eax
000052C0 66B81000               <1>       mov    ax, KDATA
000052C4 8EC0                   <1>       mov    es, ax
000052C6 BF00000700             <1>       mov    edi, Cluster_Buffer
000052CB C1E109                 <1>       shl    ecx, 9 ; * 512
000052CE E869C80000             <1>       call   transfer_from_user_buffer
000052D3 58                     <1>       pop    eax
000052D4 07                     <1>       pop    es
000052D5 5F                     <1>       pop    edi
000052D6 5E                     <1>       pop    esi
000052D7 59                     <1>       pop    ecx
000052D8 7345                   <1>       jnc    short int33h_5
000052DA 8B1D[4C960100]         <1>       mov    ebx, [user_buffer] ; 01/06/2016
000052E0 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
000052E1 804C240801             <1>       or     byte [esp+8], 1  ; set carry bit of eflags register
                                <1>       ; [esp+12] = ESP (user)
                                <1>       ; [esp+16] = SS (User)
000052E6 B8FF000000             <1>       mov    eax, 0FFh ; Unknown error !?
                                <1>       ;iretd
000052EB EB7E                   <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:
000052ED 80FC05                 <1>       cmp    ah, 05h ; format track
000052F0 770A                   <1>       ja     short int33h_3
000052F2 722B                   <1>       jb     short int33h_5
000052F4 51                     <1>       push   ecx
000052F5 B901000000             <1>       mov    ecx, 1
000052FA EBBA                   <1>       jmp    short int33h_1
                                <1> int33h_3:
000052FC 80FC1C                 <1>       cmp    ah, 1Ch ; LBA write
000052FF 771E                   <1>       ja     short int33h_5
00005301 74AF                   <1>       je     short int33h_0
00005303 80FC1B                 <1>       cmp    ah, 1Bh ; LBA read
00005306 7412                   <1>       je     short int33h_4
                                <1>       ; 29/08/2020
00005308 80FC08                 <1>       cmp    ah, 08h ; get disk parameters
                                <1>       ;jne   short int33h_5
0000530B 7405                   <1>       je     short int33h_10
0000530D 80FC15                 <1>       cmp    ah, 15h      ; read DASD type (get disk size)
00005310 750D                   <1>       jne    short int33h_5
                                <1> int33h_10:
                                <1>       ; 01/06/2016
00005312 8B1D[4C960100]         <1>       mov    ebx, [user_buffer] ; user's buffer address
00005318 EB0A                   <1>       jmp    short int33h_6
                                <1> int33h_4:
0000531A A2[868F0100]           <1>       mov    byte [scount], al ; <= 128 sectors
                                <1> int33h_5:
0000531F BB00000700             <1>       mov    ebx, Cluster_Buffer ; max. 65536 bytes
                                <1>                                  ; buf. addr: 70000h
                                <1>       ;mov   byte [ClusterBuffer_Valid], 0
```

```
                                <1> int33h_6:
00005324 1F                     <1>       pop    ds
00005325 9C                     <1>       pushfd
00005326 0E                     <1>       push   cs
00005327 E84D000000             <1>       call   DISK_IO
0000532C 2E8B1D[4C960100]       <1>       mov    ebx, [CS:user_buffer] ; 01/06/2016
00005333 723D                   <1>       jc     short int33h_9
                                <1>       ;
00005335 2E803D[868F0100]00     <1>       cmp    byte [CS:scount], 0
0000533D 762C                   <1>       jna    short int33h_7
                                <1>       ;
                                <1>       ; transfer sector buffer content to user's buffer
0000533F 06                     <1>       push   es
00005340 1E                     <1>       push   ds
00005341 50                     <1>       push   eax
00005342 66B81000               <1>       mov    ax, KDATA
00005346 8ED8                   <1>       mov    ds, ax
00005348 8EC0                   <1>       mov    es, ax
0000534A 51                     <1>       push   ecx
0000534B 56                     <1>       push   esi
0000534C 57                     <1>       push   edi
0000534D 0FB60D[868F0100]       <1>       movzx  ecx, byte [scount]
00005354 C1E109                 <1>       shl    ecx, 9 ; * 512 bytes
00005357 89DF                   <1>       mov    edi, ebx ; user's buffer address
00005359 BE00000700             <1>       mov    esi, Cluster_Buffer
0000535E E88FC70000             <1>       call   transfer_to_user_buffer
00005363 5F                     <1>       pop    edi
00005364 5E                     <1>       pop    esi
00005365 59                     <1>       pop    ecx
00005366 58                     <1>       pop    eax
00005367 1F                     <1>       pop    ds
00005368 07                     <1>       pop    es
00005369 7202                   <1>       jc     short int33h_8
                                <1> int33h_7:
0000536B 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
0000536C CF                     <1>       iretd
                                <1> int33h_8:
0000536D 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-
00005372 804C240801             <1>       or     byte [esp+8], 1  ; set carry bit of eflags register
                                <1>       ;iretd
00005377 EBF2                   <1>       jmp    short int33h_7 ; 07/01/2017
                                <1>
                                <1> ; 30/08/2020
                                <1> ; 09/12/2017
                                <1> ; 29/05/2016
                                <1> ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>
                                <1> DISK_IO:
00005379 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
0000537C 0F82FFEFFFFF           <1>       jb     DISKETTE_IO_1
                                <1> ;RET_2:
                                <1>       ;RETf  2                   ; BACK TO CALLER
                                <1> ;     retf   4
                                <1> A1:
00005382 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
00005383 80FA83                 <1>       CMP    DL,(80H + S_MAX_FILE - 1) ; 83h ; 30/08/2020
                                <1>       ;JA    short RET_2
00005386 7614                   <1>       jna    short _A0
                                <1>       ; 29/05/2016
00005388 1E                     <1>       push   ds
00005389 50                     <1>       push   eax ; 30/08/2020 (ax --> eax)
0000538A 66B81000               <1>       mov    ax, KDATA
0000538E 8ED8                   <1>       mov    ds, ax
00005390 58                     <1>       pop    eax ;
00005391 B4AA                   <1>         mov     ah, 0AAh        ; Hard disk drive not ready !
                                <1>                             ; (Programmer's guide to AMIBIOS, 1992)
00005393 8825[5B8A0100]         <1>       mov     byte [DISK_STATUS1], ah
00005399 1F                     <1>       pop    ds
0000539A EB38                   <1>       jmp    short RET_2
                                <1> _A0:
                                <1>       ; 18/01/2015
0000539C 08E4                   <1>       or     ah,ah
0000539E 743A                   <1>       jz     short A4
000053A0 80FC0D                 <1>       cmp    ah,0Dh ; Alternate reset
000053A3 7504                   <1>       jne    short A2
000053A5 28E4                   <1>       sub    ah,ah  ; Reset
000053A7 EB31                   <1>       jmp    short A4
                                <1> A2:
000053A9 80FC08                 <1>       CMP    AH,08H              ; GET PARAMETERS IS A SPECIAL CASE
                                <1>       ;JNZ   short A3
                                <1>         ;JMP    GET_PARM_N
000053AC 0F8452030000           <1>       je     GET_PARM_N
```

```
3947 000053B2 80FC15              <1> A3:   CMP    AH,15H              ; READ DASD TYPE IS ALSO
3948                              <1>       ;JNZ   short A4
3949                              <1>         ;JMP    READ_DASD_TYPE
3950 000053B5 0F84DB020000        <1>         je      READ_DASD_TYPE
3951                              <1>       ; 02/02/2015
3952 000053BB 80FC1D              <1>       cmp    ah, 1Dh                   ;(Temporary for Retro UNIX 386 v1)
3953                              <1>       ; 12/01/2015
3954 000053BE F5                  <1>       cmc
3955 000053BF 7319                <1>       jnc    short A4
3956                              <1> int33h_bad_cmd:
3957                              <1>       ; 16/05/2016
3958                              <1>       ; 30/01/2015
3959                              <1>       ; 29/05/2016
3960 000053C1 1E                  <1>       push   ds
3961 000053C2 6650                <1>       push   ax
3962 000053C4 66B81000            <1>       mov    ax, KDATA
3963 000053C8 8ED8                <1>       mov    ds, ax
3964 000053CA 6658                <1>       pop    ax
3965 000053CC B401                <1>       mov    ah, BAD_CMD
3966 000053CE 8825[5B8A0100]      <1>       mov     [DISK_STATUS1], ah ; BAD_CMD  ; COMMAND ERROR
3967                              <1>         ;jmp short RET_2
3968                              <1> RET_2:
3969                              <1>       ; (*) 29/05/2016
3970                              <1>       ; (*) retf 4
3971 000053D4 804C240801          <1>       or     byte [esp+8], 1 ; set carry bit of eflags register
3972 000053D9 CF                  <1>       iretd
3973                              <1> A4:                             ; SAVE REGISTERS DURING OPERATION
3974 000053DA C8080000            <1>       ENTER  8,0                ; SAVE (BP) AND MAKE ROOM FOR @CMD_BLOCK
3975 000053DE 53                  <1>       PUSH   eBX                ;  IN THE STACK, THE COMMAND BLOCK IS:
3976 000053DF 51                  <1>       PUSH   eCX                ;   @CMD_BLOCK == BYTE PTR [BP]-8
3977 000053E0 52                  <1>       PUSH   eDX
3978 000053E1 1E                  <1>       PUSH   DS
3979 000053E2 06                  <1>       PUSH   ES
3980 000053E3 56                  <1>       PUSH   eSI
3981 000053E4 57                  <1>       PUSH   eDI
3982                              <1>       ;;04/01/2015
3983                              <1>       ;;OR   AH,AH              ; CHECK FOR RESET
3984                              <1>       ;;JNZ  short A5
3985                              <1>       ;;MOV  DL,80H             ; FORCE DRIVE 80 FOR RESET
3986                              <1> ;;A5:
3987                              <1>       ;push  cs
3988                              <1>       ;pop   ds
3989                              <1>       ; 21/02/2015
3990 000053E5 6650                <1>       push   ax
3991 000053E7 66B81000            <1>       mov    ax, KDATA
3992 000053EB 8ED8                <1>       mov    ds, ax
3993 000053ED 8EC0                <1>       mov    es, ax
3994 000053EF 6658                <1>       pop    ax
3995 000053F1 E88D000000          <1>       CALL   DISK_IO_CONT       ; PERFORM THE OPERATION
3996                              <1>       ;;CALL DDS                ; ESTABLISH SEGMENT
3997 000053F6 8A25[5B8A0100]      <1>       MOV    AH,[DISK_STATUS1]  ; GET STATUS FROM OPERATION
3998                              <1>       ;(*) CMP AH,1             ; SET THE CARRY FLAG TO INDICATE
3999                              <1>       ;(*) CMC                  ; SUCCESS OR FAILURE
4000 000053FC 5F                  <1>       POP    eDI                ; RESTORE REGISTERS
4001 000053FD 5E                  <1>       POP    eSI
4002 000053FE 07                  <1>         POP     ES
4003 000053FF 1F                  <1>         POP     DS
4004 00005400 5A                  <1>       POP    eDX
4005 00005401 59                  <1>       POP    eCX
4006 00005402 5B                  <1>       POP    eBX
4007 00005403 C9                  <1>       LEAVE                     ; ADJUST (SP) AND RESTORE (BP)
4008                              <1>       ;RETf  2                  ; THROW AWAY SAVED FLAGS
4009                              <1>       ; (*) 29/05/2016
4010                              <1>       ; (*) retf 4
4011 00005404 80FC01              <1>       cmp    ah, 1
4012 00005407 7205                <1>       jc     short _A5
4013 00005409 804C240801          <1>       or     byte [esp+8], 1 ; set carry bit of eflags register
4014                              <1> _A5:
4015 0000540E CF                  <1>       iretd
4016                              <1>
4017                              <1> ; 21/02/2015
4018                              <1> ;       dw --> dd
4019                              <1> D1:                             ; FUNCTION TRANSFER TABLE
4020 0000540F [D2550000]          <1>       dd     DISK_RESET         ; 000H
4021 00005413 [49560000]          <1>       dd     RETURN_STATUS      ; 001H
4022 00005417 [56560000]          <1>       dd     DISK_READ          ; 002H
4023 0000541B [5F560000]          <1>       dd     DISK_WRITE         ; 003H
4024 0000541F [68560000]          <1>       dd     DISK_VERF          ; 004H
4025 00005423 [80560000]          <1>       dd     FMT_TRK            ; 005H
4026 00005427 [C8550000]          <1>       dd     BAD_COMMAND        ; 006H FORMAT BAD SECTORS
4027 0000542B [C8550000]          <1>       dd     BAD_COMMAND        ; 007H FORMAT DRIVE
4028 0000542F [C8550000]          <1>       dd     BAD_COMMAND        ; 008H RETURN PARAMETERS
4029 00005433 [9D570000]          <1>       dd     INIT_DRV           ; 009H
4030 00005437 [FC570000]          <1>       dd     RD_LONG            ; 00AH
4031 0000543B [05580000]          <1>       dd     WR_LONG            ; 00BH
4032 0000543F [0E580000]          <1>       dd     DISK_SEEK          ; 00CH
4033 00005443 [D2550000]          <1>       dd     DISK_RESET         ; 00DH
4034 00005447 [C8550000]          <1>       dd     BAD_COMMAND        ; 00EH READ BUFFER
4035 0000544B [C8550000]          <1>       dd     BAD_COMMAND        ; 00FH WRITE BUFFER
4036 0000544F [36580000]          <1>       dd     TST_RDY            ; 010H
4037 00005453 [5A580000]          <1>       dd     HDISK_RECAL        ; 011H
4038 00005457 [C8550000]          <1>       dd     BAD_COMMAND        ; 012H MEMORY DIAGNOSTIC
4039 0000545B [C8550000]          <1>       dd     BAD_COMMAND        ; 013H DRIVE DIAGNOSTIC
4040 0000545F [90580000]          <1>       dd     CTLR_DIAGNOSTIC    ; 014H CONTROLLER DIAGNOSTIC
4041                              <1>       ; 02/02/2015 (Temporary - Retro UNIX 386 v1 - DISK I/O test)
4042 00005463 [C8550000]          <1>       dd     BAD_COMMAND        ; 015h
4043 00005467 [C8550000]          <1>       dd     BAD_COMMAND        ; 016h
4044 0000546B [C8550000]          <1>       dd     BAD_COMMAND        ; 017h
4045 0000546F [C8550000]          <1>       dd     BAD_COMMAND        ; 018h
4046 00005473 [C8550000]          <1>       dd     BAD_COMMAND        ; 019h
4047 00005477 [C8550000]          <1>       dd     BAD_COMMAND        ; 01Ah
4048 0000547B [56560000]          <1>       dd     DISK_READ          ; 01Bh ; LBA read
4049 0000547F [5F560000]          <1>       dd     DISK_WRITE         ; 01Ch ; LBA write
4050                              <1> D1L     EQU    $ - D1
4051                              <1>
```

```
4052                            <1> DISK_IO_CONT:
4053                            <1>       ;;CALL DDS                ; ESTABLISH SEGMENT
4054 00005483 80FC01            <1>       CMP    AH,01H             ; RETURN STATUS
4055                            <1>       ;;JNZ  short SU0
4056                            <1>         ;;JMP      RETURN_STATUS
4057 00005486 0F84BD010000      <1>       je     RETURN_STATUS
4058                            <1> SU0:
4059 0000548C C605[5B8A0100]00  <1>       MOV    byte [DISK_STATUS1],0     ; RESET THE STATUS INDICATOR
4060                            <1>       ;;PUSH BX                ; SAVE DATA ADDRESS
4061                            <1>       ;mov   si, bx ;; 14/02/2015
4062 00005493 89DE              <1>       mov    esi, ebx ; 21/02/2015
4063 00005495 8A1D[5C8A0100]    <1>       MOV    BL,[HF_NUM]        ; GET NUMBER OF DRIVES
4064                            <1>       ;; 04/01/2015
4065                            <1>       ;;PUSH AX
4066 0000549B 80E27F            <1>       AND    DL,7FH             ; GET DRIVE AS 0 OR 1
4067                            <1>                                 ; (get drive number as 0 to 3)
4068 0000549E 38D3              <1>       CMP    BL,DL
4069                            <1>         ;;JBE   BAD_COMMAND_POP         ; INVALID DRIVE
4070 000054A0 0F8622010000      <1>         jbe     BAD_COMMAND ;; 14/02/2015
4071                            <1>         ;
4072                            <1>       ;;03/01/2015
4073 000054A6 29DB              <1>       sub    ebx, ebx
4074 000054A8 88D3              <1>       mov    bl, dl
4075                            <1>       ;sub   bh, bh
4076 000054AA 883D[708A0100]    <1>       mov    [LBAMode], bh      ; 0
4077                            <1>       ;;test byte [bx+hd0_type], 1     ; LBA ready ?
4078                            <1>       ;test  byte [ebx+hd0_type], 1
4079                            <1>       ;jz    short su1          ; no
4080                            <1>       ;inc   byte [LBAMode]
4081                            <1> ;su1:
4082                            <1>       ; 21/02/2015 (32 bit modification)
4083                            <1>       ;04/01/2015
4084 000054B0 6650              <1>       push   ax ; ***
4085                            <1>       ;PUSH  ES ; **
4086 000054B2 6652              <1>       PUSH   DX ; *
4087 000054B4 6650              <1>       push   ax
4088 000054B6 E8BB060000        <1>       CALL   GET_VEC            ; GET DISK PARAMETERS
4089                            <1>       ; 02/02/2015
4090                            <1>       ;mov   ax, [ES:BX+16] ; I/O port base address (1F0h, 170h)
4091 000054BB 668B4310          <1>       mov    ax, [ebx+16]
4092 000054BF 66A3[F06D0000]    <1>       mov    [HF_PORT], ax
4093                            <1>       ;mov   dx, [ES:BX+18] ; control port address (3F6h, 376h)
4094 000054C5 668B5312          <1>       mov    dx, [ebx+18]
4095 000054C9 668915[F26D0000]  <1>       mov    [HF_REG_PORT], dx
4096                            <1>       ;mov   al, [ES:BX+20] ; head register upper nibble (A0h,B0h,E0h,F0h)
4097 000054D0 8A4314            <1>       mov    al, [ebx+20]
4098                            <1>       ; 23/02/2015
4099 000054D3 A840              <1>       test   al, 40h       ; LBA bit (bit 6)
4100 000054D5 7406              <1>       jz     short su1
4101 000054D7 FE05[708A0100]    <1>       inc    byte [LBAMode] ; 1
4102                            <1> su1:
4103 000054DD C0E804            <1>       shr    al, 4
4104 000054E0 2401              <1>       and    al, 1
4105 000054E2 A2[F46D0000]      <1>       mov    [hf_m_s], al
4106                            <1>       ;
4107                            <1>       ; 03/01/2015
4108                            <1>       ;MOV   AL,byte [ES:BX+8]   ; GET CONTROL BYTE MODIFIER
4109 000054E7 8A4308            <1>       mov    al, [ebx+8]
4110                            <1>       ;MOV   DX,[HF_REG_PORT]    ; Device Control register
4111 000054EA EE                <1>       OUT    DX,AL               ; SET EXTRA HEAD OPTION
4112                            <1>                                  ; Control Byte:  (= 08h, here)
4113                            <1>                                  ; bit 0 - 0
4114                            <1>                                  ; bit 1 - nIEN (1 = disable irq)
4115                            <1>                                  ; bit 2 - SRST (software RESET)
4116                            <1>                                  ; bit 3 - use extra heads (8 to 15)
4117                            <1>                                  ;         -always set to 1-
4118                            <1>                                  ; (bits 3 to 7 are reserved
4119                            <1>                                  ;          for ATA devices)
4120 000054EB 8A25[5D8A0100]    <1>       MOV    AH,[CONTROL_BYTE]   ; SET EXTRA HEAD OPTION IN
4121 000054F1 80E4C0            <1>       AND    AH,0C0H             ; CONTROL BYTE
4122 000054F4 08C4              <1>       OR     AH,AL
4123 000054F6 8825[5D8A0100]    <1>       MOV    [CONTROL_BYTE],AH
4124                            <1>       ; 04/01/2015
4125 000054FC 6658              <1>       pop    ax
4126 000054FE 665A              <1>       pop    dx ; * ;; 14/02/2015
4127 00005500 20E4              <1>       and    ah, ah ; Reset function ?
4128 00005502 7507              <1>       jnz    short su2
4129                            <1>       ;;pop  dx ; * ;; 14/02/2015
4130                            <1>       ;pop   es ; **
4131 00005504 6658              <1>       pop    ax ; ***
4132                            <1>       ;;pop  bx
4133 00005506 E9C7000000        <1>         jmp     DISK_RESET
4134                            <1> su2:
4135 0000550B 803D[708A0100]00  <1>       cmp    byte [LBAMode], 0
4136 00005512 7662              <1>       jna    short su3
4137                            <1>       ;
4138                            <1>       ; 02/02/2015 (LBA read/write function calls)
4139 00005514 80FC1B            <1>       cmp    ah, 1Bh
4140 00005517 720B              <1>       jb     short lbarw1
4141 00005519 80FC1C            <1>       cmp    ah, 1Ch
4142 0000551C 775D              <1>       ja     short invldfnc
4143                            <1>       ;;pop  dx ; * ; 14/02/2015
4144                            <1>       ;mov   ax, cx ; Lower word of LBA address (bits 0-15)
4145 0000551E 89C8              <1>       mov    eax, ecx ; LBA address (21/02/2015)
4146                            <1>       ;; 14/02/2015
4147 00005520 88D1              <1>       mov    cl, dl ; 14/02/2015
4148                            <1>       ;;mov  dx, bx
4149                            <1>       ;mov   dx, si ; higher word of LBA address (bits 16-23)
4150                            <1>       ;;mov  bx, di
4151                            <1>       ;mov   si, di ; Buffer offset
4152 00005522 EB32              <1>       jmp    short lbarw2
4153                            <1> lbarw1:
4154                            <1>       ; convert CHS to LBA
4155                            <1>       ;
4156                            <1>       ; LBA calculation - AWARD BIOS - 1999 - AHDSK.ASM
```

```
4157                                    <1>      ; LBA = "# of Heads" * Sectors/Track * Cylinder + Head * Sectors/Track
4158                                    <1>      ;      + Sector - 1
4159 00005524 6652                      <1>      push   dx ; * ;; 14/02/2015
4160                                    <1>      ;xor   dh, dh
4161 00005526 31D2                      <1>      xor    edx, edx
4162                                    <1>      ;mov   dl, [ES:BX+14]      ; sectors per track (logical)
4163 00005528 8A530E                    <1>      mov    dl, [ebx+14]
4164                                    <1>      ;xor   ah, ah
4165 0000552B 31C0                      <1>      xor    eax, eax
4166                                    <1>      ;mov   al, [ES:BX+2]; heads (logical)
4167 0000552D 8A4302                    <1>      mov    al, [ebx+2]
4168 00005530 FEC8                      <1>      dec    al
4169 00005532 6640                      <1>      inc    ax           ; 0 =  256
4170 00005534 66F7E2                    <1>      mul    dx
4171                                    <1>             ; AX = # of Heads" * Sectors/Track
4172 00005537 6689CA                    <1>      mov    dx, cx
4173                                    <1>      ;and   cx, 3Fh       ; sector  (1 to 63)
4174 0000553A 83E13F                    <1>      and    ecx, 3fh
4175 0000553D 86D6                      <1>      xchg   dl, dh
4176 0000553F C0EE06                    <1>      shr    dh, 6
4177                                    <1>             ; DX = cylinder (0 to 1023)
4178                                    <1>      ;mul   dx
4179                                    <1>             ; DX:AX = # of Heads" * Sectors/Track * Cylinder
4180 00005542 F7E2                      <1>      mul    edx
4181 00005544 FEC9                      <1>      dec    cl  ; sector - 1
4182                                    <1>      ;add   ax, cx
4183                                    <1>      ;adc   dx, 0
4184                                    <1>             ; DX:AX = # of Heads" * Sectors/Track * Cylinder + Sector -1
4185 00005546 01C8                      <1>      add    eax, ecx
4186 00005548 6659                      <1>      pop    cx ; * ; ch = head, cl = drive number (zero based)
4187                                    <1>      ;push  dx
4188                                    <1>      ;push  ax
4189 0000554A 50                        <1>      push   eax
4190                                    <1>      ;mov   al, [ES:BX+14]      ; sectors per track (logical)
4191 0000554B 8A430E                    <1>      mov    al, [ebx+14]
4192 0000554E F6E5                      <1>      mul    ch
4193                                    <1>             ;  AX = Head * Sectors/Track
4194 00005550 0FB7C0                    <1>        movzx      eax, ax ; 09/12/2017
4195                                    <1>      ;pop   dx
4196 00005553 5A                        <1>      pop    edx
4197                                    <1>      ;add   ax, dx
4198                                    <1>      ;pop   dx
4199                                    <1>      ;adc   dx, 0 ; add carry bit
4200 00005554 01D0                      <1>      add    eax, edx
4201                                    <1> lbarw2:
4202 00005556 29D2                      <1>      sub    edx, edx ; 21/02/2015
4203 00005558 88CA                      <1>      mov    dl, cl ; 21/02/2015
4204 0000555A C645F800                  <1>        mov     byte [CMD_BLOCK], 0 ; Features Register
4205                                    <1>                         ; NOTE: Features register (1F1h, 171h)
4206                                    <1>                         ; is not used for ATA device R/W functions.
4207                                    <1>                         ; It is old/obsolete 'write precompensation'
4208                                    <1>                         ; register and error register
4209                                    <1>                         ; for old ATA/IDE devices.
4210                                    <1>      ; 18/01/2014
4211                                    <1>      ;mov   ch, [hf_m_s] ; Drive 0 (master) or 1 (slave)
4212 0000555E 8A0D[F46D0000]            <1>      mov    cl, [hf_m_s]
4213                                    <1>      ;shl   ch, 4        ; bit 4 (drive bit)
4214                                    <1>      ;or    ch, 0E0h     ; bit 5 = 1
4215                                    <1>                         ; bit 6 = 1 = LBA mode
4216                                    <1>                         ; bit 7 = 1
4217 00005564 80C90E                    <1>      or     cl, 0Eh ; 1110b
4218                                    <1>      ;and   dh, 0Fh             ; LBA byte 4 (bits 24 to 27)
4219 00005567 25FFFFFF0F                <1>      and    eax, 0FFFFFFFh
4220 0000556C C1E11C                    <1>      shl    ecx, 28 ; 21/02/2015
4221                                    <1>      ;or    dh, ch
4222 0000556F 09C8                      <1>      or     eax, ecx
4223                                    <1>      ;;mov  [CMD_BLOCK+2], al ; LBA byte 1 (bits 0 to 7)
4224                                    <1>                               ; (Sector Number Register)
4225                                    <1>      ;;mov  [CMD_BLOCK+3], ah ; LBA byte 2 (bits 8 to 15)
4226                                    <1>                               ; (Cylinder Low Register)
4227                                    <1>      ;mov   [CMD_BLOCK+2], ax ; LBA byte 1, 2
4228                                    <1>      ;mov   [CMD_BLOCK+4], dl ; LBA byte 3 (bits 16 to 23)
4229                                    <1>                               ; (Cylinder High Register)
4230                                    <1>      ;;mov  [CMD_BLOCK+5], dh ; LBA byte 4 (bits 24 to 27)
4231                                    <1>                               ; (Drive/Head Register)
4232                                    <1>
4233                                    <1>      ;mov   [CMD_BLOCK+4], dx ; LBA byte 4, LBA & DEV select bits
4234 00005571 8945FA                    <1>      mov    [CMD_BLOCK+2], eax ; 21/02/2015
4235                                    <1>      ;14/02/2015
4236                                    <1>      ;mov   dl, cl ; Drive number (INIT_DRV)
4237 00005574 EB38                      <1>      jmp    short su4
4238                                    <1> su3:
4239                                    <1>      ; 02/02/2015
4240                                    <1>      ; (Temporary functions 1Bh & 1Ch are not valid for CHS mode)
4241 00005576 80FC14                    <1>      cmp    ah, 14h
4242 00005579 7604                      <1>      jna    short chsfnc
4243                                    <1> invldfnc:
4244                                    <1>        ; 14/02/2015
4245                                    <1>      ;pop   es ; **
4246 0000557B 6658                      <1>        pop     ax ; ***
4247                                    <1>        ;jmp     short BAD_COMMAND_POP
4248 0000557D EB49                      <1>        jmp     short BAD_COMMAND
4249                                    <1> chsfnc:
4250                                    <1>      ;MOV   AX,[ES:BX+5]        ; GET WRITE PRE-COMPENSATION CYLINDER
4251 0000557F 668B4305                  <1>      mov    ax, [ebx+5]
4252 00005583 66C1E802                  <1>      SHR    AX,2
4253 00005587 8845F8                    <1>      MOV    [CMD_BLOCK],AL
4254                                    <1>      ;;MOV  AL,[ES:BX+8]        ; GET CONTROL BYTE MODIFIER
4255                                    <1>      ;;PUSH DX
4256                                    <1>      ;;MOV  DX,[HF_REG_PORT]
4257                                    <1>      ;;OUT  DX,AL               ; SET EXTRA HEAD OPTION
4258                                    <1>      ;;POP  DX ; *
4259                                    <1>      ;;POP  ES ; **
4260                                    <1>      ;;MOV  AH,[CONTROL_BYTE]   ; SET EXTRA HEAD OPTION IN
4261                                    <1>      ;;AND  AH,0C0H             ; CONTROL BYTE
```

```
                        <1>       ;;OR   AH,AL
                        <1>       ;;MOV  [CONTROL_BYTE],AH
                        <1>       ;
0000558A 88C8           <1>       MOV    AL,CL              ; GET SECTOR NUMBER
0000558C 243F           <1>       AND    AL,3FH
0000558E 8845FA         <1>       MOV    [CMD_BLOCK+2],AL
00005591 886DFB         <1>       MOV    [CMD_BLOCK+3],CH   ; GET CYLINDER NUMBER
00005594 88C8           <1>       MOV    AL,CL
00005596 C0E806         <1>       SHR    AL,6
00005599 8845FC         <1>       MOV    [CMD_BLOCK+4],AL   ; CYLINDER HIGH ORDER 2 BITS
                        <1>       ;;05/01/2015
                        <1>       ;;MOV  AL,DL              ; DRIVE NUMBER
0000559C A0[F46D0000]   <1>       mov    al, [hf_m_s]
000055A1 C0E004         <1>       SHL    AL,4
000055A4 80E60F         <1>       AND    DH,0FH             ; HEAD NUMBER
000055A7 08F0           <1>       OR     AL,DH
                        <1>       ;OR    AL,80H or 20H
000055A9 0CA0           <1>       OR     AL,80h+20h         ; ECC AND 512 BYTE SECTORS
000055AB 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
000055AE 6658           <1>         pop     ax ; ***
000055B0 8845F9         <1>         mov     [CMD_BLOCK+1], al
000055B3 29DB           <1>         sub  ebx, ebx
000055B5 88E3           <1>       mov     bl, ah
                        <1>         ;xor     bh, bh
000055B7 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
000055BB 83FB74         <1>       cmp    ebx, D1L
000055BE 7308           <1>       jnb    short BAD_COMMAND
                        <1>         ;xchg    bx, si
000055C0 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]
000055C2 FFA6[0F540000] <1>       jmp    dword [esi+D1]
                        <1> ;;BAD_COMMAND_POP:
                        <1> ;;    POP    AX
                        <1> ;;    POP    BX
                        <1> BAD_COMMAND:
000055C8 C605[5B8A0100]01 <1>         MOV     byte [DISK_STATUS1],BAD_CMD  ; COMMAND ERROR
000055CF B000           <1>       MOV    AL,0
000055D1 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:
000055D2 FA             <1>       CLI
000055D3 E4A1           <1>       IN     AL,INTB01          ; GET THE MASK REGISTER
                        <1>       ;JMP   $+2
                        <1>       IODELAY
000055D5 EB00           <2>  jmp short $+2
000055D7 EB00           <2>  jmp short $+2
                        <1>       ;AND   AL,0BFH             ; ENABLE FIXED DISK INTERRUPT
000055D9 243F           <1>       and    al,3Fh             ; 22/12/2014 (IRQ 14 & IRQ 15)
000055DB E6A1           <1>       OUT    INTB01,AL
000055DD FB             <1>       STI                       ; START INTERRUPTS
                        <1>       ; 14/02/2015
000055DE 6689D7         <1>       mov    di, dx
                        <1>       ; 04/01/2015
                        <1>       ;xor   di,di
                        <1> drst0:
000055E1 B004           <1>       MOV    AL,04H  ; bit 2 - SRST
                        <1>       ;MOV   DX,HF_REG_PORT
000055E3 668B15[F26D0000] <1>     MOV    DX,[HF_REG_PORT]
000055EA 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
000055EB B902000000     <1>         mov  ecx, 2 ; 21/02/2015
000055F0 E851CEFFFF     <1>       call    WAITF                   ; (Award Bios 1999 - WAIT_REFRESH,
                        <1>                                         ; 40 micro seconds)
000055F5 A0[5D8A0100]   <1>       mov    al,[CONTROL_BYTE]
000055FA 240F           <1>       AND    AL,0FH             ; SET HEAD OPTION
000055FC EE             <1>       OUT    DX,AL              ; TURN RESET OFF
000055FD E86A040000     <1>       CALL   NOT_BUSY
00005602 7515           <1>       JNZ    short DRERR        ; TIME OUT ON RESET
```

```
4365 00005604 668B15[F06D0000]    <1>       MOV    DX,[HF_PORT]
4366 0000560B FEC2                <1>       inc    dl  ; HF_PORT+1
4367                              <1>       ; 02/01/2015 - Award BIOS 1999 - AHDSK.ASM
4368                              <1>         ;mov     cl, 10
4369 0000560D B90A000000          <1>         mov     ecx, 10 ; 21/02/2015
4370                              <1> drst1:
4371 00005612 EC                  <1>       IN     AL,DX              ; GET RESET STATUS
4372 00005613 3C01                <1>       CMP    AL,1
4373                              <1>       ; 04/01/2015
4374 00005615 740A                <1>       jz     short drst2
4375                              <1>       ;JNZ   short DRERR        ; BAD RESET STATUS
4376                              <1>              ; Drive/Head Register - bit 4
4377 00005617 E2F9                <1>       loop   drst1
4378                              <1> DRERR:
4379 00005619 C605[5B8A0100]05    <1>       MOV    byte [DISK_STATUS1],BAD_RESET ; CARD FAILED
4380 00005620 C3                  <1>       RETn
4381                              <1> drst2:
4382                              <1>       ; 14/02/2015
4383 00005621 6689FA              <1>       mov    dx,di
4384                              <1> ;drst3:
4385                              <1> ;     ; 05/01/2015
4386                              <1> ;     shl    di,1
4387                              <1> ;     ; 04/01/2015
4388                              <1> ;     mov    ax,[di+hd_cports]
4389                              <1> ;     cmp    ax,[HF_REG_PORT]
4390                              <1> ;     je     short drst4
4391                              <1> ;     mov    [HF_REG_PORT], ax
4392                              <1> ;     ; 03/01/2015
4393                              <1> ;     mov    ax,[di+hd_ports]
4394                              <1> ;       mov     [HF_PORT], ax
4395                              <1> ;     ; 05/01/2014
4396                              <1> ;     shr    di,1
4397                              <1> ;     ; 04/01/2015
4398                              <1> ;     jmp    short drst0  ; reset other controller
4399                              <1> ;drst4:
4400                              <1> ;     ; 05/01/2015
4401                              <1> ;     shr    di,1
4402                              <1> ;     mov    al,[di+hd_dregs]
4403                              <1> ;     and    al,10h ; bit 4 only
4404                              <1> ;     shr    al,4 ; bit 4  -> bit 0
4405                              <1> ;     mov    [hf_m_s], al ; (0 = master, 1 = slave)
4406                              <1>       ;
4407 00005624 A0[F46D0000]        <1>       mov    al, [hf_m_s] ; 18/01/2015
4408 00005629 A801                <1>       test   al,1
4409                              <1> ;     jnz    short drst6
4410 0000562B 7516                <1>         jnz     short drst4
4411 0000562D 8065FDEF            <1>       AND     byte [CMD_BLOCK+5],0EFH ; SET TO DRIVE 0
4412                              <1> ;drst5:
4413                              <1> drst3:
4414 00005631 E867010000          <1>       CALL   INIT_DRV           ; SET MAX HEADS
4415                              <1>       ;mov   dx,di
4416 00005636 E81F020000          <1>       CALL   HDISK_RECAL        ; RECAL TO RESET SEEK SPEED
4417                              <1>       ; 04/01/2014
4418                              <1> ;     inc    di
4419                              <1> ;     mov    dx,di
4420                              <1> ;     cmp    dl,[HF_NUM]
4421                              <1> ;     jb     short drst3
4422                              <1> ;DRE:
4423 0000563B C605[5B8A0100]00    <1>       MOV    byte [DISK_STATUS1],0     ; IGNORE ANY SET UP ERRORS
4424 00005642 C3                  <1>       RETn
4425                              <1> ;drst6:
4426                              <1> drst4:             ; Drive/Head Register - bit 4
4427 00005643 804DFD10            <1>       OR      byte [CMD_BLOCK+5],010H ; SET TO DRIVE 1
4428                              <1>         ;jmp    short drst5
4429 00005647 EBE8                <1>         jmp     short drst3
4430                              <1>
4431                              <1> ;----------------------------------------
4432                              <1> ;     DISK STATUS ROUTINE  (AH = 01H) :
4433                              <1> ;----------------------------------------
4434                              <1>
4435                              <1> RETURN_STATUS:
4436 00005649 A0[5B8A0100]        <1>       MOV    AL,[DISK_STATUS1]   ; OBTAIN PREVIOUS STATUS
4437 0000564E C605[5B8A0100]00    <1>         MOV     byte [DISK_STATUS1],0   ; RESET STATUS
4438 00005655 C3                  <1>       RETn
4439                              <1>
4440                              <1> ;----------------------------------------
4441                              <1> ;     DISK READ ROUTINE    (AH = 02H) :
4442                              <1> ;----------------------------------------
4443                              <1>
4444                              <1> DISK_READ:
4445 00005656 C645FE20            <1>       MOV    byte [CMD_BLOCK+6],READ_CMD
4446 0000565A E986020000          <1>         JMP     COMMANDI
4447                              <1>
4448                              <1> ;----------------------------------------
4449                              <1> ;     DISK WRITE ROUTINE   (AH = 03H) :
4450                              <1> ;----------------------------------------
4451                              <1>
4452                              <1> DISK_WRITE:
4453 0000565F C645FE30            <1>       MOV    byte [CMD_BLOCK+6],WRITE_CMD
4454 00005663 E9D8020000          <1>         JMP     COMMANDO
4455                              <1>
4456                              <1> ;----------------------------------------
4457                              <1> ;     DISK VERIFY       (AH = 04H) :
4458                              <1> ;----------------------------------------
4459                              <1>
4460                              <1> DISK_VERF:
4461 00005668 C645FE40            <1>       MOV    byte [CMD_BLOCK+6],VERIFY_CMD
4462 0000566C E846030000          <1>       CALL   COMMAND
4463 00005671 750C                <1>       JNZ    short VERF_EXIT           ; CONTROLLER STILL BUSY
4464 00005673 E8B8030000          <1>       CALL   _WAIT               ; (Original: CALL WAIT)
4465 00005678 7505                <1>       JNZ    short VERF_EXIT           ; TIME OUT
4466 0000567A E845040000          <1>       CALL   CHECK_STATUS
4467                              <1> VERF_EXIT:
4468 0000567F C3                  <1>       RETn
4469                              <1>
```

```
4470                            <1> ;----------------------------------------
4471                            <1> ;     FORMATTING        (AH = 05H) :
4472                            <1> ;----------------------------------------
4473                            <1>
4474                            <1> FMT_TRK:                        ; FORMAT TRACK      (AH = 005H)
4475 00005680 C645FE50         <1>       MOV    byte [CMD_BLOCK+6],FMTTRK_CMD
4476                            <1>       ;PUSH  ES
4477                            <1>       ;PUSH  BX
4478 00005684 53               <1>       push   ebx
4479 00005685 E8EC040000       <1>       CALL   GET_VEC              ; GET DISK PARAMETERS ADDRESS
4480                            <1>       ;MOV   AL,[ES:BX+14]        ; GET SECTORS/TRACK
4481 0000568A 8A430E           <1>       mov    al, [ebx+14]
4482 0000568D 8845F9           <1>       MOV    [CMD_BLOCK+1],AL     ; SET SECTOR COUNT IN COMMAND
4483 00005690 5B               <1>       pop    ebx
4484                            <1>       ;POP   BX
4485                            <1>       ;POP   ES
4486 00005691 E9B1020000       <1>         JMP     CMD_OF                  ; GO EXECUTE THE COMMAND
4487                            <1>
4488                            <1> ; 30/08/2020
4489                            <1>
4490                            <1> ;----------------------------------------
4491                            <1> ;     READ DASD TYPE           (AH = 15H) :
4492                            <1> ;----------------------------------------
4493                            <1>
4494                            <1> READ_DASD_TYPE:
4495                            <1> READ_D_T:                       ; GET DRIVE PARAMETERS
4496 00005696 1E               <1>       PUSH   DS                   ; SAVE REGISTERS
4497                            <1>       ;PUSH  ES
4498 00005697 53               <1>       PUSH   eBX
4499                            <1>       ;CALL  DDS                  ; ESTABLISH ADDRESSING
4500                            <1>       ;push  cs
4501                            <1>       ;pop   ds
4502 00005698 66BB1000         <1>         mov  bx, KDATA
4503 0000569C 8EDB             <1>       mov    ds, bx
4504                            <1>       ;mov   es, bx
4505 0000569E C605[5B8A0100]00 <1>       MOV     byte [DISK_STATUS1],0
4506 000056A5 8A1D[5C8A0100]   <1>       MOV    BL,[HF_NUM]          ; GET NUMBER OF DRIVES
4507 000056AB 80E27F           <1>       AND    DL,7FH               ; GET DRIVE NUMBER
4508 000056AE 38D3             <1>       CMP    BL,DL
4509 000056B0 763C             <1>       JBE    short RDT_NOT_PRESENT     ; RETURN DRIVE NOT PRESENT
4510 000056B2 0FB6C2           <1>       movzx  eax, dl ; 28/08/2020
4511 000056B5 E8BC040000       <1>       CALL   GET_VEC              ; GET DISK PARAMETERS ADDRESS
4512                            <1>
4513                            <1>       ; 28/08/2020 - TRDOS 386 v2
4514 000056BA F6431440         <1>       test   byte [ebx+20], 40h  ; LBA bit (bit 6)
4515 000056BE 751D             <1>       jnz    short RDT3 ; LBA disk (may be > 8GB)
4516                            <1>
4517                            <1>       ;MOV   AL,[ES:BX+2]         ; HEADS
4518 000056C0 8A4302           <1>       mov    al, [ebx+2]
4519                            <1>       ;MOV   CL,[ES:BX+14]
4520 000056C3 8A4B0E           <1>       mov    cl, [ebx+14]
4521 000056C6 F6E9             <1>       IMUL   CL                   ; * NUMBER OF SECTORS
4522                            <1>       ;MOV   CX,[ES:BX]           ; MAX NUMBER OF CYLINDERS
4523 000056C8 668B0B           <1>       mov    cx, [ebx]
4524                            <1>       ;
4525                            <1>       ; 02/01/2015
4526                            <1>       ; ** leave the last cylinder as reserved for diagnostics **
4527                            <1>       ; (Also in Award BIOS - 1999, AHDSK.ASM, FUN15 -> sub ax, 1)
4528 000056CB 6649             <1>       DEC    CX                   ; LEAVE ONE FOR DIAGNOSTICS
4529                            <1>       ;
4530 000056CD 66F7E9           <1>       IMUL   CX                   ; NUMBER OF SECTORS
4531 000056D0 6689D1           <1>       MOV    CX,DX                ; HIGH ORDER HALF
4532 000056D3 6689C2           <1>       MOV    DX,AX                ; LOW ORDER HALF
4533                            <1>       ;SUB   AX,AX
4534                            <1>       ; 28/08/2020
4535                            <1>       ;sub   al, al
4536                            <1> RDT4:
4537 000056D6 29C0             <1>       sub    eax, eax ; 28/08/2020
4538                            <1>       ;
4539 000056D8 B403             <1>       MOV    AH,03H               ; INDICATE FIXED DISK
4540                            <1>       ; 30/08/2020 (clc is not needed here)
4541                            <1>       ;and   byte [esp+8], 0FEh ; clear carry bit of eflags register
4542                            <1> RDT2:
4543 000056DA 5B               <1>       POP    eBX                  ; RESTORE REGISTERS
4544                            <1>       ;POP   ES
4545 000056DB 1F               <1>       POP    DS
4546                            <1>       ; (*) CLC                   ; CLEAR CARRY
4547                            <1>       ;RETf  2
4548                            <1>       ; (*) 29/05/2016
4549                            <1>       ; (*) retf 4
4550                            <1>       ; 30/08/2020 (clc is not needed here)
4551                            <1>       ;and   byte [esp+8], 0FEh ; clear carry bit of eflags register
4552 000056DC CF               <1>       iretd
4553                            <1>
4554                            <1> RDT3: ; 28/08/2020
4555                            <1>       ; (use the result of INT 13h, function 48h as disk size)
4556                            <1>       ; eax = al = zero based hard disk number
4557                            <1>       ; 29/08/2020
4558                            <1>       ;add   al, 2 ; hd0 = physical disk drive 2
4559 000056DD C0E002           <1>       shl    al, 2 ; * 4
4560                            <1> RDT5:
4561                            <1>       ;add   eax, drv.size
4562 000056E0 05[366E0000]     <1>       add    eax, drv.size+8 ; 29/08/2020
4563 000056E5 668B10           <1>       mov    dx, [eax]   ; low word of disk size
4564 000056E8 668B4802         <1>       mov    cx, [eax+2] ; high word of disk size
4565                            <1>       ;sub   eax, eax
4566 000056EC EBE8             <1>       jmp    short RDT4
4567                            <1>
4568                            <1> RDT_NOT_PRESENT:
4569                            <1>       ; 30/08/2020
4570 000056EE C605[5B8A0100]AA <1>       mov    byte [DISK_STATUS1], NOT_RDY ; DRIVE NOT READY
4571                            <1>
4572                            <1>       ;SUB   AX,AX                ; DRIVE NOT PRESENT RETURN
4573 000056F5 29C0             <1>       sub    eax, eax ; 30/08/2020
4574 000056F7 6689C1           <1>       MOV    CX,AX                ; ZERO BLOCK COUNT
```

```
4575 000056FA 6689C2              <1>       MOV    DX,AX
4576                              <1>       ; 30/08/2020
4577 000056FD 804C241001          <1>       or     byte [esp+16], 1 ; set carry bit of eflags register
4578                              <1>              ; cf = 1 -> ah = 0, drive not ready, disk type = 0
4579 00005702 EBD6                <1>       JMP    short RDT2
4580                              <1>
4581                              <1> ; 28/05/2016
4582                              <1> ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)
4583                              <1>
4584                              <1> ;----------------------------------------
4585                              <1> ;     GET PARAMETERS           (AH = 08H) :
4586                              <1> ;----------------------------------------
4587                              <1>
4588                              <1> GET_PARM_N:
4589                              <1>       ; ebx = user's buffer address for parameters table
4590                              <1> ;GET_PARM:                       ; GET DRIVE PARAMETERS
4591 00005704 1E                  <1>       PUSH   DS                 ; SAVE REGISTERS
4592 00005705 06                  <1>       PUSH   ES
4593 00005706 53                  <1>       PUSH   eBX
4594                              <1>       ;MOV   AX,ABS0            ; ESTABLISH ADDRESSING
4595                              <1>       ;MOV   DS,AX
4596                              <1>       ;TEST  DL,1               ; CHECK FOR DRIVE 1
4597                              <1>       ;JZ    short G0
4598                              <1>       ;LES   BX,@HF1_TBL_VEC
4599                              <1>       ;JMP   SHORT G1
4600                              <1> ;G0:  LES    BX,@HF_TBL_VEC
4601                              <1> ;G1:
4602                              <1>       ;CALL  DDS                ; ESTABLISH SEGMENT
4603                              <1>       ; 22/12/2014
4604                              <1>       ;push  cs
4605                              <1>       ;pop   ds
4606 00005707 66BB1000            <1>       mov    bx, KDATA
4607 0000570B 8EDB                <1>       mov    ds, bx
4608 0000570D 8EC3                <1>       mov    es, bx ; 27/05/2016
4609                              <1>       ;
4610 0000570F 80EA80              <1>       SUB    DL,80H
4611                              <1>       ;CMP   DL,MAX_FILE        ; TEST WITHIN RANGE
4612                              <1>       ;JAE   short G4 ; 29/08/2020 - BugFix
4613                              <1>       ; 30/08/2020
4614 00005712 3A15[5C8A0100]      <1>       cmp    dl, [HF_NUM] ; is hard disk index < hard disk count ?
4615 00005718 736A                <1>       jae    short G4     ; no, error ! drive not ready !
4616                              <1>       ;
4617 0000571A 31DB                <1>       xor    ebx, ebx ; 21/02/2015
4618                              <1>       ; 22/12/2014
4619 0000571C 88D3                <1>       mov    bl, dl
4620                              <1>       ;xor   bh, bh
4621 0000571E C0E302              <1>       shl    bl, 2              ; convert index to offset
4622                              <1>       ;add   bx, HF_TBL_VEC
4623 00005721 81C3[608A0100]      <1>       add    ebx, HF_TBL_VEC
4624                              <1>       ;mov   ax, [bx+2]
4625                              <1>       ;mov   es, ax             ; dpt segment
4626                              <1>       ;mov   bx, [bx]           ; dpt offset
4627 00005727 8B1B                <1>       mov    ebx, [ebx] ; 32 bit offset
4628                              <1>
4629 00005729 C605[5B8A0100]00    <1>       MOV    byte [DISK_STATUS1],0
4630                              <1>         ;MOV AX,[ES:BX]         ; MAX NUMBER OF CYLINDERS
4631 00005730 668B03              <1>       mov    ax, [ebx]
4632                              <1>       ;;SUB  AX,2               ; ADJUST FOR 0-N
4633 00005733 6648                <1>       dec    ax                 ; max. cylinder number
4634 00005735 88C5                <1>       MOV    CH,AL
4635 00005737 66250003            <1>       AND    AX,0300H           ; HIGH TWO BITS OF CYLINDER
4636 0000573B 66D1E8              <1>       SHR    AX,1
4637 0000573E 66D1E8              <1>       SHR    AX,1
4638                              <1>       ;OR    AL,[ES:BX+14]      ; SECTORS
4639 00005741 0A430E              <1>       or     al, [ebx+14]
4640 00005744 88C1                <1>       MOV    CL,AL
4641                              <1>       ;MOV   DH,[ES:BX+2]       ; HEADS
4642 00005746 8A7302              <1>       mov    dh, [ebx+2]
4643 00005749 FECE                <1>       DEC    DH                 ; 0-N RANGE
4644 0000574B 8A15[5C8A0100]      <1>       MOV    DL,[HF_NUM]        ; DRIVE COUNT
4645 00005751 6629C0              <1>       SUB    AX,AX
4646                              <1>         ;27/12/2014
4647                              <1>       ;mov   di, bx             ; HDPT offset
4648                              <1>
4649                              <1>       ; 29/08/2020
4650 00005754 833C2400            <1>       cmp    dword [esp], 0
4651 00005758 7703                <1>       ja     short G7 ; ebx > 0
4652                              <1>
4653                              <1>       ; if EBX (user's buffer address) = 0, do not copy DPT
4654 0000575A 5B                  <1>       pop    ebx
4655 0000575B EB24                <1>       jmp    short G5
4656                              <1> G7:
4657                              <1>       ; 27/05/2016
4658                              <1>       ; return fixed disk parameters table to user
4659                              <1>       ; in user's buffer, which is pointed by EBX
4660                              <1>       ;
4661 0000575D 873C24              <1>       xchg   edi, [esp]         ; ebx (input)-> edi, edi -> [esp]
4662 00005760 56                  <1>       push   esi
4663 00005761 89DE                <1>       mov    esi, ebx           ; hard disk parameter table (32 bytes)
4664 00005763 89FB                <1>       mov    ebx, edi           ; ebx = user's buffer address
4665 00005765 51                  <1>       push   ecx
4666 00005766 50                  <1>       push   eax
4667 00005767 B920000000          <1>       mov    ecx, 32 ; 32 bytes
4668 0000576C E881C30000          <1>       call   transfer_to_user_buffer ; trdosk6.s (16/05/2016)
4669 00005771 58                  <1>       pop    eax
4670 00005772 59                  <1>       pop    ecx
4671 00005773 5E                  <1>       pop    esi
4672 00005774 5F                  <1>       pop    edi
4673 00005775 730A                <1>       jnc    short G5
4674                              <1>       ; 29/05/2016 (*)
4675 00005777 B8FF000000          <1>       mov    eax, 0FFh ; unknown error !
4676                              <1> G6:
4677 0000577C 804C241001          <1>       or     byte [esp+16], 1 ; set carry bit of eflags register
4678                              <1> G5:
4679                              <1>       ; 27/05/2016
```

```
4680                                    <1>       ;POP   eBX                ; RESTORE REGISTERS
4681 00005781 07                        <1>       POP    ES
4682 00005782 1F                        <1>       POP    DS
4683                                    <1>       ;RETf  2
4684                                    <1>       ; (*) 29/05/2016
4685                                    <1>       ; (*) retf 4
4686                                    <1>       ; (*) or byte [esp+8], 1 ; set carry bit of eflags register
4687 00005783 CF                        <1>       iretd
4688                                    <1> G4:
4689 00005784 C605[5B8A0100]07          <1>       MOV     byte [DISK_STATUS1],INIT_FAIL ; OPERATION FAILED
4690                                    <1>       ;mov   ah, NOT_DRY ; 30/08/2020 - 'drive not ready' error code
4691 0000578B B407                      <1>       MOV    AH,INIT_FAIL
4692 0000578D 28C0                      <1>       SUB    AL,AL
4693                                    <1>       ;SUB   DX,DX
4694                                    <1>       ; 30/08/2020
4695 0000578F 8A15[5C8A0100]            <1>       mov    dl, [HF_NUM] ; disk count
4696 00005795 28F6                      <1>       sub    dh, dh
4697 00005797 6629C9                    <1>       SUB    CX,CX
4698                                    <1>       ; 29/05/2016 (*)
4699                                    <1>       ;STC                      ; SET ERROR FLAG
4700                                    <1>       ;JMP   short G5
4701                                    <1>       ; 29/08/2020 - BugFix
4702 0000579A 5B                        <1>       pop    ebx
4703 0000579B EBDF                      <1>       jmp    short G6
4704                                    <1>
4705                                    <1> ;---------------------------------------
4706                                    <1> ;     INITIALIZE DRIVE     (AH = 09H) :
4707                                    <1> ;---------------------------------------
4708                                    <1>       ; 03/01/2015
4709                                    <1>       ; According to ATA-ATAPI specification v2.0 to v5.0
4710                                    <1>       ; logical sector per logical track
4711                                    <1>       ; and logical heads - 1 would be set but
4712                                    <1>       ; it is seen as it will be good
4713                                    <1>       ; if physical parameters will be set here
4714                                    <1>       ; because, number of heads <= 16.
4715                                    <1>       ; (logical heads usually more than 16)
4716                                    <1>       ; NOTE: ATA logical parameters (software C, H, S)
4717                                    <1>       ;      == INT 13h physical parameters
4718                                    <1>
4719                                    <1> ;INIT_DRV:
4720                                    <1> ;     MOV    byte [CMD_BLOCK+6],SET_PARM_CMD
4721                                    <1> ;     CALL   GET_VEC             ; ES:BX -> PARAMETER BLOCK
4722                                    <1> ;     MOV    AL,[ES:BX+2]        ; GET NUMBER OF HEADS
4723                                    <1> ;     DEC    AL                  ; CONVERT TO 0-INDEX
4724                                    <1> ;     MOV    AH,[CMD_BLOCK+5]    ; GET SDH REGISTER
4725                                    <1> ;     AND    AH,0F0H             ; CHANGE HEAD NUMBER
4726                                    <1> ;     OR     AH,AL               ; TO MAX HEAD
4727                                    <1> ;     MOV    [CMD_BLOCK+5],AH
4728                                    <1> ;     MOV    AL,[ES:BX+14]       ; MAX SECTOR NUMBER
4729                                    <1> ;     MOV    [CMD_BLOCK+1],AL
4730                                    <1> ;     SUB    AX,AX
4731                                    <1> ;     MOV    [CMD_BLOCK+3],AL    ; ZERO FLAGS
4732                                    <1> ;     CALL   COMMAND             ; TELL CONTROLLER
4733                                    <1> ;     JNZ    short INIT_EXIT           ; CONTROLLER BUSY ERROR
4734                                    <1> ;     CALL   NOT_BUSY            ; WAIT FOR IT TO BE DONE
4735                                    <1> ;     JNZ    short INIT_EXIT           ; TIME OUT
4736                                    <1> ;     CALL   CHECK_STATUS
4737                                    <1> ;INIT_EXIT:
4738                                    <1> ;     RETn
4739                                    <1>
4740                                    <1> ; 04/01/2015
4741                                    <1> ; 02/01/2015 - Derived from from AWARD BIOS 1999
4742                                    <1> ;                          AHDSK.ASM - INIT_DRIVE
4743                                    <1> INIT_DRV:
4744                                    <1>       ;xor   ah,ah
4745 0000579D 31C0                      <1>       xor    eax, eax ; 21/02/2015
4746 0000579F B00B                      <1>       mov    al,11 ; Physical heads from translated HDPT
4747 000057A1 3825[708A0100]            <1>         cmp     [LBAMode], ah   ; 0
4748 000057A7 7702                      <1>       ja     short idrv0
4749 000057A9 B002                      <1>       mov    al,2  ; Physical heads from standard HDPT
4750                                    <1> idrv0:
4751                                    <1>       ; DL = drive number (0 based)
4752 000057AB E8C6030000                <1>       call   GET_VEC
4753                                    <1>       ;push  bx
4754 000057B0 53                        <1>       push   ebx ; 21/02/2015
4755                                    <1>       ;add   bx,ax
4756 000057B1 01C3                      <1>       add    ebx, eax
4757                                    <1>       ;; 05/01/2015
4758 000057B3 8A25[F46D0000]            <1>       mov    ah, [hf_m_s] ; drive number (0= master, 1= slave)
4759                                    <1>       ;;and  ah,1
4760 000057B9 C0E404                    <1>       shl    ah,4
4761 000057BC 80CCA0                    <1>       or     ah,0A0h  ; Drive/Head register - 10100000b (A0h)
4762                                    <1>       ;mov   al,[es:bx]
4763 000057BF 8A03                      <1>       mov    al, [ebx] ; 21/02/2015
4764 000057C1 FEC8                      <1>       dec    al      ; last head number
4765                                    <1>       ;and   al,0Fh
4766 000057C3 08E0                      <1>       or     al,ah   ; lower 4 bits for head number
4767                                    <1>       ;
4768 000057C5 C645FE91                  <1>       mov    byte [CMD_BLOCK+6],SET_PARM_CMD
4769 000057C9 8845FD                    <1>       mov    [CMD_BLOCK+5],al
4770                                    <1>       ;pop   bx
4771 000057CC 5B                        <1>       pop    ebx
4772 000057CD 29C0                      <1>       sub    eax, eax ; 21/02/2015
4773 000057CF B004                      <1>       mov    al,4 ; Physical sec per track from translated HDPT
4774 000057D1 803D[708A0100]00          <1>       cmp    byte [LBAMode], 0
4775 000057D8 7702                      <1>       ja     short idrv1
4776 000057DA B00E                      <1>       mov    al,14 ; Physical sec per track from standard HDPT
4777                                    <1> idrv1:
4778                                    <1>       ;xor   ah,ah
4779                                    <1>       ;add   bx,ax
4780 000057DC 01C3                      <1>       add    ebx, eax ; 21/02/2015
4781                                    <1>       ;mov   al,[es:bx]
4782                                    <1>                     ; sector number
4783 000057DE 8A03                      <1>       mov    al, [ebx]
4784 000057E0 8845F9                    <1>       mov    [CMD_BLOCK+1],al
```

```
4785 000057E3 28C0              <1>       sub    al,al
4786 000057E5 8845FB            <1>       mov    [CMD_BLOCK+3],al  ; ZERO FLAGS
4787 000057E8 E8CA010000        <1>       call   COMMAND         ; TELL CONTROLLER
4788 000057ED 750C              <1>       jnz    short INIT_EXIT       ; CONTROLLER BUSY ERROR
4789 000057EF E878020000        <1>       call   NOT_BUSY       ; WAIT FOR IT TO BE DONE
4790 000057F4 7505              <1>       jnz    short INIT_EXIT       ; TIME OUT
4791 000057F6 E8C9020000        <1>       call   CHECK_STATUS
4792                            <1> INIT_EXIT:
4793 000057FB C3                <1>       RETn
4794                            <1>
4795                            <1> ;---------------------------------------
4796                            <1> ;     READ LONG         (AH = 0AH) :
4797                            <1> ;---------------------------------------
4798                            <1>
4799                            <1> RD_LONG:
4800                            <1>       ;MOV   @CMD_BLOCK+6,READ_CMD OR ECC_MODE
4801 000057FC C645FE22          <1>         mov     byte [CMD_BLOCK+6],READ_CMD + ECC_MODE
4802 00005800 E9E0000000        <1>         JMP     COMMANDI
4803                            <1>
4804                            <1> ;---------------------------------------
4805                            <1> ;     WRITE LONG        (AH = 0BH) :
4806                            <1> ;---------------------------------------
4807                            <1>
4808                            <1> WR_LONG:
4809                            <1>       ;MOV   @CMD_BLOCK+6,WRITE_CMD OR ECC_MODE
4810 00005805 C645FE32          <1>         MOV     byte [CMD_BLOCK+6],WRITE_CMD + ECC_MODE
4811 00005809 E932010000        <1>         JMP     COMMANDO
4812                            <1>
4813                            <1> ;---------------------------------------
4814                            <1> ;     SEEK              (AH = 0CH) :
4815                            <1> ;---------------------------------------
4816                            <1>
4817                            <1> DISK_SEEK:
4818 0000580E C645FE70          <1>         MOV     byte [CMD_BLOCK+6],SEEK_CMD
4819 00005812 E8A0010000        <1>       CALL   COMMAND
4820 00005817 751C              <1>       JNZ    short DS_EXIT             ; CONTROLLER BUSY ERROR
4821 00005819 E812020000        <1>       CALL   _WAIT
4822 0000581E 7515              <1>         JNZ     DS_EXIT                ; TIME OUT ON SEEK
4823 00005820 E89F020000        <1>       CALL   CHECK_STATUS
4824 00005825 803D[5B8A0100]40  <1>         CMP     byte [DISK_STATUS1],BAD_SEEK
4825 0000582C 7507              <1>       JNE    short DS_EXIT
4826 0000582E C605[5B8A0100]00  <1>         MOV     byte [DISK_STATUS1],0
4827                            <1> DS_EXIT:
4828 00005835 C3                <1>       RETn
4829                            <1>
4830                            <1> ;---------------------------------------
4831                            <1> ;     TEST DISK READY      (AH = 10H) :
4832                            <1> ;---------------------------------------
4833                            <1>
4834                            <1> TST_RDY:                          ; WAIT FOR CONTROLLER
4835 00005836 E831020000        <1>       CALL   NOT_BUSY
4836 0000583B 751C              <1>       JNZ    short TR_EX
4837 0000583D 8A45FD            <1>       MOV    AL,[CMD_BLOCK+5]    ; SELECT DRIVE
4838 00005840 668B15[F06D0000]  <1>       MOV    DX,[HF_PORT]
4839 00005847 80C206            <1>       add    dl,6
4840 0000584A EE                <1>       OUT    DX,AL
4841 0000584B E88C020000        <1>       CALL   CHECK_ST           ; CHECK STATUS ONLY
4842 00005850 7507              <1>       JNZ    short TR_EX
4843 00005852 C605[5B8A0100]00  <1>       MOV    byte [DISK_STATUS1],0     ; WIPE OUT DATA CORRECTED ERROR
4844                            <1> TR_EX:
4845 00005859 C3                <1>       RETn
4846                            <1>
4847                            <1> ;---------------------------------------
4848                            <1> ;     RECALIBRATE       (AH = 11H) :
4849                            <1> ;---------------------------------------
4850                            <1>
4851                            <1> HDISK_RECAL:
4852 0000585A C645FE10          <1>         MOV     byte [CMD_BLOCK+6],RECAL_CMD ; 10h, 16
4853 0000585E E854010000        <1>       CALL   COMMAND            ; START THE OPERATION
4854 00005863 7523              <1>       JNZ    short RECAL_EXIT    ; ERROR
4855 00005865 E8C6010000        <1>       CALL   _WAIT              ; WAIT FOR COMPLETION
4856 0000586A 7407              <1>       JZ     short RECAL_X             ; TIME OUT ONE OK ?
4857 0000586C E8BF010000        <1>       CALL   _WAIT              ; WAIT FOR COMPLETION LONGER
4858 00005871 7515              <1>       JNZ    short RECAL_EXIT    ; TIME OUT TWO TIMES IS ERROR
4859                            <1> RECAL_X:
4860 00005873 E84C020000        <1>       CALL   CHECK_STATUS
4861 00005878 803D[5B8A0100]40  <1>       CMP    byte [DISK_STATUS1],BAD_SEEK ; SEEK NOT COMPLETE
4862 0000587F 7507              <1>       JNE    short RECAL_EXIT    ; IS OK
4863 00005881 C605[5B8A0100]00  <1>       MOV    byte [DISK_STATUS1],0
4864                            <1> RECAL_EXIT:
4865 00005888 803D[5B8A0100]00  <1>         CMP     byte [DISK_STATUS1],0
4866 0000588F C3                <1>       RETn
4867                            <1>
4868                            <1> ;---------------------------------------
4869                            <1> ;      CONTROLLER DIAGNOSTIC (AH = 14H) :
4870                            <1> ;---------------------------------------
4871                            <1>
4872                            <1> CTLR_DIAGNOSTIC:
4873 00005890 FA                <1>         CLI                             ; DISABLE INTERRUPTS WHILE CHANGING MASK
4874 00005891 E4A1              <1>       IN     AL,INTB01          ; TURN ON SECOND INTERRUPT CHIP
4875                            <1>       ;AND   AL,0BFH
4876 00005893 243F              <1>       and    al, 3Fh                  ; enable IRQ 14 & IRQ 15
4877                            <1>       ;JMP   $+2
4878                            <1>       IODELAY
4878 00005895 EB00              <2>  jmp short $+2
4878 00005897 EB00              <2>  jmp short $+2
4879 00005899 E6A1              <1>       OUT    INTB01,AL
4880                            <1>       IODELAY
4880 0000589B EB00              <2>  jmp short $+2
4880 0000589D EB00              <2>  jmp short $+2
4881 0000589F E421              <1>       IN     AL,INTA01          ; LET INTERRUPTS PASS THRU TO
4882 000058A1 24FB              <1>       AND    AL,0FBH            ;  SECOND CHIP
4883                            <1>       ;JMP   $+2
4884                            <1>       IODELAY
4884 000058A3 EB00              <2>  jmp short $+2
```

```
4884 000058A5 EB00              <2>  jmp short $+2
4885 000058A7 E621              <1>       OUT    INTA01,AL
4886 000058A9 FB                <1>       STI
4887 000058AA E8BD010000        <1>       CALL   NOT_BUSY           ; WAIT FOR CARD
4888 000058AF 752B              <1>       JNZ    short CD_ERR       ; BAD CARD
4889                            <1>       ;MOV   DX, HF_PORT+7
4890 000058B1 668B15[F06D0000]  <1>       mov    dx, [HF_PORT]
4891 000058B8 80C207            <1>       add    dl, 7
4892 000058BB B090              <1>       MOV    AL,DIAG_CMD        ; START DIAGNOSE
4893 000058BD EE                <1>       OUT    DX,AL
4894 000058BE E8A9010000        <1>       CALL   NOT_BUSY           ; WAIT FOR IT TO COMPLETE
4895 000058C3 B480              <1>       MOV    AH,TIME_OUT
4896 000058C5 7517              <1>       JNZ    short CD_EXIT            ; TIME OUT ON DIAGNOSTIC
4897                            <1>       ;MOV   DX,HF_PORT+1       ; GET ERROR REGISTER
4898 000058C7 668B15[F06D0000]  <1>       mov    dx, [HF_PORT]
4899 000058CE FEC2              <1>       inc    dl
4900 000058D0 EC                <1>       IN     AL,DX
4901 000058D1 A2[528A0100]      <1>       MOV    [HF_ERROR],AL      ; SAVE IT
4902 000058D6 B400              <1>       MOV    AH,0
4903 000058D8 3C01              <1>       CMP    AL,1               ; CHECK FOR ALL OK
4904 000058DA 7402              <1>       JE     SHORT CD_EXIT
4905 000058DC B420              <1> CD_ERR: MOV  AH,BAD_CNTLR
4906                            <1> CD_EXIT:
4907 000058DE 8825[5B8A0100]    <1>       MOV    [DISK_STATUS1],AH
4908 000058E4 C3                <1>       RETn
4909                            <1>
4910                            <1> ;----------------------------------------
4911                            <1> ; COMMANDI                        :
4912                            <1> ;     REPEATEDLY INPUTS DATA TILL      :
4913                            <1> ;     NSECTOR RETURNS ZERO             :
4914                            <1> ;----------------------------------------
4915                            <1> COMMANDI:
4916 000058E5 E862020000        <1>       CALL   CHECK_DMA          ; CHECK 64K BOUNDARY ERROR
4917 000058EA 7253              <1>       JC     short CMD_ABORT
4918                            <1>       ;MOV   DI,BX
4919 000058EC 89DF              <1>       mov    edi, ebx ; 21/02/2015
4920 000058EE E8C4000000        <1>       CALL   COMMAND            ; OUTPUT COMMAND
4921 000058F3 754A              <1>       JNZ    short CMD_ABORT
4922                            <1> CMD_I1:
4923 000058F5 E836010000        <1>       CALL   _WAIT              ; WAIT FOR DATA REQUEST INTERRUPT
4924 000058FA 7543              <1>       JNZ    short TM_OUT       ; TIME OUT
4925                            <1> cmd_i1x: ; 18/02/2016
4926                            <1>       ;MOV   CX,256             ; SECTOR SIZE IN WORDS
4927 000058FC B900010000        <1>       mov    ecx, 256 ; 21/02/2015
4928                            <1>       ;MOV   DX,HF_PORT
4929 00005901 668B15[F06D0000]  <1>       mov    dx,[HF_PORT]
4930 00005908 FA                <1>       CLI
4931 00005909 FC                <1>       CLD
4932 0000590A F3666D            <1>       REP    INSW               ; GET THE SECTOR
4933 0000590D FB                <1>       STI
4934 0000590E F645FE02          <1>       TEST   byte [CMD_BLOCK+6],ECC_MODE ; CHECK FOR NORMAL INPUT
4935 00005912 7419              <1>       JZ     short CMD_I3
4936 00005914 E880010000        <1>       CALL   WAIT_DRQ           ; WAIT FOR DATA REQUEST
4937 00005919 7224              <1>       JC     short TM_OUT
4938                            <1>       ;MOV   DX,HF_PORT
4939 0000591B 668B15[F06D0000]  <1>       mov    dx,[HF_PORT]
4940                            <1>       ;MOV   CX,4               ; GET ECC BYTES
4941 00005922 B904000000        <1>       mov    ecx, 4 ; mov cx, 4
4942 00005927 EC                <1> CMD_I2: IN   AL,DX
4943                            <1>       ;MOV   [ES:DI],AL         ; GO SLOW FOR BOARD
4944 00005928 8807              <1>       mov    [edi], al ; 21/02/2015
4945 0000592A 47                <1>       INC    eDI
4946 0000592B E2FA              <1>       LOOP   CMD_I2
4947                            <1> CMD_I3:
4948                            <1>       ; wait for 400 ns
4949 0000592D 80C207            <1>       add    dl, 7
4950 00005930 EC                <1>       in     al, dx
4951 00005931 EC                <1>       in     al, dx
4952 00005932 EC                <1>       in     al, dx
4953                            <1>       ;
4954 00005933 E88C010000        <1>       CALL   CHECK_STATUS
4955 00005938 7505              <1>       JNZ    short CMD_ABORT          ; ERROR RETURNED
4956 0000593A FE4DF9            <1>       DEC    byte [CMD_BLOCK+1]  ; CHECK FOR MORE
4957                            <1>       ;JNZ   SHORT CMD_I1
4958 0000593D 75BD              <1>       jnz    short cmd_i1x ; 18/02/2016
4959                            <1> CMD_ABORT:
4960 0000593F C3                <1> TM_OUT: RETn
4961                            <1>
4962                            <1> ;----------------------------------------
4963                            <1> ; COMMANDO                        :
4964                            <1> ;     REPEATEDLY OUTPUTS DATA TILL     :
4965                            <1> ;     NSECTOR RETURNS ZERO             :
4966                            <1> ;----------------------------------------
4967                            <1> COMMANDO:
4968 00005940 E807020000        <1>       CALL   CHECK_DMA          ; CHECK 64K BOUNDARY ERROR
4969 00005945 72F8              <1>       JC     short CMD_ABORT
4970 00005947 89DE              <1> CMD_OF: MOV  eSI,eBX ; 21/02/2015
4971 00005949 E869000000        <1>       CALL   COMMAND            ; OUTPUT COMMAND
4972 0000594E 75EF              <1>       JNZ    short CMD_ABORT
4973 00005950 E844010000        <1>       CALL   WAIT_DRQ           ; WAIT FOR DATA REQUEST
4974 00005955 72E8              <1>       JC     short TM_OUT              ; TOO LONG
4975                            <1> CMD_O1: ;PUSH      DS
4976                            <1>       ;PUSH  ES                 ; MOVE ES TO DS
4977                            <1>       ;POP   DS
4978                            <1>       ;MOV   CX,256             ; PUT THE DATA OUT TO THE CARD
4979                            <1>       ;MOV   DX,HF_PORT
4980                            <1>       ; 01/02/2015
4981 00005957 668B15[F06D0000]  <1>       mov    dx, [HF_PORT]
4982                            <1>       ;push  es
4983                            <1>       ;pop   ds
4984                            <1>       ;mov   cx, 256
4985 0000595E B900010000        <1>       mov    ecx, 256 ; 21/02/2015
4986 00005963 FA                <1>       CLI
4987 00005964 FC                <1>       CLD
4988 00005965 F3666F            <1>       REP    OUTSW
```

```
4989 00005968 FB                  <1>       STI
4990                              <1>       ;POP   DS                   ; RESTORE DS
4991 00005969 F645FE02            <1>       TEST   byte [CMD_BLOCK+6],ECC_MODE ; CHECK FOR NORMAL OUTPUT
4992 0000596D 7419                <1>       JZ     short CMD_O3
4993 0000596F E825010000          <1>       CALL   WAIT_DRQ            ; WAIT FOR DATA REQUEST
4994 00005974 72C9                <1>       JC     short TM_OUT
4995                              <1>       ;MOV   DX,HF_PORT
4996 00005976 668B15[F06D0000]    <1>       mov    dx, [HF_PORT]
4997                              <1>       ;MOV   CX,4                 ; OUTPUT THE ECC BYTES
4998 0000597D B904000000          <1>       mov    ecx, 4  ; mov cx, 4
4999                              <1> CMD_O2: ;MOV AL,[ES:SI]
5000 00005982 8A06                <1>       mov    al, [esi]
5001 00005984 EE                  <1>       OUT    DX,AL
5002 00005985 46                  <1>       INC    eSI
5003 00005986 E2FA                <1>       LOOP   CMD_O2
5004                              <1> CMD_O3:
5005 00005988 E8A3000000          <1>       CALL   _WAIT                ; WAIT FOR SECTOR COMPLETE INTERRUPT
5006 0000598D 75B0                <1>       JNZ    short TM_OUT         ; ERROR RETURNED
5007 0000598F E830010000          <1>       CALL   CHECK_STATUS
5008 00005994 75A9                <1>       JNZ    short CMD_ABORT
5009 00005996 F605[518A0100]08    <1>       TEST   byte [HF_STATUS],ST_DRQ    ; CHECK FOR MORE
5010 0000599D 75B8                <1>       JNZ    SHORT CMD_O1
5011                              <1>       ;MOV   DX,HF_PORT+2         ; CHECK RESIDUAL SECTOR COUNT
5012 0000599F 668B15[F06D0000]    <1>       mov    dx, [HF_PORT]
5013                              <1>       ;add   dl, 2
5014 000059A6 FEC2                <1>       inc    dl
5015 000059A8 FEC2                <1>       inc    dl
5016 000059AA EC                  <1>       IN     AL,DX                ;
5017 000059AB A8FF                <1>       TEST   AL,0FFH              ;
5018 000059AD 7407                <1>       JZ     short CMD_O4                ; COUNT = 0  OK
5019 000059AF C605[5B8A0100]BB    <1>       MOV    byte [DISK_STATUS1],UNDEF_ERR
5020                              <1>                                   ; OPERATION ABORTED - PARTIAL TRANSFER
5021                              <1> CMD_O4:
5022 000059B6 C3                  <1>       RETn
5023                              <1>
5024                              <1> ;--------------------------------------------------
5025                              <1> ; COMMAND                                        :
5026                              <1> ;     THIS ROUTINE OUTPUTS THE COMMAND BLOCK        :
5027                              <1> ; OUTPUT                                         :
5028                              <1> ;     BL = STATUS                                :
5029                              <1> ;     BH = ERROR REGISTER                        :
5030                              <1> ;--------------------------------------------------
5031                              <1>
5032                              <1> COMMAND:
5033 000059B7 53                  <1>       PUSH   eBX                  ; WAIT FOR SEEK COMPLETE AND READY
5034                              <1>       ;;MOV  CX,DELAY_2           ; SET INITIAL DELAY BEFORE TEST
5035                              <1> COMMAND1:
5036                              <1>       ;;PUSH CX                   ; SAVE LOOP COUNT
5037 000059B8 E879FEFFFF          <1>       CALL   TST_RDY              ; CHECK DRIVE READY
5038                              <1>       ;;POP  CX
5039 000059BD 7419                <1>       JZ     short COMMAND2             ; DRIVE IS READY
5040 000059BF 803D[5B8A0100]80    <1>         CMP     byte [DISK_STATUS1],TIME_OUT ; TST_RDY TIMED OUT--GIVE UP
5041                              <1>       ;JZ    short CMD_TIMEOUT
5042                              <1>       ;;LOOP COMMAND1             ; KEEP TRYING FOR A WHILE
5043                              <1>       ;JMP   SHORT COMMAND4             ; ITS NOT GOING TO GET READY
5044 000059C6 7507                <1>       jne    short COMMAND4
5045                              <1> CMD_TIMEOUT:
5046 000059C8 C605[5B8A0100]20    <1>       MOV    byte [DISK_STATUS1],BAD_CNTLR
5047                              <1> COMMAND4:
5048 000059CF 5B                  <1>       POP    eBX
5049 000059D0 803D[5B8A0100]00    <1>         CMP     byte [DISK_STATUS1],0   ; SET CONDITION CODE FOR CALLER
5050 000059D7 C3                  <1>       RETn
5051                              <1> COMMAND2:
5052 000059D8 5B                  <1>       POP    eBX
5053 000059D9 57                  <1>       PUSH   eDI
5054 000059DA C605[538A0100]00    <1>       MOV    byte [HF_INT_FLAG],0       ; RESET INTERRUPT FLAG
5055 000059E1 FA                  <1>       CLI                         ; INHIBIT INTERRUPTS WHILE CHANGING MASK
5056 000059E2 E4A1                <1>       IN     AL,INTB01            ; TURN ON SECOND INTERRUPT CHIP
5057                              <1>       ;AND   AL,0BFH
5058 000059E4 243F                <1>       and    al, 3Fh                     ; Enable IRQ 14 & 15
5059                              <1>       ;JMP   $+2
5060                              <1>       IODELAY
5060 000059E6 EB00                <2>  jmp short $+2
5060 000059E8 EB00                <2>  jmp short $+2
5061 000059EA E6A1                <1>       OUT    INTB01,AL
5062 000059EC E421                <1>       IN     AL,INTA01           ; LET INTERRUPTS PASS THRU TO
5063 000059EE 24FB                <1>       AND    AL,0FBH             ;  SECOND CHIP
5064                              <1>       ;JMP   $+2
5065                              <1>       IODELAY
5065 000059F0 EB00                <2>  jmp short $+2
5065 000059F2 EB00                <2>  jmp short $+2
5066 000059F4 E621                <1>       OUT    INTA01,AL
5067 000059F6 FB                  <1>       STI
5068 000059F7 31FF                <1>       XOR    eDI,eDI                     ; INDEX THE COMMAND TABLE
5069                              <1>       ;MOV   DX,HF_PORT+1         ; DISK ADDRESS
5070 000059F9 668B15[F06D0000]    <1>       mov    dx, [HF_PORT]
5071 00005A00 FEC2                <1>       inc    dl
5072 00005A02 F605[5D8A0100]C0    <1>       TEST   byte [CONTROL_BYTE],0C0H ; CHECK FOR RETRY SUPPRESSION
5073 00005A09 7411                <1>       JZ     short COMMAND3
5074 00005A0B 8A45FE              <1>       MOV    AL, [CMD_BLOCK+6]   ; YES-GET OPERATION CODE
5075 00005A0E 24F0                <1>       AND    AL,0F0H             ; GET RID OF MODIFIERS
5076 00005A10 3C20                <1>       CMP    AL,20H              ; 20H-40H IS READ, WRITE, VERIFY
5077 00005A12 7208                <1>       JB     short COMMAND3
5078 00005A14 3C40                <1>       CMP    AL,40H
5079 00005A16 7704                <1>       JA     short COMMAND3
5080 00005A18 804DFE01            <1>       OR     byte [CMD_BLOCK+6],NO_RETRIES
5081                              <1>                                   ; VALID OPERATION FOR RETRY SUPPRESS
5082                              <1> COMMAND3:
5083 00005A1C 8A443DF8            <1>       MOV    AL,[CMD_BLOCK+eDI]  ; GET THE COMMAND STRING BYTE
5084 00005A20 EE                  <1>       OUT    DX,AL               ; GIVE IT TO CONTROLLER
5085                              <1>       IODELAY
5085 00005A21 EB00                <2>  jmp short $+2
5085 00005A23 EB00                <2>  jmp short $+2
5086 00005A25 47                  <1>       INC    eDI                 ; NEXT BYTE IN COMMAND BLOCK
5087 00005A26 6642                <1>       INC    DX                  ; NEXT DISK ADAPTER REGISTER
```

```
5088 00005A28 6683FF07          <1>       cmp    di, 7  ; 1/1/2015   ; ALL DONE?
5089 00005A2C 75EE              <1>       JNZ    short COMMAND3            ; NO--GO DO NEXT ONE
5090 00005A2E 5F                <1>       POP    eDI
5091 00005A2F C3                <1>       RETn                       ; ZERO FLAG IS SET
5092                            <1>
5093                            <1> ;CMD_TIMEOUT:
5094                            <1> ;     MOV    byte [DISK_STATUS1],BAD_CNTLR
5095                            <1> ;COMMAND4:
5096                            <1> ;     POP    BX
5097                            <1> ;     CMP    [DISK_STATUS1],0    ; SET CONDITION CODE FOR CALLER
5098                            <1> ;     RETn
5099                            <1>
5100                            <1> ;----------------------------------------
5101                            <1> ;     WAIT FOR INTERRUPT        :
5102                            <1> ;----------------------------------------
5103                            <1> ;WAIT:
5104                            <1> _WAIT:
5105 00005A30 FB                <1>       STI                        ; MAKE SURE INTERRUPTS ARE ON
5106                            <1>       ;SUB   CX,CX               ; SET INITIAL DELAY BEFORE TEST
5107                            <1>       ;CLC
5108                            <1>       ;MOV   AX,9000H            ; DEVICE WAIT INTERRUPT
5109                            <1>       ;INT   15H
5110                            <1>       ;JC    WT2                 ; DEVICE TIMED OUT
5111                            <1>       ;MOV   BL,DELAY_1          ; SET DELAY COUNT
5112                            <1>
5113                            <1>       ;mov   bl, WAIT_HDU_INT_HI
5114                            <1>       ;; 21/02/2015
5115                            <1>       ;;mov  bl, WAIT_HDU_INT_HI + 1
5116                            <1>       ;;mov  cx, WAIT_HDU_INT_LO
5117 00005A31 B915160500        <1>       mov    ecx, WAIT_HDU_INT_LH
5118                            <1>                                  ; (AWARD BIOS -> WAIT_FOR_MEM)
5119                            <1> ;-----      WAIT LOOP
5120                            <1>
5121                            <1> WT1:
5122                            <1>       ;TEST  byte [HF_INT_FLAG],80H    ; TEST FOR INTERRUPT
5123 00005A36 F605[538A0100]C0  <1>       test   byte [HF_INT_FLAG],0C0h
5124                            <1>       ;LOOPZ WT1
5125 00005A3D 7517              <1>       JNZ    short WT3           ; INTERRUPT--LETS GO
5126                            <1>       ;DEC   BL
5127                            <1>       ;JNZ   short WT1           ; KEEP TRYING FOR A WHILE
5128                            <1>
5129                            <1> WT1_hi:
5130 00005A3F E461              <1>       in     al, SYS1 ; 61h (PORT_B)   ; wait for lo to hi
5131 00005A41 A810              <1>       test   al, 10h                   ; transition on memory
5132 00005A43 75FA              <1>       jnz    short WT1_hi        ; refresh.
5133                            <1> WT1_lo:
5134 00005A45 E461              <1>       in     al, SYS1            ; 061h (PORT_B)
5135 00005A47 A810              <1>       test   al, 10h
5136 00005A49 74FA              <1>       jz     short WT1_lo
5137 00005A4B E2E9              <1>       loop   WT1
5138                            <1>       ;;or   bl, bl
5139                            <1>       ;;jz   short WT2
5140                            <1>       ;;dec  bl
5141                            <1>       ;;jmp  short WT1
5142                            <1>       ;dec   bl
5143                            <1>       ;jnz   short WT1
5144                            <1>
5145 00005A4D C605[5B8A0100]80  <1> WT2:  MOV    byte [DISK_STATUS1],TIME_OUT ; REPORT TIME OUT ERROR
5146 00005A54 EB0E              <1>       JMP    SHORT WT4
5147 00005A56 C605[5B8A0100]00  <1> WT3:  MOV    byte [DISK_STATUS1],0
5148 00005A5D C605[538A0100]00  <1>       MOV    byte [HF_INT_FLAG],0
5149 00005A64 803D[5B8A0100]00  <1> WT4:  CMP    byte [DISK_STATUS1],0     ; SET CONDITION CODE FOR CALLER
5150 00005A6B C3                <1>       RETn
5151                            <1>
5152                            <1> ;----------------------------------------
5153                            <1> ;     WAIT FOR CONTROLLER NOT BUSY     :
5154                            <1> ;----------------------------------------
5155                            <1> NOT_BUSY:
5156 00005A6C FB                <1>       STI                        ; MAKE SURE INTERRUPTS ARE ON
5157                            <1>       ;PUSH  eBX
5158                            <1>       ;SUB   CX,CX               ; SET INITIAL DELAY BEFORE TEST
5159 00005A6D 668B15[F06D0000]  <1>       mov    DX, [HF_PORT]
5160 00005A74 80C207            <1>       add    dl, 7               ; Status port (HF_PORT+7)
5161                            <1>       ;MOV   BL,DELAY_1
5162                            <1>                                  ; wait for 10 seconds
5163                            <1>       ;mov   cx, WAIT_HDU_INT_LO ; 1615h
5164                            <1>       ;;mov  bl, WAIT_HDU_INT_HI ;   05h
5165                            <1>       ;mov   bl, WAIT_HDU_INT_HI + 1
5166 00005A77 B915160500        <1>       mov    ecx, WAIT_HDU_INT_LH  ; 21/02/2015
5167                            <1>       ;
5168                            <1> ;;      mov     byte [wait_count], 0    ; Reset wait counter
5169                            <1> NB1:
5170 00005A7C EC                <1>       IN     AL,DX               ; CHECK STATUS
5171                            <1>       ;TEST  AL,ST_BUSY
5172 00005A7D 2480              <1>       and    al, ST_BUSY
5173                            <1>       ;LOOPNZ      NB1
5174 00005A7F 7410              <1>       JZ     short NB2           ; NOT BUSY--LETS GO
5175                            <1>       ;DEC   BL
5176                            <1>       ;JNZ   short NB1           ; KEEP TRYING FOR A WHILE
5177                            <1>
5178 00005A81 E461              <1> NB1_hi: IN   AL,SYS1                   ; wait for hi to lo
5179 00005A83 A810              <1>       TEST   AL,010H                   ; transition on memory
5180 00005A85 75FA              <1>       JNZ    SHORT NB1_hi        ; refresh.
5181 00005A87 E461              <1> NB1_lo: IN   AL,SYS1
5182 00005A89 A810              <1>       TEST   AL,010H
5183 00005A8B 74FA              <1>       JZ     short NB1_lo
5184 00005A8D E2ED              <1>       LOOP   NB1
5185                            <1>       ;dec   bl
5186                            <1>       ;jnz   short NB1
5187                            <1>       ;
5188                            <1> ;;      cmp     byte [wait_count], 182  ; 10 seconds (182 timer ticks)
5189                            <1> ;;    jb     short NB1
5190                            <1>       ;
5191                            <1>       ;MOV   [DISK_STATUS1],TIME_OUT   ; REPORT TIME OUT ERROR
5192                            <1>       ;JMP   SHORT NB3
```

```
5193 00005A8F B080                <1>       mov    al, TIME_OUT
5194                              <1> NB2:
5195                              <1>       ;MOV   byte [DISK_STATUS1],0
5196                              <1> ;NB3:
5197                              <1>       ;POP   eBX
5198 00005A91 A2[5B8A0100]        <1>       mov    [DISK_STATUS1], al  ;;; will be set after return
5199                              <1>       ;CMP   byte [DISK_STATUS1],0     ; SET CONDITION CODE FOR CALLER
5200 00005A96 08C0                <1>       or     al, al              ; (zf = 0 --> timeout)
5201 00005A98 C3                  <1>       RETn
5202                              <1>
5203                              <1> ;----------------------------------------
5204                              <1> ;     WAIT FOR DATA REQUEST            :
5205                              <1> ;----------------------------------------
5206                              <1> WAIT_DRQ:
5207                              <1>       ;MOV   CX,DELAY_3
5208                              <1>       ;MOV   DX,HF_PORT+7
5209 00005A99 668B15[F06D0000]    <1>       mov    dx, [HF_PORT]
5210 00005AA0 80C207              <1>       add    dl, 7
5211                              <1>       ;;MOV  bl, WAIT_HDU_DRQ_HI ; 0
5212                              <1>       ;MOV   cx, WAIT_HDU_DRQ_LO ; 1000 (30 milli seconds)
5213                              <1>                                  ; (but it is written as 2000
5214                              <1>                                  ; micro seconds in ATORGS.ASM file
5215                              <1>                                  ; of Award Bios - 1999, D1A0622)
5216 00005AA3 B9E8030000          <1>       mov    ecx, WAIT_HDU_DRQ_LH ; 21/02/2015
5217 00005AA8 EC                  <1> WQ_1: IN     AL,DX               ; GET STATUS
5218 00005AA9 A808                <1>       TEST   AL,ST_DRQ           ; WAIT FOR DRQ
5219 00005AAB 7516                <1>       JNZ    short WQ_OK
5220                              <1>       ;LOOP  WQ_1                ; KEEP TRYING FOR A SHORT WHILE
5221                              <1> WQ_hi:
5222 00005AAD E461                <1>       IN     AL,SYS1                   ; wait for hi to lo
5223 00005AAF A810                <1>       TEST   AL,010H                   ; transition on memory
5224 00005AB1 75FA                <1>       JNZ    SHORT WQ_hi         ; refresh.
5225 00005AB3 E461                <1> WQ_lo:  IN      AL,SYS1
5226 00005AB5 A810                <1>       TEST   AL,010H
5227 00005AB7 74FA                <1>       JZ     SHORT WQ_lo
5228 00005AB9 E2ED                <1>       LOOP   WQ_1
5229                              <1>
5230 00005ABB C605[5B8A0100]80    <1>         MOV     byte [DISK_STATUS1],TIME_OUT ; ERROR
5231 00005AC2 F9                  <1>       STC
5232                              <1> WQ_OK:
5233 00005AC3 C3                  <1>       RETn
5234                              <1> ;WQ_OK:     ;CLC
5235                              <1> ;     RETn
5236                              <1>
5237                              <1> ;----------------------------------------
5238                              <1> ;     CHECK FIXED DISK STATUS   :
5239                              <1> ;----------------------------------------
5240                              <1> CHECK_STATUS:
5241 00005AC4 E813000000          <1>       CALL   CHECK_ST            ; CHECK THE STATUS BYTE
5242 00005AC9 7509                <1>       JNZ    short CHECK_S1            ; AN ERROR WAS FOUND
5243 00005ACB A801                <1>       TEST   AL,ST_ERROR         ; WERE THERE ANY OTHER ERRORS
5244 00005ACD 7405                <1>       JZ     short CHECK_S1            ; NO ERROR REPORTED
5245 00005ACF E849000000          <1>       CALL   CHECK_ER            ; ERROR REPORTED
5246                              <1> CHECK_S1:
5247 00005AD4 803D[5B8A0100]00    <1>       CMP    byte [DISK_STATUS1],0     ; SET STATUS FOR CALLER
5248 00005ADB C3                  <1>       RETn
5249                              <1>
5250                              <1> ;----------------------------------------
5251                              <1> ;     CHECK FIXED DISK STATUS BYTE     :
5252                              <1> ;----------------------------------------
5253                              <1> CHECK_ST:
5254                              <1>       ;MOV   DX,HF_PORT+7        ; GET THE STATUS
5255 00005ADC 668B15[F06D0000]    <1>       mov    dx, [HF_PORT]
5256 00005AE3 80C207              <1>       add    dl, 7
5257                              <1>
5258                              <1>       ; 17/02/2016
5259                              <1>       ;(http://wiki.osdev.org/ATA_PIO_Mode)
5260                              <1>       ;"delay 400ns to allow drive to set new values of BSY and DRQ"
5261 00005AE6 EC                  <1>       IN     AL,DX
5262                              <1>       ;in    al, dx ; 100ns
5263                              <1>       ;in    al, dx ; 100ns
5264                              <1>       ;in    al, dx ; 100ns
5265                              <1>       NEWIODELAY ; 18/02/2016 (AWARD BIOS - 1999, 'CKST' in AHSDK.ASM)
5265 00005AE7 E6EB                <2>  out 0ebh,al
5266                              <1>       ;
5267 00005AE9 A2[518A0100]        <1>       MOV    [HF_STATUS],AL
5268 00005AEE B400                <1>       MOV    AH,0
5269 00005AF0 A880                <1>       TEST   AL,ST_BUSY          ; IF STILL BUSY
5270 00005AF2 751A                <1>       JNZ    short CKST_EXIT           ;  REPORT OK
5271 00005AF4 B4CC                <1>       MOV    AH,WRITE_FAULT
5272 00005AF6 A820                <1>       TEST   AL,ST_WRT_FLT       ; CHECK FOR WRITE FAULT
5273 00005AF8 7514                <1>       JNZ    short CKST_EXIT
5274 00005AFA B4AA                <1>       MOV    AH,NOT_RDY
5275 00005AFC A840                <1>       TEST   AL,ST_READY         ; CHECK FOR NOT READY
5276 00005AFE 740E                <1>       JZ     short CKST_EXIT
5277 00005B00 B440                <1>       MOV    AH,BAD_SEEK
5278 00005B02 A810                <1>       TEST   AL,ST_SEEK_COMPL    ; CHECK FOR SEEK NOT COMPLETE
5279 00005B04 7408                <1>       JZ     short CKST_EXIT
5280 00005B06 B411                <1>       MOV    AH,DATA_CORRECTED
5281 00005B08 A804                <1>       TEST   AL,ST_CORRCTD       ; CHECK FOR CORRECTED ECC
5282 00005B0A 7502                <1>       JNZ    short CKST_EXIT
5283 00005B0C B400                <1>       MOV    AH,0
5284                              <1> CKST_EXIT:
5285 00005B0E 8825[5B8A0100]      <1>       MOV    [DISK_STATUS1],AH   ; SET ERROR FLAG
5286 00005B14 80FC11              <1>       CMP    AH,DATA_CORRECTED   ; KEEP GOING WITH DATA CORRECTED
5287 00005B17 7403                <1>       JZ     short CKST_EX1
5288 00005B19 80FC00              <1>       CMP    AH,0
5289                              <1> CKST_EX1:
5290 00005B1C C3                  <1>       RETn
5291                              <1>
5292                              <1> ;----------------------------------------
5293                              <1> ;     CHECK FIXED DISK ERROR REGISTER :
5294                              <1> ;----------------------------------------
5295                              <1> CHECK_ER:
5296                              <1>       ;MOV   DX, HF_PORT+1       ; GET THE ERROR REGISTER
```

```
5297 00005B1D 668B15[F06D0000]    <1>       mov    dx, [HF_PORT]       ;
5298 00005B24 FEC2                <1>       inc    dl
5299 00005B26 EC                  <1>       IN     AL,DX
5300 00005B27 A2[528A0100]        <1>       MOV    [HF_ERROR],AL
5301 00005B2C 53                  <1>       PUSH   eBX ; 21/02/2015
5302 00005B2D B908000000          <1>       MOV    eCX,8               ; TEST ALL 8 BITS
5303 00005B32 D0E0                <1> CK1:  SHL    AL,1                ; MOVE NEXT ERROR BIT TO CARRY
5304 00005B34 7202                <1>       JC     short CK2           ; FOUND THE ERROR
5305 00005B36 E2FA                <1>       LOOP   CK1                 ; KEEP TRYING
5306 00005B38 BB[E46D0000]        <1> CK2:  MOV    eBX, ERR_TBL        ; COMPUTE ADDRESS OF
5307 00005B3D 01CB                <1>       ADD    eBX,eCX                   ; ERROR CODE
5308                              <1>       ;;MOV  AH,BYTE [CS:BX]           ; GET ERROR CODE
5309                              <1>       ;mov   ah, [bx]
5310 00005B3F 8A23                <1>       mov    ah, [ebx] ; 21/02/2015
5311 00005B41 8825[5B8A0100]      <1> CKEX: MOV    [DISK_STATUS1],AH   ; SAVE ERROR CODE
5312 00005B47 5B                  <1>       POP    eBX
5313 00005B48 80FC00              <1>       CMP    AH,0
5314 00005B4B C3                  <1>       RETn
5315                              <1>
5316                              <1> ;--------------------------------------------------------
5317                              <1> ; CHECK_DMA                                    :
5318                              <1> ;  -CHECK ES:BX AND # SECTORS TO MAKE SURE THAT IT WILL :
5319                              <1> ;   FIT WITHOUT SEGMENT OVERFLOW.                    :
5320                              <1> ;  -ES:BX HAS BEEN REVISED TO THE FORMAT SSSS:000X  :
5321                              <1> ;  -OK IF # SECTORS < 80H (7FH IF LONG READ OR WRITE)     :
5322                              <1> ;  -OK IF # SECTORS = 80H (7FH) AND BX <= 00H (04H) :
5323                              <1> ;  -ERROR OTHERWISE                                  :
5324                              <1> ;--------------------------------------------------------
5325                              <1> CHECK_DMA:
5326 00005B4C 6650                <1>       PUSH   AX                  ; SAVE REGISTERS
5327 00005B4E 66B80080            <1>       MOV    AX,8000H            ; AH = MAX # SECTORS AL = MAX OFFSET
5328 00005B52 F645FE02            <1>         TEST    byte [CMD_BLOCK+6],ECC_MODE
5329 00005B56 7404                <1>       JZ     short CKD1
5330 00005B58 66B8047F            <1>       MOV    AX,7F04H            ; ECC IS 4 MORE BYTES
5331 00005B5C 3A65F9              <1> CKD1: CMP    AH, [CMD_BLOCK+1]   ; NUMBER OF SECTORS
5332 00005B5F 7706                <1>       JA     short CKDOK         ; IT WILL FIT
5333 00005B61 7208                <1>       JB     short CKDERR        ; TOO MANY
5334 00005B63 38D8                <1>       CMP    AL,BL               ; CHECK OFFSET ON MAX SECTORS
5335 00005B65 7204                <1>       JB     short CKDERR        ; ERROR
5336 00005B67 F8                  <1> CKDOK:       CLC                       ; CLEAR CARRY
5337 00005B68 6658                <1>       POP    AX
5338 00005B6A C3                  <1>       RETn                       ; NORMAL RETURN
5339 00005B6B F9                  <1> CKDERR: STC                      ; INDICATE ERROR
5340 00005B6C C605[5B8A0100]09    <1>         MOV     byte [DISK_STATUS1],DMA_BOUNDARY
5341 00005B73 6658                <1>       POP    AX
5342 00005B75 C3                  <1>       RETn
5343                              <1>
5344                              <1> ;----------------------------------------
5345                              <1> ;     SET UP ES:BX-> DISK PARMS :
5346                              <1> ;----------------------------------------
5347                              <1>
5348                              <1> ; INPUT -> DL = 0 based drive number
5349                              <1> ; OUTPUT -> ES:BX = disk parameter table address
5350                              <1>
5351                              <1> GET_VEC:
5352                              <1>       ;SUB   AX,AX               ; GET DISK PARAMETER ADDRESS
5353                              <1>       ;MOV   ES,AX
5354                              <1>       ;TEST  DL,1
5355                              <1>       ;JZ    short GV_0
5356                              <1> ;     LES    BX,[HF1_TBL_VEC]    ; ES:BX -> DRIVE PARAMETERS
5357                              <1> ;     JMP    SHORT GV_EXIT
5358                              <1> ;GV_0:
5359                              <1> ;     LES    BX,[HF_TBL_VEC]           ; ES:BX -> DRIVE PARAMETERS
5360                              <1> ;
5361                              <1>       ;xor   bh, bh
5362 00005B76 31DB                <1>       xor    ebx, ebx
5363 00005B78 88D3                <1>       mov    bl, dl
5364                              <1>       ;;02/01/2015
5365                              <1>       ;;shl  bl, 1               ; port address offset
5366                              <1>       ;;mov  ax, [bx+hd_ports]   ; Base port address (1F0h, 170h)
5367                              <1>       ;;shl  bl, 1               ; dpt pointer offset
5368 00005B7A C0E302              <1>       shl    bl, 2  ;;
5369                              <1>       ;add   bx, HF_TBL_VEC            ; Disk parameter table pointer
5370 00005B7D 81C3[608A0100]      <1>       add    ebx, HF_TBL_VEC ; 21/02/2015
5371                              <1>       ;push  word [bx+2]         ; dpt segment
5372                              <1>       ;pop   es
5373                              <1>       ;mov   bx, [bx]            ; dpt offset
5374 00005B83 8B1B                <1>       mov    ebx, [ebx]
5375                              <1> ;GV_EXIT:
5376 00005B85 C3                  <1>       RETn
5377                              <1>
5378                              <1> hdc1_int: ; 21/02/2015
5379                              <1> ;--- HARDWARE INT 76H -- ( IRQ LEVEL  14 ) ----------------------
5380                              <1> ;                                                    :
5381                              <1> ;     FIXED DISK INTERRUPT ROUTINE                         :
5382                              <1> ;                                                    :
5383                              <1> ;--------------------------------------------------------------
5384                              <1>
5385                              <1> ; 22/12/2014
5386                              <1> ; IBM PC-XT Model 286 System BIOS Source Code - DISK.ASM (HD_INT)
5387                              <1> ;      '11/15/85'
5388                              <1> ; AWARD BIOS 1999 (D1A0622)
5389                              <1> ;     Source Code - ATORGS.ASM (INT_HDISK, INT_HDISK1)
5390                              <1>
5391                              <1> ;int_76h:
5392                              <1> HD_INT:
5393 00005B86 6650                <1>       PUSH   AX
5394 00005B88 1E                  <1>       PUSH   DS
5395                              <1>       ;CALL  DDS
5396                              <1>       ; 21/02/2015 (32 bit, 386 pm modification)
5397 00005B89 66B81000            <1>       mov    ax, KDATA
5398 00005B8D 8ED8                <1>       mov    ds, ax
5399                              <1>       ;
5400                              <1>       ;;MOV  @HF_INT_FLAG,0FFH   ; ALL DONE
5401                              <1>         ;mov     byte [CS:HF_INT_FLAG], 0FFh
```

```
5402 00005B8F C605[538A0100]FF    <1>       mov    byte [HF_INT_FLAG], 0FFh
5403                              <1>       ;
5404 00005B96 6652                <1>       push   dx
5405 00005B98 66BAF701            <1>       mov    dx, HDC1_BASEPORT+7 ; Status Register (1F7h)
5406                              <1>                                  ; Clear Controller
5407                              <1> Clear_IRQ1415:                         ; (Award BIOS - 1999)
5408 00005B9C EC                  <1>       in     al, dx              ;
5409 00005B9D 665A                <1>       pop    dx
5410                              <1>       NEWIODELAY
5410 00005B9F E6EB                <2>  out 0ebh,al
5411                              <1>       ;
5412 00005BA1 B020                <1>       MOV    AL,EOI              ; NON-SPECIFIC END OF INTERRUPT
5413 00005BA3 E6A0                <1>       OUT    INTB00,AL           ; FOR CONTROLLER #2
5414                              <1>       ;JMP   $+2                 ; WAIT
5415                              <1>       NEWIODELAY
5415 00005BA5 E6EB                <2>  out 0ebh,al
5416 00005BA7 E620                <1>       OUT    INTA00,AL           ; FOR CONTROLLER #1
5417 00005BA9 1F                  <1>       POP    DS
5418                              <1>       ;STI                       ; RE-ENABLE INTERRUPTS
5419                              <1>       ;MOV   AX,9100H            ; DEVICE POST
5420                              <1>       ;INT   15H                 ;  INTERRUPT
5421                              <1> irq15_iret: ; 25/02/2015
5422 00005BAA 6658                <1>       POP    AX
5423 00005BAC CF                  <1>       IRETd                      ; RETURN FROM INTERRUPT
5424                              <1>
5425                              <1> hdc2_int: ; 21/02/2015
5426                              <1> ;++++ HARDWARE INT 77H ++ ( IRQ LEVEL  15 ) ++++++++++++++++++++
5427                              <1> ;                                                    :
5428                              <1> ;     FIXED DISK INTERRUPT ROUTINE                         :
5429                              <1> ;                                                    :
5430                              <1> ;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
5431                              <1>
5432                              <1> ;int 77h:
5433                              <1> HD1_INT:
5434 00005BAD 6650                <1>       PUSH   AX
5435                              <1>       ; Check if that is a spurious IRQ (from slave PIC)
5436                              <1>       ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
5437 00005BAF B00B                <1>       mov    al, 0Bh  ; In-Service Register
5438 00005BB1 E6A0                <1>       out    0A0h, al
5439 00005BB3 EB00                <1>         jmp short $+2
5440 00005BB5 EB00                <1>       jmp short $+2
5441 00005BB7 E4A0                <1>       in     al, 0A0h
5442 00005BB9 2480                <1>       and    al, 80h ; bit 7 (is it real IRQ 15 or fake?)
5443 00005BBB 74ED                <1>       jz     short irq15_iret ; Fake (spurious)IRQ, do not send EOI)
5444                              <1>       ;
5445 00005BBD 1E                  <1>       PUSH   DS
5446                              <1>       ;CALL  DDS
5447                              <1>       ; 21/02/2015 (32 bit, 386 pm modification)
5448 00005BBE 66B81000            <1>       mov    ax, KDATA
5449 00005BC2 8ED8                <1>       mov    ds, ax
5450                              <1>       ;
5451                              <1>       ;;MOV  @HF_INT_FLAG,0FFH   ; ALL DONE
5452                              <1>         ;or      byte [CS:HF_INT_FLAG],0C0h
5453 00005BC4 800D[538A0100]C0    <1>       or     byte [HF_INT_FLAG], 0C0h
5454                              <1>       ;
5455 00005BCB 6652                <1>       push   dx
5456 00005BCD 66BA7701            <1>       mov    dx, HDC2_BASEPORT+7 ; Status Register (177h)
5457                              <1>                                  ; Clear Controller (Award BIOS 1999)
5458 00005BD1 EBC9                <1>       jmp    short Clear_IRQ1415
5459                              <1>
5460                              <1>
5461                              <1> ;%include 'diskdata.inc' ; 11/03/2015
5462                              <1> ;%include 'diskbss.inc' ; 11/03/2015
5463                              <1>
5464                              <1>
5465                              <1> ;////////////////////////////////////////////////////////////////
5466                              <1> ;; END OF DISK I/O SYTEM ///
2887                                  %include 'memory.s'  ; 09/03/2015
   1                              <1> ; ****************************************************************************
   2                              <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.3 - memory.s
   3                              <1> ; ----------------------------------------------------------------------------
   4                              <1> ; Last Update: 15/12/2020
   5                              <1> ; ----------------------------------------------------------------------------
   6                              <1> ; Beginning: 24/01/2016
   7                              <1> ; ----------------------------------------------------------------------------
   8                              <1> ; Assembler: NASM version 2.15 (trdos386.s)
   9                              <1> ; ----------------------------------------------------------------------------
  10                              <1> ; Turkish Rational DOS
  11                              <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
  12                              <1> ;
  13                              <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
  14                              <1> ; memory.inc (18/10/2015)
  15                              <1> ; ****************************************************************************
  16                              <1>
  17                              <1> ; MEMORY.ASM - Retro UNIX 386 v1 MEMORY MANAGEMENT FUNCTIONS (PROCEDURES)
  18                              <1> ; Retro UNIX 386 v1 Kernel (unix386.s, v0.2.0.14) - MEMORY.INC
  19                              <1> ; Last Modification: 18/10/2015
  20                              <1>
  21                              <1> ; ///////// MEMORY MANAGEMENT FUNCTIONS (PROCEDURES) ///////////////
  22                              <1>
  23                              <1> ;;04/11/2014 (unix386.s)
  24                              <1> ;PDE_A_PRESENT     equ    1            ; Present flag for PDE
  25                              <1> ;PDE_A_WRITE equ    2            ; Writable (write permission) flag
  26                              <1> ;PDE_A_USER  equ    4            ; User (non-system/kernel) page flag
  27                              <1> ;;
  28                              <1> ;PTE_A_PRESENT     equ    1            ; Present flag for PTE (bit 0)
  29                              <1> ;PTE_A_WRITE equ    2            ; Writable (write permission) flag (bit 1)
  30                              <1> ;PTE_A_USER  equ    4            ; User (non-system/kernel) page flag (bit 2)
  31                              <1> ;PTE_A_ACCESS   equ       32           ; Accessed flag (bit 5) ; 09/03/2015
  32                              <1>
  33                              <1> ; 27/04/2015
  34                              <1> ; 09/03/2015
  35                              <1> PAGE_SIZE    equ 4096           ; page size in bytes
  36                              <1> PAGE_SHIFT   equ 12             ; page table shift count
  37                              <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> ; 14/12/2020
<1> ; (Linear Frame Buffer - video memory mark : AVL bit 1, outside M.A.T.)
<1> PDE_EXTERNAL equ 400h     ; Page directory entry for external memory blocks
<1> PTE_EXTERNAL equ 400h     ; Allocated kernel pages for Linear Frame Buffer
<1>                           ; (Out of memory allocation table)
<1>
<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)
```

```
143                     <1> ;;      procedures will be called as responce to 'out of memory'
144                     <1> ;;      error except errors caused by attribute conflicts.
145                     <1> ;;     (swapper functions)
146                     <1> ;;
147                     <1> ;;    At second,
148                     <1> ;;    - page directory entry will be updated then page table
149                     <1> ;;      entry will be updated.
150                     <1> ;;
151                     <1> ;; MEMORY ALLOCATION TABLE FORMAT:
152                     <1> ;;    - M.A.T. has a size according to available memory as
153                     <1> ;;      follows:
154                     <1> ;;            - 1 (allocation) bit per 1 page (4096 bytes)
155                     <1> ;;            - a bit with value of 0 means allocated page
156                     <1> ;;            - a bit with value of 1 means a free page
157                     <1> ;,    - 'free_pages' pointer holds count of free pages
158                     <1> ;;      depending on M.A.T.
159                     <1> ;;          (NOTE: Free page count will not be checked
160                     <1> ;;          again -on M.A.T.- after initialization.
161                     <1> ;;          Kernel will trust on initial count.)
162                     <1> ;,    - 'free_pages' count will be decreased by allocation
163                     <1> ;;      and it will be increased by deallocation procedures.
164                     <1> ;;
165                     <1> ;;    - Available memory will be calculated during
166                     <1> ;;      the kernel's initialization stage (in real mode).
167                     <1> ;;      Memory allocation table and kernel page tables
168                     <1> ;;      will be formatted/sized as result of available
169                     <1> ;;      memory calculation before paging is enabled.
170                     <1> ;;
171                     <1> ;; For 4GB Available/Present Memory: (max. possible memory size)
172                     <1> ;;    - Memory Allocation Table size will be 128 KB.
173                     <1> ;;    - Memory allocation for kernel page directory size
174                     <1> ;;      is always 4 KB. (in addition to total allocation size
175                     <1> ;;      for page tables)
176                     <1> ;;    - Memory allocation for kernel page tables (1024 tables)
177                     <1> ;;      is 4 MB (1024*4*1024 bytes).
178                     <1> ;;    - User (available) space will be started
179                     <1> ;;      at 6th MB of the memory (after 1MB+4MB).
180                     <1> ;;    - The first 640 KB is for kernel's itself plus
181                     <1> ;;      memory allocation table and kernel's page directory
182                     <1> ;;      (D0000h-EFFFFh may be used as kernel space...)
183                     <1> ;;    - B0000h to B7FFFh address space (32 KB) will be used
184                     <1> ;;      for buffers.
185                     <1> ;;    - ROMBIOS, VIDEO BUFFER and VIDEO ROM space are reserved.
186                     <1> ;,      (A0000h-AFFFFh, C0000h-CFFFFh, F0000h-FFFFFh)
187                     <1> ;;    - Kernel page tables start at 100000h (2nd MB)
188                     <1> ;;
189                     <1> ;; For 1GB Available Memory:
190                     <1> ;;    - Memory Allocation Table size will be 32 KB.
191                     <1> ;;    - Memory allocation for kernel page directory size
192                     <1> ;;      is always 4 KB. (in addition to total allocation size
193                     <1> ;;      for page tables)
194                     <1> ;;    - Memory allocation for kernel page tables (256 tables)
195                     <1> ;;      is 1 MB (256*4*1024 bytes).
196                     <1> ;;    - User (available) space will be started
197                     <1> ;;      at 3th MB of the memory (after 1MB+1MB).
198                     <1> ;;    - The first 640 KB is for kernel's itself plus
199                     <1> ;;      memory allocation table and kernel's page directory
200                     <1> ;;      (D0000h-EFFFFh may be used as kernel space...)
201                     <1> ;;    - B0000h to B7FFFh address space (32 KB) will be used
202                     <1> ;;      for buffers.
203                     <1> ;;    - ROMBIOS, VIDEO BUFFER and VIDEO ROM space are reserved.
204                     <1> ;,      (A0000h-AFFFFh, C0000h-CFFFFh, F0000h-FFFFFh)
205                     <1> ;;    - Kernel page tables start at 100000h (2nd MB).
206                     <1> ;;
207                     <1> ;;
208                     <1>
209                     <1>
210                     <1> ;;*****************************************************************************
211                     <1> ;;
212                     <1> ;; RETRO UNIX 386 v1 - Paging (Method for Copy On Write paging principle)
213                     <1> ;; DEMAND PAGING - PARENT&CHILD PAGE TABLE DUPLICATION PRINCIPLES (23/04/2015)
214                     <1>
215                     <1> ;; Main factor: "sys fork" system call
216                     <1> ;;
217                     <1> ;;          FORK
218                     <1> ;;                      |----> parent - duplicated PTEs, read only pages
219                     <1> ;;  writable pages ---->|
220                     <1> ;;                      |----> child - duplicated PTEs, read only pages
221                     <1> ;;
222                     <1> ;; AVL bit (0) of Page Table Entry is used as duplication sign
223                     <1> ;;
224                     <1> ;; AVL Bit 0 [PTE Bit 9] = 'Duplicated PTE belongs to child' sign/flag (if it is set)
225                     <1> ;; Note: Dirty bit (PTE bit 6) may be used instead of AVL bit 0 (PTE bit 9)
226                     <1> ;;       -while R/W bit is 0-.
227                     <1> ;;
228                     <1> ;; Duplicate page tables with writable pages (the 1st sys fork in the process):
229                     <1> ;; # Parent's Page Table Entries are updated to point same pages as read only,
230                     <1> ;;   as duplicated PTE bit  -AVL bit 0, PTE bit 9- are reset/clear.
231                     <1> ;; # Then Parent's Page Table is copied to Child's Page Table.
232                     <1> ;; # Child's Page Table Entries are updated as duplicated child bit
233                     <1> ;;   -AVL bit 0, PTE bit 9- is set.
234                     <1> ;;
235                     <1> ;; Duplicate page tables with read only pages (several sys fork system calls):
236                     <1> ;; # Parent's read only pages are copied to new child pages.
237                     <1> ;;   Parent's PTE attributes are not changed.
238                     <1> ;;   (Because, there is another parent-child fork before this fork! We must not
239                     <1> ;;    destroy/mix previous fork result).
240                     <1> ;; # Child's Page Table Entries (which are corresponding to Parent's
241                     <1> ;;   read only pages) are set as writable (while duplicated PTE bit is clear).
242                     <1> ;; # Parent's PTEs with writable page attribute are updated to point same pages
243                     <1> ;;   as read only, (while) duplicated PTE bit is reset (clear).
244                     <1> ;; # Parent's Page Table Entries (with writable page attribute) are duplicated
245                     <1> ;;   as Child's Page Table Entries without copying actual page.
246                     <1> ;; # Child 's Page Table Entries (which are corresponding to Parent's writable
247                     <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>       ;
00005BD3 A1[C8890100]   <1>       mov   eax, [free_pages]
00005BD8 21C0           <1>       and   eax, eax
00005BDA 7438           <1>       jz    short out_of_memory
                        <1>       ;
00005BDC 53             <1>       push  ebx
00005BDD 51             <1>       push  ecx
                        <1>       ;
00005BDE BB00001000     <1>       mov   ebx, MEM_ALLOC_TBL   ; Memory Allocation Table offset
00005BE3 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)
00005BE5 031D[CC890100] <1>       add   ebx, [next_page] ; Free page searching starts from here
                        <1>                              ; next_free_page >> 5
```

```
00005BEB 030D[D0890100]      <1>       add    ecx, [last_page] ; Free page searching ends here
                             <1>                               ; (total_pages - 1) >> 5
                             <1> al_p_scan:
00005BF1 39CB                <1>       cmp    ebx, ecx
00005BF3 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>       ;
00005BF5 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.
00005BF8 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!)
00005BFA 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.
00005BFD EBF2                <1>         jmp     short al_p_scan
                             <1>       ;
                             <1> al_p_notfound:
00005BFF 81E900001000        <1>       sub    ecx, MEM_ALLOC_TBL
00005C05 890D[CC890100]      <1>       mov    [next_page], ecx ; next/first free page = last page
                             <1>                               ; (deallocate_page procedure will change it)
00005C0B 31C0                <1>       xor    eax, eax
00005C0D A3[C8890100]        <1>       mov    [free_pages], eax ; 0
00005C12 59                  <1>       pop    ecx
00005C13 5B                  <1>       pop    ebx
                             <1>       ;
                             <1> out_of_memory:
00005C14 E85B040000          <1>       call   swap_out
00005C19 7325                <1>       jnc    short al_p_ok  ; [free_pages] = 0, re-allocation by swap_out
                             <1>       ;
00005C1B 29C0                <1>       sub    eax, eax ; 0
00005C1D F9                  <1>       stc
00005C1E C3                  <1>       retn
                             <1>
                             <1> al_p_found:
00005C1F 89D9                <1>       mov    ecx, ebx
00005C21 81E900001000        <1>       sub    ecx, MEM_ALLOC_TBL
00005C27 890D[CC890100]      <1>       mov    [next_page], ecx ; Set first free page searching start
                             <1>                               ; address/offset (to the next)
00005C2D FF0D[C8890100]      <1>         dec     dword [free_pages] ; 1 page has been allocated (X = X-1)
                             <1>       ;
00005C33 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)
00005C36 C1E103              <1>       shl    ecx, 3        ; (page block offset * 32) + page index
00005C39 01C8                <1>       add    eax, ecx      ; = page number
00005C3B 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...
00005C3E 59                  <1>       pop    ecx
00005C3F 5B                  <1>       pop    ebx
                             <1> al_p_ok:
00005C40 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>       ;
00005C41 E88DFFFFFF          <1>       call   allocate_page
00005C46 7216                <1>       jc     short mkpd_error
                             <1>       ;
00005C48 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>     ;
00005C4D 57                     <1>     push   edi
00005C4E 51                     <1>     push   ecx
00005C4F 50                     <1>     push   eax
00005C50 B900040000             <1>     mov    ecx, PAGE_SIZE / 4
00005C55 89C7                   <1>     mov    edi, eax
00005C57 31C0                   <1>     xor    eax, eax
00005C59 F3AB                   <1>     rep    stosd
00005C5B 58                     <1>     pop    eax
00005C5C 59                     <1>     pop    ecx
00005C5D 5F                     <1>     pop    edi
                                <1> mkpd_error:
                                <1> mkpt_error:
00005C5E 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>     ;
00005C5F E86FFFFFFF             <1>     call   allocate_page
00005C64 72F8                   <1>     jc     short mkpt_error
00005C66 E811000000             <1>     call   set_pde
00005C6B 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>     ;
00005C6D E861FFFFFF             <1>     call   allocate_page
00005C72 7207                   <1>     jc     short mkp_err
00005C74 E821000000             <1>     call   set_pte
00005C79 73D2                   <1>     jnc    short clear_page ; 18/04/2015
                                <1> mkp_err:
00005C7B C3                     <1>     retn
                                <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>     ;
00005C7C 89DA                   <1>     mov    edx, ebx
00005C7E C1EA16                 <1>     shr    edx, PAGE_D_SHIFT ; 22
00005C81 C1E202                 <1>     shl    edx, 2 ; offset to page directory (1024*4)
00005C84 0315[B8030300]         <1>     add    edx, [u.pgdir]
                                <1>     ;
```

```
00005C8A 21C0                <1>       and    eax, eax
00005C8C 7506                <1>       jnz    short spde_1
                             <1>       ;
00005C8E 8B02                <1>       mov    eax, [edx]  ; old PDE value
                             <1>       ;test  al, 1
                             <1>       ;jz    short spde_2
00005C90 662500F0            <1>       and    ax, PDE_A_CLEAR ; 0F000h  ; clear lower 12 bits
                             <1> spde_1:
                             <1>       ;and   cx, 0FFFh
00005C94 8902                <1>       mov    [edx], eax
00005C96 66090A              <1>       or     [edx], cx
00005C99 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>       ;
00005C9A 50                  <1>       push   eax
00005C9B A1[B8030300]        <1>       mov    eax, [u.pgdir] ; 20/07/2015
00005CA0 E837000000          <1>       call   get_pde
                             <1>              ; EDX = PDE address
                             <1>              ; EAX = PDE value
00005CA5 5A                  <1>       pop    edx ; physical page address
00005CA6 722A                <1>       jc     short spte_err ; PDE not present
                             <1>       ;
00005CA8 53                  <1>       push   ebx ; 24/07/2015
00005CA9 662500F0            <1>       and    ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 bits
                             <1>                       ; EDX = PT address (physical)
00005CAD C1EB0C              <1>       shr    ebx, PAGE_SHIFT ; 12
00005CB0 81E3FF030000        <1>       and    ebx, PTE_MASK; 03FFh
                             <1>                      ; clear higher 10 bits (PD bits)
00005CB6 C1E302              <1>       shl    ebx, 2   ; offset to page table (1024*4)
00005CB9 01C3                <1>       add    ebx, eax
                             <1>       ;
00005CBB 8B03                <1>       mov    eax, [ebx] ; Old PTE value
00005CBD A801                <1>       test   al, 1
00005CBF 740C                <1>       jz     short spte_0
00005CC1 09D2                <1>       or     edx, edx
00005CC3 750F                <1>       jnz    short spte_1
00005CC5 662500F0            <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 bits
00005CC9 89C2                <1>       mov    edx, eax
00005CCB 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!
00005CCD 21C0                <1>       and    eax, eax
00005CCF 7403                <1>       jz     short spte_1
                             <1>       ; 24/07/2015
                             <1>       ; swapped page ! (on disk)
00005CD1 5B                  <1>       pop    ebx
                             <1> spte_err:
00005CD2 F9                  <1>       stc
00005CD3 C3                  <1>       retn
                             <1> spte_1:
00005CD4 89D0                <1>       mov    eax, edx
                             <1> spte_2:
00005CD6 09CA                <1>       or     edx, ecx
                             <1>       ; 23/06/2015
00005CD8 8913                <1>       mov    [ebx], edx ; PTE value in EDX
                             <1>       ; 24/07/2015
00005CDA 5B                  <1>       pop    ebx
00005CDB 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>       ;
00005CDC 89DA                <1>       mov    edx, ebx
00005CDE C1EA16              <1>       shr    edx, PAGE_D_SHIFT ; 22  (12+10)
```

```
00005CE1 C1E202             <1>       shl    edx, 2 ; offset to page directory (1024*4)
00005CE4 01C2               <1>       add    edx, eax ; page directory address (physical)
00005CE6 8B02               <1>       mov    eax, [edx]
00005CE8 A801               <1>       test   al, PDE_A_PRESENT ; page table is present or not !
00005CEA 751F               <1>       jnz    short gpte_retn
00005CEC F9                 <1>       stc
                            <1> gpde_retn:
00005CED 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>       ;
00005CEE E8E9FFFFFF         <1>       call   get_pde
00005CF3 72F8               <1>       jc     short gpde_retn     ; page table is not present
                            <1>       ;jnc   short gpte_1
                            <1>       ;retn
                            <1> ;gpte_1:
00005CF5 662500F0           <1>       and    ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 bits
00005CF9 89DA               <1>       mov    edx, ebx
00005CFB C1EA0C             <1>       shr    edx, PAGE_SHIFT ; 12
00005CFE 81E2FF030000       <1>       and    edx, PTE_MASK; 03FFh
                            <1>                     ; clear higher 10 bits (PD bits)
00005D04 C1E202             <1>       shl    edx, 2 ; offset from start of page table (1024*4)
00005D07 01C2               <1>       add    edx, eax
00005D09 8B02               <1>       mov    eax, [edx]
                            <1> gpte_retn:
00005D0B 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>       ;
00005D0C 56                 <1>       push   esi
00005D0D 51                 <1>       push   ecx
00005D0E 50                 <1>       push   eax
00005D0F 89C6               <1>       mov    esi, eax
00005D11 31C9               <1>       xor    ecx, ecx
                            <1>       ; The 1st PDE points to Kernel Page Table 0 (the 1st 4MB),
                            <1>       ; it must not be deallocated
00005D13 890E               <1>       mov    [esi], ecx ; 0 ; clear PDE 0
                            <1> dapd_0:
00005D15 AD                 <1>       lodsd
00005D16 A801               <1>       test   al, PDE_A_PRESENT ; bit 0, present flag (must be 1)
00005D18 7409               <1>       jz     short dapd_1
00005D1A 662500F0           <1>       and    ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
00005D1E E812000000         <1>       call   deallocate_page_table
                            <1> dapd_1:
00005D23 41                 <1>       inc    ecx ; page directory entry index
00005D24 81F900040000       <1>       cmp    ecx, PAGE_SIZE / 4 ; 1024
00005D2A 72E9               <1>       jb     short dapd_0
                            <1> dapd_2:
00005D2C 58                 <1>       pop    eax
00005D2D E87F000000         <1>       call   deallocate_page     ; deallocate the page dir's itself
00005D32 59                 <1>       pop    ecx
00005D33 5E                 <1>       pop    esi
00005D34 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>       ;
00005D35 56             <1>       push  esi
00005D36 57             <1>       push  edi
00005D37 52             <1>       push  edx
00005D38 50             <1>       push  eax ; *
00005D39 89C6           <1>       mov   esi, eax
00005D3B 31FF           <1>       xor   edi, edi ; 0
                        <1> dapt_0:
00005D3D AD             <1>       lodsd
00005D3E A801           <1>       test  al, PTE_A_PRESENT ; bit 0, present flag (must be 1)
00005D40 7441           <1>       jz    short dapt_1
                        <1>       ;
00005D42 A802           <1>       test  al, PTE_A_WRITE   ; bit 1, writable (r/w) flag
                        <1>                               ; (must be 1)
00005D44 754C           <1>       jnz   short dapt_3
                        <1>       ; Read only -duplicated- page (belongs to a parent or a child)
00005D46 66A90002       <1>         test    ax, PTE_DUPLICATED ; Was this page duplicated
                        <1>                                    ; as child's page ?
00005D4A 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)
00005D4C 53             <1>       push  ebx
00005D4D 51             <1>       push  ecx
00005D4E 66C1E102       <1>       shl   cx, 2 ; *4
00005D52 01CB           <1>       add   ebx, ecx ; PDE offset (for the parent)
00005D54 8B0B           <1>       mov   ecx, [ebx]
00005D56 F6C101         <1>       test  cl, PDE_A_PRESENT ; present (valid) or not ?
00005D59 7435           <1>       jz    short dapt_2 ; parent process does not use this page
00005D5B 6681E100F0     <1>       and   cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
                        <1>       ; EDI = page table entry index (0-1023)
00005D60 89FA           <1>       mov   edx, edi
00005D62 66C1E202       <1>       shl   dx, 2 ; *4
00005D66 01CA           <1>       add   edx, ecx ; PTE offset (for the parent)
00005D68 8B1A           <1>       mov   ebx, [edx]
00005D6A F6C301         <1>       test  bl, PTE_A_PRESENT ; present or not ?
00005D6D 7421           <1>       jz    short dapt_2 ; parent process does not use this page
00005D6F 662500F0       <1>       and   ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
00005D73 6681E300F0     <1>       and   bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
00005D78 39D8           <1>       cmp   eax, ebx      ; parent's and child's pages are same ?
00005D7A 7514           <1>       jne   short dapt_2 ; not same page
                        <1>                          ; deallocate the child's page
00005D7C 800A02         <1>         or      byte [edx], PTE_A_WRITE ; convert to writable page (parent)
00005D7F 59             <1>       pop   ecx
00005D80 5B             <1>       pop   ebx
00005D81 EB1A           <1>       jmp   short dapt_4
                        <1> dapt_1:
00005D83 09C0           <1>       or    eax, eax      ; swapped page ?
00005D85 741D           <1>       jz    short dapt_5 ; no
                        <1>                          ; yes
00005D87 D1E8           <1>       shr   eax, 1
00005D89 E8CA040000     <1>       call  unlink_swap_block ; Deallocate swapped page block
                        <1>                               ; on the swap disk (or in file)
00005D8E EB14           <1>       jmp   short dapt_5
                        <1> dapt_2:
00005D90 59             <1>       pop   ecx
00005D91 5B             <1>       pop   ebx
                        <1> dapt_3:
                        <1>       ; 12/07/2016
00005D92 66A90004       <1>       test  ax, PTE_SHARED ; shared or direct memory access indicator
00005D96 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
00005D98 E814000000     <1>       call  deallocate_page ; set the mem allocation bit of this page
                        <1> dapt_4:
00005D9D C746FC00000000 <1>       mov   dword [esi-4], 0 ; clear/reset PTE (child, dupl. as parent)
                        <1> dapt_5:
00005DA4 47             <1>       inc   edi ; page table entry index
00005DA5 81FF00040000   <1>       cmp   edi, PAGE_SIZE / 4 ; 1024
00005DAB 7290           <1>       jb    short dapt_0
                        <1>       ;
00005DAD 58             <1>       pop   eax ; *
00005DAE 5A             <1>       pop   edx
00005DAF 5F             <1>       pop   edi
00005DB0 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>       ;
00005DB1 53             <1>       push  ebx
00005DB2 52             <1>       push  edx
                        <1>       ;
00005DB3 C1E80C         <1>       shr   eax, PAGE_SHIFT      ; shift physical address to
                        <1>                                  ; 12 bits right
```

```
                               <1>                                ; to get page number
00005DB6 89C2                  <1>        mov    edx, eax
                               <1>        ; 15/09/2015
00005DB8 C1EA03                <1>        shr    edx, 3           ; to get offset to M.A.T.
                               <1>                                ; (1 allocation bit = 1 page)
                               <1>                                ; (1 allocation bytes = 8 pages)
00005DBB 80E2FC                <1>        and    dl, 0FCh         ; clear lower 2 bits
                               <1>                                ; (to get 32 bit position)
                               <1>        ;
00005DBE BB00001000            <1>        mov    ebx, MEM_ALLOC_TBL   ; Memory Allocation Table address
00005DC3 01D3                  <1>        add    ebx, edx
00005DC5 83E01F                <1>        and    eax, 1Fh         ; lower 5 bits only
                               <1>                                ; (allocation bit position)
00005DC8 3B15[CC890100]        <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)
00005DCE 7306                  <1>        jnb    short dap_1      ; no
00005DD0 8915[CC890100]        <1>        mov    [next_page], edx     ; yes
                               <1> dap_1:
00005DD6 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.
00005DD9 FF05[C8890100]        <1>          inc     dword [free_pages]
                               <1> dap_2:
00005DDF 5A                    <1>        pop    edx
00005DE0 5B                    <1>        pop    ebx
00005DE1 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>       ;
00005DE2 E8ECFDFFFF     <1>       call   allocate_page
00005DE7 723E           <1>       jc     short dpd_err
                        <1>       ;
00005DE9 55             <1>       push   ebp ; 20/07/2015
00005DEA 56             <1>       push   esi
00005DEB 57             <1>       push   edi
00005DEC 53             <1>       push   ebx
00005DED 51             <1>       push   ecx
00005DEE 8B35[B8030300] <1>       mov    esi, [u.pgdir]
00005DF4 89C7           <1>       mov    edi, eax
00005DF6 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)
00005DF7 A5             <1>       movsd
00005DF8 BD00004000     <1>       mov    ebp, 1024*4096 ; pass the 1st 4MB (system space)
00005DFD B9FF030000     <1>       mov    ecx, (PAGE_SIZE / 4) - 1 ; 1023
                        <1> dpd_0:
00005E02 AD             <1>       lodsd
                        <1>       ;or    eax, eax
                        <1>         ;jnz     short dpd_1
00005E03 A801           <1>       test   al, PDE_A_PRESENT ;  bit 0 =  1
00005E05 7508           <1>       jnz    short dpd_1
                        <1>       ; 20/07/2015 (virtual address at the end of the page table)
00005E07 81C500004000   <1>       add    ebp, 1024*4096 ; page size * PTE count
00005E0D EB0F           <1>       jmp    short dpd_2
                        <1> dpd_1:
00005E0F 662500F0       <1>       and    ax, PDE_A_CLEAR ; 0F000h ; clear attribute bits
00005E13 89C3           <1>       mov    ebx, eax
                        <1>       ; EBX = Parent's page table address
00005E15 E81F000000     <1>       call   duplicate_page_table
00005E1A 720C           <1>       jc     short dpd_p_err
                        <1>       ; EAX = Child's page table address
00005E1C 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:
00005E1E AB             <1>       stosd
00005E1F E2E1           <1>       loop   dpd_0
                        <1>       ;
00005E21 58             <1>       pop    eax  ; restore child's page directory address
                        <1> dpd_3:
00005E22 59             <1>       pop    ecx
00005E23 5B             <1>       pop    ebx
00005E24 5F             <1>       pop    edi
00005E25 5E             <1>       pop    esi
00005E26 5D             <1>       pop    ebp ; 20/07/2015
                        <1> dpd_err:
00005E27 C3             <1>       retn
                        <1> dpd_p_err:
                        <1>       ; release the allocated pages missing (recover free space)
00005E28 58             <1>       pop    eax  ; the new page directory address (physical)
00005E29 8B1D[B8030300] <1>       mov    ebx, [u.pgdir] ; parent's page directory address
00005E2F E8D8FEFFFF     <1>       call   deallocate_page_dir
00005E34 29C0           <1>       sub    eax, eax ; 0
00005E36 F9             <1>       stc
00005E37 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>       ;
00005E39 E895FDFFFF     <1>       call   allocate_page
00005E3E 726A           <1>       jc     short dpt_err
                        <1>       ;
00005E40 50             <1>       push   eax ; *
00005E41 56             <1>       push   esi
00005E42 57             <1>       push   edi
00005E43 52             <1>       push   edx
00005E44 51             <1>       push   ecx
                        <1>       ;
00005E45 89DE           <1>       mov    esi, ebx
00005E47 89C7           <1>       mov    edi, eax
00005E49 89C2           <1>       mov    edx, eax
00005E4B 81C200100000   <1>       add    edx, PAGE_SIZE
```

```
                        <1> dpt_0:
00005E51 AD             <1>         lodsd
00005E52 21C0           <1>         and     eax, eax
00005E54 7444           <1>         jz      short dpt_3
00005E56 A801           <1>         test    al, PTE_A_PRESENT ;  bit 0 =  1
00005E58 7507           <1>         jnz     short dpt_1
                        <1>         ; 20/07/2015
                        <1>         ; ebp = virtual (linear) address of the memory page
00005E5A E83F050000     <1>         call    reload_page ; 28/04/2015
00005E5F 7244           <1>         jc      short dpt_p_err
                        <1> dpt_1:
                        <1>         ; 21/09/2015
00005E61 89C1           <1>         mov     ecx, eax
00005E63 662500F0       <1>         and     ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
00005E67 F6C102         <1>         test    cl, PTE_A_WRITE ; writable page ?
00005E6A 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
00005E6C E862FDFFFF     <1>         call    allocate_page
00005E71 7232           <1>         jc      short dpt_p_err
00005E73 57             <1>         push    edi
00005E74 56             <1>         push    esi
00005E75 89CE           <1>         mov     esi, ecx
00005E77 89C7           <1>         mov     edi, eax
00005E79 B900040000     <1>         mov     ecx, PAGE_SIZE/4
00005E7E F3A5           <1>         rep     movsd  ; copy page (4096 bytes)
00005E80 5E             <1>         pop     esi
00005E81 5F             <1>         pop     edi
                        <1>         ;
00005E82 53             <1>         push    ebx
00005E83 50             <1>         push    eax
                        <1>         ; 20/07/2015
00005E84 89EB           <1>         mov     ebx, ebp
                        <1>         ; ebx = virtual (linear) address of the memory page
00005E86 E887030000     <1>         call    add_to_swap_queue
00005E8B 58             <1>         pop     eax
00005E8C 5B             <1>         pop     ebx
                        <1>         ; 21/09/2015
00005E8D 0C07           <1>         or      al, PTE_A_USER+PTE_A_WRITE+PTE_A_PRESENT
                        <1>                 ; user + writable + present page
00005E8F EB09           <1>         jmp     short dpt_3
                        <1> dpt_2:
                        <1>         ;or     ax, PTE_A_USER+PTE_A_PRESENT
00005E91 0C05           <1>         or      al, PTE_A_USER+PTE_A_PRESENT
                        <1>                     ; (read only page!)
00005E93 8946FC         <1>         mov     [esi-4], eax ; update parent's PTE
00005E96 660D0002       <1>         or       ax, PTE_DUPLICATED  ; (read only page & duplicated PTE!)
                        <1> dpt_3:
00005E9A AB             <1>         stosd  ; EDI points to child's PTE
                        <1>         ;
00005E9B 81C500100000   <1>         add     ebp, 4096 ; 20/07/2015 (next page)
                        <1>         ;
00005EA1 39D7           <1>         cmp     edi, edx
00005EA3 72AC           <1>         jb      short dpt_0
                        <1> dpt_p_err:
00005EA5 59             <1>         pop     ecx
00005EA6 5A             <1>         pop     edx
00005EA7 5F             <1>         pop     edi
00005EA8 5E             <1>         pop     esi
00005EA9 58             <1>         pop     eax ; *
                        <1> dpt_err:
00005EAA 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
```

```
1193                                 <1>       ;            a page-protection violation. When not set,
1194                                 <1>       ;            it was caused by a non-present page.
1195                                 <1>       ; W : WRITE   -     When set, the page fault was caused by
1196                                 <1>       ;            a page write. When not set, it was caused
1197                                 <1>       ;            by a page read.
1198                                 <1>       ; U : USER    -     When set, the page fault was caused
1199                                 <1>       ;            while CPL = 3.
1200                                 <1>       ;            This does not necessarily mean that
1201                                 <1>       ;            the page fault was a privilege violation.
1202                                 <1>       ; R : RESERVD -     When set, the page fault was caused by
1203                                 <1>       ;     WRITE  reading a 1 in a reserved field.
1204                                 <1>       ; I : INSTRUC -     When set, the page fault was caused by
1205                                 <1>       ;     FETCH  an instruction fetch
1206                                 <1>       ;
1207                                 <1>       ;; x86 (32 bit) VIRTUAL ADDRESS TRANSLATION
1208                                 <1>       ;  31               22                  12 11                     0
1209                                 <1>       ; +------------------+------------------+-----------------------+
1210                                 <1>             ; | PAGE DIR. ENTRY # | PAGE TAB. ENTRY # |        OFFSET         |
1211                                 <1>             ; +-------------------+-------------------+-----------------------+
1212                                 <1>       ;
1213                                 <1>
1214                                 <1>       ;; CR3 REGISTER (Control Register 3)
1215                                 <1>       ;  31                                    12             5 4 3 2   0
1216                                 <1>       ; +----------------------------------------+-------------+---+-----+
1217                                 <1>             ; |                                        |             |P|P|     |
1218                                 <1>             ; |   PAGE DIRECTORY TABLE BASE ADDRESS    |  reserved   |C|W|rsvrd|
1219                                 <1>             ; |                                        |             |D|T|     |
1220                                 <1>       ; +----------------------------------------+-------------+---+-----+
1221                                 <1>       ;
1222                                 <1>       ;      PWT    - WRITE THROUGH
1223                                 <1>       ;      PCD    - CACHE DISABLE
1224                                 <1>       ;
1225                                 <1>       ;
1226                                 <1>       ;; x86 PAGE DIRECTORY ENTRY (4 KByte Page)
1227                                 <1>       ;  31                                  12 11  9 8 7 6 5 4 3 2 1 0
1228                                 <1>       ; +--------------------------------------+-----+---+-+-+---+-+-+-+
1229                                 <1>             ; |                                      |     | | | | |P|P|U|R| |
1230                                 <1>             ; |     PAGE TABLE BASE ADDRESS 31..12   | AVL |G|0|D|A|C|W|/|/|P|
1231                                 <1>             ; |                                      |     | | | | |D|T|S|W| |
1232                                 <1>       ; +--------------------------------------+-----+---+-+-+---+-+-+-+
1233                                 <1>       ;
1234                                 <1>         ;       P      - PRESENT
1235                                 <1>         ;       R/W    - READ/WRITE
1236                                 <1>         ;       U/S    - USER/SUPERVISOR
1237                                 <1>       ;      PWT    - WRITE THROUGH
1238                                 <1>       ;      PCD    - CACHE DISABLE
1239                                 <1>       ;      A      - ACCESSED
1240                                 <1>         ;       D      - DIRTY (IGNORED)
1241                                 <1>       ;      PAT    - PAGE ATTRIBUTE TABLE INDEX (CACHE BEHAVIOR)
1242                                 <1>       ;      G      - GLOBAL     (IGNORED)
1243                                 <1>         ;       AVL    - AVAILABLE FOR SYSTEMS PROGRAMMER USE
1244                                 <1>       ;
1245                                 <1>       ;
1246                                 <1>       ;; x86 PAGE TABLE ENTRY (4 KByte Page)
1247                                 <1>       ;  31                                  12 11  9 8 7 6 5 4 3 2 1 0
1248                                 <1>       ; +--------------------------------------+-----+---+-+-+---+-+-+-+
1249                                 <1>             ; |                                      |     | |P| | |P|P|U|R| |
1250                                 <1>             ; |     PAGE FRAME BASE ADDRESS 31..12   | AVL |G|A|D|A|C|W|/|/|P|
1251                                 <1>             ; |                                      |     | |T| | |D|T|S|W| |
1252                                 <1>       ; +--------------------------------------+-----+---+-+-+---+-+-+-+
1253                                 <1>       ;
1254                                 <1>         ;       P      - PRESENT
1255                                 <1>         ;       R/W    - READ/WRITE
1256                                 <1>         ;       U/S    - USER/SUPERVISOR
1257                                 <1>       ;      PWT    - WRITE THROUGH
1258                                 <1>       ;      PCD    - CACHE DISABLE
1259                                 <1>       ;      A      - ACCESSED
1260                                 <1>         ;       D      - DIRTY
1261                                 <1>       ;      PAT    - PAGE ATTRIBUTE TABLE INDEX (CACHE BEHAVIOR)
1262                                 <1>       ;      G      - GLOBAL
1263                                 <1>         ;       AVL    - AVAILABLE FOR SYSTEMS PROGRAMMER USE
1264                                 <1>       ;
1265                                 <1>       ;
1266                                 <1>       ;; 80386 PAGE TABLE ENTRY (4 KByte Page)
1267                                 <1>       ;  31                                  12 11  9 8 7 6 5 4 3 2 1 0
1268                                 <1>       ; +--------------------------------------+-----+-+-+-+-+---+-+-+-+
1269                                 <1>             ; |                                      |     | | | | | | |U|R| |
1270                                 <1>             ; |     PAGE FRAME BASE ADDRESS 31..12   | AVL |0|0|D|A|0|0|/|/|P|
1271                                 <1>             ; |                                      |     | | | | | | |S|W| |
1272                                 <1>             ; +--------------------------------------+-----+-+-+-+-+---+-+-+-+
1273                                 <1>       ;
1274                                 <1>         ;       P      - PRESENT
1275                                 <1>         ;       R/W    - READ/WRITE
1276                                 <1>         ;       U/S    - USER/SUPERVISOR
1277                                 <1>         ;       D      - DIRTY
1278                                 <1>         ;       AVL    - AVAILABLE FOR SYSTEMS PROGRAMMER USE
1279                                 <1>       ;
1280                                 <1>         ;       NOTE: 0 INDICATES INTEL RESERVED. DO NOT DEFINE.
1281                                 <1>       ;
1282                                 <1>       ;
1283                                 <1>       ;; Invalid Page Table Entry
1284                                 <1>       ; 31                                                          1 0
1285                                 <1>             ; +-------------------------------------------------------------+-+
1286                                 <1>             ; |                                                             | |
1287                                 <1>             ; |                           AVAILABLE                         |0|
1288                                 <1>             ; |                                                             | |
1289                                 <1>             ; +-------------------------------------------------------------+-+
1290                                 <1>       ;
1291                                 <1>
1292 00005EAB 53                    <1>       push   ebx
1293 00005EAC 52                    <1>       push   edx
1294 00005EAD 51                    <1>       push   ecx
1295                                 <1>       ;
1296                                 <1>       ; 21/09/2015 (debugging)
1297 00005EAE FF05[CC030300]        <1>       inc    dword [u.pfcount] ; page fault count for running process
```

```
1298 00005EB4 FF05[80050300]      <1>       inc    dword [PF_Count] ; total page fault count
1299                              <1>       ; 28/06/2015
1300                              <1>       ;mov   edx, [error_code] ; Lower 5 bits are valid
1301 00005EBA 8A15[78050300]      <1>       mov    dl, [error_code]
1302                              <1>       ;
1303 00005EC0 F6C201              <1>       test   dl, 1  ; page fault was caused by a non-present page
1304                              <1>                     ; sign
1305 00005EC3 7422                <1>       jz     short pfh_alloc_np
1306                              <1>       ;
1307                              <1>       ; If it is not a 'write on read only page' type page fault
1308                              <1>       ; major page fault error with minor reason must be returned without
1309                              <1>       ; fixing the problem. 'sys_exit with error' will be needed
1310                              <1>       ; after return here!
1311                              <1>       ; Page fault will be remedied, by copying page contents
1312                              <1>       ; to newly allocated page with write permission;
1313                              <1>       ; sys_fork -> sys_exec -> copy on write, demand paging method is
1314                              <1>       ; used for working with minimum possible memory usage.
1315                              <1>       ; sys_fork will duplicate page directory and tables of parent
1316                              <1>       ; process with 'read only' flag. If the child process attempts to
1317                              <1>       ; write on these read only pages, page fault will be directed here
1318                              <1>       ; for allocating a new page with same data/content.
1319                              <1>       ;
1320                              <1>       ; IMPORTANT : Retro UNIX 386 v1 (and SINGLIX and TR-DOS)
1321                              <1>       ; will not force to separate CODE and DATA space
1322                              <1>       ; in a process/program...
1323                              <1>       ; CODE segment/section may contain DATA!
1324                              <1>       ; It is flat, smoth and simplest programming method already as in
1325                              <1>       ; Retro UNIX 8086 v1 and MS-DOS programs.
1326                              <1>       ;
1327 00005EC5 F6C202              <1>       test   dl, 2  ; page fault was caused by a page write
1328                              <1>                     ; sign
1329 00005EC8 0F84AB000000        <1>         jz      pfh_p_err
1330                              <1>       ; 31/08/2015
1331 00005ECE F6C204              <1>       test   dl, 4  ; page fault was caused while CPL = 3 (user mode)
1332                              <1>                     ; sign.  (U+W+P = 4+2+1 = 7)
1333 00005ED1 0F84A2000000        <1>         jz   pfh_pv_err
1334                              <1>       ;
1335                              <1>       ; make a new page and copy the parent's page content
1336                              <1>       ; as the child's new page content
1337                              <1>       ;
1338 00005ED7 0F20D3              <1>       mov    ebx, cr2 ; CR2 contains the linear address
1339                              <1>                       ; which has caused to page fault
1340 00005EDA E8A2000000          <1>       call   copy_page
1341 00005EDF 0F828D000000        <1>         jc     pfh_im_err ; insufficient memory
1342                              <1>       ;
1343 00005EE5 EB7D                <1>         jmp     pfh_cpp_ok
1344                              <1>       ;
1345                              <1> pfh_alloc_np:
1346 00005EE7 E8E7FCFFFF          <1>       call   allocate_page; (allocate a new page)
1347 00005EEC 0F8280000000        <1>         jc     pfh_im_err      ; 'insufficient memory' error
1348                              <1> pfh_chk_cpl:
1349                              <1>       ; EAX = Physical (base) address of the allocated (new) page
1350                              <1>             ; (Lower 12 bits are ZERO, because
1351                              <1>             ;     the address is on a page boundary)
1352 00005EF2 80E204              <1>       and    dl, 4  ; CPL = 3 ?
1353 00005EF5 7505                <1>       jnz    short pfh_um
1354                              <1>                     ; Page fault handler for kernel/system mode (CPL=0)
1355 00005EF7 0F20DB              <1>       mov    ebx, cr3 ; CR3 (Control Register 3) contains physical address
1356                              <1>                       ; of the current/active page directory
1357                              <1>                       ; (Always kernel/system mode page directory, here!)
1358                              <1>                       ; Note: Lower 12 bits are 0. (page boundary)
1359 00005EFA EB06                <1>       jmp    short pfh_get_pde
1360                              <1>       ;
1361                              <1> pfh_um:                   ; Page fault handler for user/appl. mode (CPL=3)
1362 00005EFC 8B1D[B8030300]      <1>       mov    ebx, [u.pgdir] ; Page directory of current/active process
1363                              <1>                     ; Physical address of the USER's page directory
1364                              <1>                     ; Note: Lower 12 bits are 0. (page boundary)
1365                              <1> pfh_get_pde:
1366 00005F02 80CA03              <1>       or     dl, 3  ; USER + WRITE + PRESENT or SYSTEM + WRITE + PRESENT
1367 00005F05 0F20D1              <1>       mov    ecx, cr2 ; CR2 contains the virtual address
1368                              <1>                       ; which has been caused to page fault
1369                              <1>                       ;
1370 00005F08 C1E914              <1>       shr    ecx, 20       ; shift 20 bits right
1371 00005F0B 80E1FC              <1>       and    cl, 0FCh ; mask lower 2 bits to get PDE offset
1372                              <1>       ;
1373 00005F0E 01CB                <1>       add    ebx, ecx ; now, EBX points to the relevant page dir entry
1374 00005F10 8B0B                <1>       mov    ecx, [ebx] ; physical (base) address of the page table
1375 00005F12 F6C101              <1>       test   cl, 1   ; check bit 0 is set (1) or not (0).
1376 00005F15 740B                <1>       jz     short pfh_set_pde ; Page directory entry is not valid,
1377                              <1>                                ; set/validate page directory entry
1378 00005F17 6681E100F0          <1>       and    cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
1379 00005F1C 89CB                <1>       mov    ebx, ecx ; Physical address of the page table
1380 00005F1E 89C1                <1>       mov    ecx, eax ; new page address (physical)
1381 00005F20 EB16                <1>       jmp    short pfh_get_pte
1382                              <1> pfh_set_pde:
1383                              <1>       ;; NOTE: Page directories and page tables never be swapped out!
1384                              <1>       ;;      (So, we know this PDE is empty or invalid)
1385                              <1>       ;
1386 00005F22 08D0                <1>       or     al, dl  ; lower 3 bits are used as U/S, R/W, P flags
1387 00005F24 8903                <1>       mov    [ebx], eax ; Let's put the new page directory entry here !
1388 00005F26 30C0                <1>       xor    al, al  ; clear lower (3..8) bits
1389 00005F28 89C3                <1>       mov    ebx, eax
1390 00005F2A E8A4FCFFFF          <1>       call   allocate_page ; (allocate a new page)
1391 00005F2F 7241                <1>       jc     short pfh_im_err   ; 'insufficient memory' error
1392                              <1> pfh_spde_1:
1393                              <1>       ; EAX = Physical (base) address of the allocated (new) page
1394 00005F31 89C1                <1>       mov    ecx, eax
1395 00005F33 E815FDFFFF          <1>       call   clear_page ; Clear page content
1396                              <1> pfh_get_pte:
1397 00005F38 0F20D0              <1>       mov    eax, cr2 ; virtual address
1398                              <1>                       ; which has been caused to page fault
1399 00005F3B 89C7                <1>       mov    edi, eax ; 20/07/2015
1400 00005F3D C1E80C              <1>       shr    eax, 12       ; shift 12 bit right to get
1401                              <1>                       ; higher 20 bits of the page fault address
1402 00005F40 25FF030000          <1>       and    eax, 3FFh ; mask PDE# bits, the result is PTE# (0 to 1023)
```

```
1403 00005F45 C1E002              <1>       shl    eax, 2 ; shift 2 bits left to get PTE offset
1404 00005F48 01C3                <1>       add    ebx, eax ; now, EBX points to the relevant page table entry
1405 00005F4A 8B03                <1>       mov    eax, [ebx] ; get previous value of pte
1406                              <1>              ; bit 0 of EAX is always 0 (otherwise we would not be here)
1407 00005F4C 21C0                <1>       and    eax, eax
1408 00005F4E 7410                <1>       jz     short pfh_gpte_1
1409                              <1>       ; 20/07/2015
1410 00005F50 87D9                <1>       xchg   ebx, ecx ; new page address (physical)
1411 00005F52 55                  <1>       push   ebp ; 20/07/2015
1412 00005F53 0F20D5              <1>       mov    ebp, cr2
1413                              <1>              ; ECX = physical address of the page table entry
1414                              <1>              ; EBX = Memory page address (physical!)
1415                              <1>              ; EAX = Swap disk (offset) address
1416                              <1>              ; EBP = virtual address (page fault address)
1417 00005F56 E8B7000000          <1>       call   swap_in
1418 00005F5B 5D                  <1>       pop    ebp
1419 00005F5C 7210                <1>       jc      short pfh_err_retn
1420 00005F5E 87CB                <1>       xchg   ecx, ebx
1421                              <1>              ; EBX = physical address of the page table entry
1422                              <1>              ; ECX = new page
1423                              <1> pfh_gpte_1:
1424 00005F60 08D1                <1>       or     cl, dl ; lower 3 bits are used as U/S, R/W, P flags
1425 00005F62 890B                <1>       mov    [ebx], ecx ; Let's put the new page table entry here !
1426                              <1> pfh_cpp_ok:
1427                              <1>       ; 20/07/2015
1428 00005F64 0F20D3              <1>       mov    ebx, cr2
1429 00005F67 E8A6020000          <1>       call   add_to_swap_queue
1430                              <1>       ;
1431                              <1>       ; The new PTE (which contains the new page) will be added to
1432                              <1>       ; the swap queue, here.
1433                              <1>       ; (Later, if memory will become insufficient,
1434                              <1>       ; one page will be swapped out which is at the head of
1435                              <1>       ; the swap queue by using FIFO and access check methods.)
1436                              <1>       ;
1437 00005F6C 31C0                <1>       xor    eax, eax  ; 0
1438                              <1>       ;
1439                              <1> pfh_err_retn:
1440 00005F6E 59                  <1>       pop    ecx
1441 00005F6F 5A                  <1>       pop    edx
1442 00005F70 5B                  <1>       pop    ebx
1443 00005F71 C3                  <1>       retn
1444                              <1>
1445                              <1> pfh_im_err:
1446 00005F72 B8E4000000          <1>       mov    eax, ERR_MAJOR_PF + ERR_MINOR_IM ; Error code in AX
1447                              <1>                     ; Major (Primary) Error: Page Fault
1448                              <1>                     ; Minor (Secondary) Error: Insufficient Memory !
1449 00005F77 EBF5                <1>       jmp    short pfh_err_retn
1450                              <1>
1451                              <1>
1452                              <1> pfh_p_err: ; 09/03/2015
1453                              <1> pfh_pv_err:
1454                              <1>       ; Page fault was caused by a protection-violation
1455 00005F79 B8E6000000          <1>       mov    eax, ERR_MAJOR_PF + ERR_MINOR_PV ; Error code in AX
1456                              <1>                     ; Major (Primary) Error: Page Fault
1457                              <1>                     ; Minor (Secondary) Error: Protection violation !
1458 00005F7E F9                  <1>       stc
1459 00005F7F EBED                <1>       jmp    short pfh_err_retn
1460                              <1>
1461                              <1> copy_page:
1462                              <1>       ; 22/09/2015
1463                              <1>       ; 21/09/2015
1464                              <1>       ; 19/09/2015
1465                              <1>       ; 07/09/2015
1466                              <1>       ; 31/08/2015
1467                              <1>       ; 20/07/2015
1468                              <1>       ; 05/05/2015
1469                              <1>       ; 03/05/2015
1470                              <1>       ; 18/04/2015
1471                              <1>       ; 12/04/2015
1472                              <1>       ; 30/10/2014
1473                              <1>       ; 18/10/2014 (Retro UNIX 386 v1 - beginning)
1474                              <1>       ;
1475                              <1>       ; INPUT ->
1476                              <1>       ;      EBX = Virtual (linear) address of source page
1477                              <1>       ;           (Page fault address)
1478                              <1>       ; OUTPUT ->
1479                              <1>       ;      EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
1480                              <1>       ;      (corresponding PAGE TABLE ENTRY is mapped/set)
1481                              <1>       ;      EAX = 0 (CF = 1)
1482                              <1>       ;            if there is not a free page to be allocated
1483                              <1>       ;      (page content of the source page will be copied
1484                              <1>       ;      onto the target/new page)
1485                              <1>       ;
1486                              <1>       ; Modified Registers -> ecx, ebx (except EAX)
1487                              <1>       ;
1488 00005F81 56                  <1>       push   esi
1489 00005F82 57                  <1>       push   edi
1490                              <1>       ;push  ebx
1491                              <1>       ;push  ecx
1492 00005F83 31F6                <1>       xor    esi, esi
1493 00005F85 C1EB0C              <1>       shr    ebx, 12 ; shift 12 bits right to get PDE & PTE numbers
1494 00005F88 89D9                <1>       mov    ecx, ebx ; save page fault address (as 12 bit shifted)
1495 00005F8A C1EB08              <1>       shr    ebx, 8  ; shift 8 bits right and then
1496 00005F8D 80E3FC              <1>       and    bl, 0FCh ; mask lower 2 bits to get PDE offset
1497 00005F90 89DF                <1>       mov    edi, ebx ; save it for the parent of current process
1498 00005F92 031D[B8030300]      <1>       add    ebx, [u.pgdir] ; EBX points to the relevant page dir entry
1499 00005F98 8B03                <1>       mov    eax, [ebx] ; physical (base) address of the page table
1500 00005F9A 662500F0            <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
1501 00005F9E 89CB                <1>       mov    ebx, ecx   ; (restore higher 20 bits of page fault address)
1502 00005FA0 81E3FF030000        <1>       and    ebx, 3FFh  ; mask PDE# bits, the result is PTE# (0 to 1023)
1503 00005FA6 66C1E302            <1>       shl    bx, 2    ; shift 2 bits left to get PTE offset
1504 00005FAA 01C3                <1>       add    ebx, eax   ; EBX points to the relevant page table entry
1505                              <1>       ; 07/09/2015
1506 00005FAC 66F7030002          <1>         test    word [ebx], PTE_DUPLICATED ; (Does current process share this
1507                              <1>                              ; read only page as a child process?)
```

```
1508 00005FB1 7509                <1>       jnz    short cpp_0 ; yes
1509 00005FB3 8B0B                <1>       mov    ecx, [ebx] ; PTE value
1510 00005FB5 6681E100F0          <1>       and    cx, PTE_A_CLEAR ; 0F000h  ; clear page attributes
1511 00005FBA EB32                <1>       jmp    short cpp_1
1512                              <1> cpp_0:
1513 00005FBC 89FE                <1>       mov    esi, edi
1514 00005FBE 0335[BC030300]      <1>       add    esi, [u.ppgdir] ; the parent's page directory entry
1515 00005FC4 8B06                <1>       mov    eax, [esi] ; physical (base) address of the page table
1516 00005FC6 662500F0            <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
1517 00005FCA 89CE                <1>       mov    esi, ecx   ; (restore higher 20 bits of page fault address)
1518 00005FCC 81E6FF030000        <1>       and    esi, 3FFh  ; mask PDE# bits, the result is PTE# (0 to 1023)
1519 00005FD2 66C1E602            <1>       shl    si, 2     ; shift 2 bits left to get PTE offset
1520 00005FD6 01C6                <1>       add    esi, eax   ; EDX points to the relevant page table entry
1521 00005FD8 8B0E                <1>       mov    ecx, [esi] ; PTE value of the parent process
1522                              <1>       ; 21/09/2015
1523 00005FDA 8B03                <1>       mov    eax, [ebx] ; PTE value of the child process
1524 00005FDC 662500F0            <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear page attributes
1525                              <1>       ;
1526 00005FE0 F6C101              <1>       test   cl, PTE_A_PRESENT ; is it a present/valid page ?
1527 00005FE3 7424                <1>       jz     short cpp_3 ; the parent's page is not same page
1528                              <1>       ;
1529 00005FE5 6681E100F0          <1>       and    cx, PTE_A_CLEAR ; 0F000h ; clear page attributes
1530 00005FEA 39C8                <1>       cmp    eax, ecx   ; Same page?
1531 00005FEC 751B                <1>       jne    short cpp_3 ; Parent page and child page are not same
1532                              <1>                        ; Convert child's page to writable page
1533                              <1> cpp_1:
1534 00005FEE E8E0FBFFFF          <1>       call   allocate_page
1535 00005FF3 721A                <1>       jc     short cpp_4 ; 'insufficient memory' error
1536 00005FF5 21F6                <1>       and    esi, esi    ; check ESI is valid or not
1537 00005FF7 7405                <1>       jz     short cpp_2
1538                              <1>              ; Convert read only page to writable page
1539                              <1>              ;(for the parent of the current process)
1540                              <1>       ;and   word [esi], PTE_A_CLEAR ; 0F000h
1541                              <1>       ; 22/09/2015
1542 00005FF9 890E                <1>       mov    [esi], ecx
1543 00005FFB 800E07              <1>       or     byte [esi], PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER
1544                              <1>                          ; 1+2+4 = 7
1545                              <1> cpp_2:
1546 00005FFE 89C7                <1>       mov    edi, eax ; new page address of the child process
1547                              <1>       ; 07/09/2015
1548 00006000 89CE                <1>       mov    esi, ecx ; the page address of the parent process
1549 00006002 B900040000          <1>       mov    ecx, PAGE_SIZE / 4
1550 00006007 F3A5                <1>       rep    movsd ; 31/08/2015
1551                              <1> cpp_3:
1552 00006009 0C07                <1>       or     al, PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER ; 1+2+4 = 7
1553 0000600B 8903                <1>       mov    [ebx], eax ; Update PTE
1554 0000600D 28C0                <1>       sub    al, al ; clear attributes
1555                              <1> cpp_4:
1556                              <1>       ;pop   ecx
1557                              <1>       ;pop   ebx
1558 0000600F 5F                  <1>       pop    edi
1559 00006010 5E                  <1>       pop    esi
1560 00006011 C3                  <1>       retn
1561                              <1>
1562                              <1> ;; 28/04/2015
1563                              <1> ;; 24/10/2014
1564                              <1> ;; 21/10/2014 (Retro UNIX 386 v1 - beginning)
1565                              <1> ;; SWAP_PAGE_QUEUE (4096 bytes)
1566                              <1> ;;
1567                              <1> ;;   0000   0001   0002   0003   ....   1020   1021   1022   1023
1568                              <1> ;; +------+------+------+------+-    -+------+------+------+------+
1569                              <1> ;; |  pg1 |  pg2 |  pg3 |  pg4 | .... |pg1021|pg1022|pg1023|pg1024|
1570                              <1> ;; +------+------+------+------+-    -+------+------+------+------+
1571                              <1> ;;
1572                              <1> ;; [swpq_last] = 0 to 4096 (step 4) -> the last position on the queue
1573                              <1> ;;
1574                              <1> ;; Method:
1575                              <1> ;;    Swap page queue is a list of allocated pages with physical
1576                              <1> ;;    addresses (system mode virtual adresses = physical addresses).
1577                              <1> ;;    It is used for 'swap_in' and 'swap_out' procedures.
1578                              <1> ;;    When a new page is being allocated, swap queue is updated
1579                              <1> ;;    by 'swap_queue_shift' procedure, header of the queue (offset 0)
1580                              <1> ;;    is checked for 'accessed' flag. If the 1st page on the queue
1581                              <1> ;;    is 'accessed' or 'read only', it is dropped from the list;
1582                              <1> ;;    other pages from the 2nd to the last (in [swpq_last]) shifted
1583                              <1> ;;    to head then the 2nd page becomes the 1st and '[swpq_last]'
1584                              <1> ;;    offset value becomes it's previous offset value - 4.
1585                              <1> ;;    If the 1st page of the swap page queue is not 'accessed'
1586                              <1> ;;    the queue/list is not shifted.
1587                              <1> ;;    After the queue/list shift, newly allocated page is added
1588                              <1> ;;    to the tail of the queue at the [swpq_count*4] position.
1589                              <1> ;;    But, if [swpq_count] > 1023, the newly allocated page
1590                              <1> ;;    will not be added to the tail of swap page queue.
1591                              <1> ;;
1592                              <1> ;;    During 'swap_out' procedure, swap page queue is checked for
1593                              <1> ;;    the first non-accessed, writable page in the list,
1594                              <1> ;;    from the head to the tail. The list is shifted to left
1595                              <1> ;;    (to the head) till a non-accessed page will be found in the list.
1596                              <1> ;;    Then, this page     is swapped out (to disk) and then it is dropped
1597                              <1> ;;    from the list by a final swap queue shift. [swpq_count] value
1598                              <1> ;;    is changed. If all pages on the queue' are 'accessed',
1599                              <1> ;;    'insufficient memory' error will be returned ('swap_out'
1600                              <1> ;;    procedure will be failed)...
1601                              <1> ;;
1602                              <1> ;;    Note: If the 1st page of the queue is an 'accessed' page,
1603                              <1> ;;    'accessed' flag of the page will be reset (0) and that page
1604                              <1> ;;    (PTE) will be added to the tail of the queue after
1605                              <1> ;;    the check, if [swpq_count] < 1023. If [swpq_count] = 1024
1606                              <1> ;;    the queue will be rotated and the PTE in the head will be
1607                              <1> ;;    added to the tail after resetting 'accessed' bit.
1608                              <1> ;;
1609                              <1> ;;
1610                              <1> ;;
1611                              <1> ;; SWAP DISK/FILE (with 4096 bytes swapped page blocks)
1612                              <1> ;;
```

```
1613                    <1> ;;  00000000  00000004  00000008  0000000C   ...   size-8    size-4
1614                    <1> ;; +---------+---------+---------+---------+-- --+---------+---------+
1615                    <1> ;; |descriptr| page(1) | page(2) | page(3) | ... |page(n-1)| page(n) |
1616                    <1> ;; +---------+---------+---------+---------+-- --+---------+---------+
1617                    <1> ;;
1618                    <1> ;; [swpd_next] = the first free block address in swapped page records
1619                    <1> ;;                 for next free block search by 'swap_out' procedure.
1620                    <1> ;; [swpd_size] = swap disk/file size in sectors (512 bytes)
1621                    <1> ;;           NOTE: max. possible swap disk size is 1024 GB
1622                    <1> ;;           (entire swap space must be accessed by using
1623                    <1> ;;           31 bit offset address)
1624                    <1> ;; [swpd_free] = free block (4096 bytes) count in swap disk/file space
1625                    <1> ;; [swpd_start] = absolute/start address of the swap disk/file
1626                    <1> ;;            0 for file, or beginning sector of the swap partition
1627                    <1> ;; [swp_drv] = logical drive description table addr. of swap disk/file
1628                    <1> ;;
1629                    <1> ;;
1630                    <1> ;; Method:
1631                    <1> ;;    When the memory (ram) becomes insufficient, page allocation
1632                    <1> ;;    procedure swaps out a page from memory to the swap disk
1633                    <1> ;;    (partition) or swap file to get a new free page at the memory.
1634                    <1> ;;    Swapping out is performed by using swap page queue.
1635                    <1> ;;
1636                    <1> ;;    Allocation block size of swap disk/file is equal to page size
1637                    <1> ;;    (4096 bytes). Swapping address (in sectors) is recorded
1638                    <1> ;;    into relevant page file entry as 31 bit physical (logical)
1639                    <1> ;;    offset address as 1 bit shifted to left for present flag (0).
1640                    <1> ;;    Swapped page address is between 1 and swap disk/file size - 4.
1641                    <1> ;;    Absolute physical (logical) address of the swapped page is
1642                    <1> ;;    calculated by adding offset value to the swap partition's
1643                    <1> ;;    start address. If the swap device (disk) is a virtual disk
1644                    <1> ;;    or it is a file, start address of the swap disk/volume is 0,
1645                    <1> ;;    and offset value is equal to absolute (physical or logical)
1646                    <1> ;;    address/position. (It has not to be ZERO if the swap partition
1647                    <1> ;;    is in a partitioned virtual hard disk.)
1648                    <1> ;;
1649                    <1> ;;    Note: Swap addresses are always specified/declared in sectors,
1650                    <1> ;;    not in bytes or     in blocks/zones/clusters (4096 bytes) as unit.
1651                    <1> ;;
1652                    <1> ;;    Swap disk/file allocation is mapped via 'Swap Allocation Table'
1653                    <1> ;;    at memory as similar to 'Memory Allocation Table'.
1654                    <1> ;;
1655                    <1> ;;    Every bit of Swap Allocation Table repsesents one swap block
1656                    <1> ;;    (equal to page size) respectively. Bit 0 of the S.A.T. byte 0
1657                    <1> ;;    is reserved for swap disk/file block 0 as descriptor block
1658                    <1> ;;    (also for compatibility with PTE). If bit value is ZERO,
1659                    <1> ;;    it means relevant (respective) block is in use, and,
1660                    <1> ;;    of course, if bit value is 1, it means relevant (respective)
1661                    <1> ;;      swap disk/file block is free.
1662                    <1> ;;    For example: bit 1 of the byte 128 repsesents block 1025
1663                    <1> ;;    (128*8+1) or sector (offset) 8200 on the swap disk or
1664                    <1> ;;    byte (offset/position) 4198400 in the swap file.
1665                    <1> ;;    4GB swap space is represented via 128KB Swap Allocation Table.
1666                    <1> ;;    Initial layout of Swap Allocation Table is as follows:
1667                    <1> ;;    ------------------------------------------------------------
1668                    <1> ;;    01111111111111111111111 .... 11111111111111111111111111111
1669                    <1> ;;    ------------------------------------------------------------
1670                    <1> ;;    (0 is reserved block, 1s represent free blocks respectively.)
1671                    <1> ;;    (Note: Allocation cell/unit of the table is bit, not byte)
1672                    <1> ;;
1673                    <1> ;;    ............................................................
1674                    <1> ;;
1675                    <1> ;;    'swap_out' procedure checks 'free_swap_blocks' count at first,
1676                    <1> ;;    then it searches Swap Allocation Table if free count is not
1677                    <1> ;;    zero. From begining the [swpd_next] dword value, the first bit
1678                    <1> ;;    position with value of 1 on the table is converted to swap
1679                    <1> ;;    disk/file offset address, in sectors (not 4096 bytes block).
1680                    <1> ;;    'ldrv_write' procedure is called with ldrv (logical drive
1681                    <1> ;;    number of physical swap disk or virtual swap disk)
1682                    <1> ;;    number, sector offset (not absolute sector -LBA- number),
1683                    <1> ;;    and sector count (8, 512*8 = 4096) and buffer adress
1684                    <1> ;;    (memory page). That will be a direct disk write procedure.
1685                    <1> ;;    (for preventing late memory allocation, significant waiting).
1686                    <1> ;;    If disk write procedure returns with error or free count of
1687                    <1> ;;    swap blocks is ZERO, 'swap_out' procedure will return with
1688                    <1> ;;    'insufficient memory error' (cf=1).
1689                    <1> ;;
1690                    <1> ;;    (Note: Even if free swap disk/file blocks was not zero,
1691                    <1> ;;    any disk write error will not be fixed by 'swap_out' procedure,
1692                    <1> ;;    in other words, 'swap_out' will not check the table for other
1693                    <1> ;;    free blocks after a disk write error. It will return to
1694                    <1> ;;    the caller with error (CF=1) which means swapping is failed.
1695                    <1> ;;
1696                    <1> ;;    After writing the page on to swap disk/file address/sector,
1697                    <1> ;;    'swap_out' procesure returns with that swap (offset) sector
1698                    <1> ;;    address (cf=0).
1699                    <1> ;;
1700                    <1> ;;    ............................................................
1701                    <1> ;;
1702                    <1> ;;    'swap_in' procedure loads addressed (relevant) swap disk or
1703                    <1> ;;    file sectors at specified memory page. Then page allocation
1704                    <1> ;;    procedure updates relevant page table entry with 'present'
1705                    <1> ;;    attribute. If swap disk or file reading fails there is nothing
1706                    <1> ;;    to do, except to terminate the process which is the owner of
1707                    <1> ;;    the swapped page.
1708                    <1> ;;
1709                    <1> ;;    'swap_in' procedure sets the relevant/respective bit value
1710                    <1> ;;    in the Swap Allocation Table (as free block). 'swap_in' also
1711                    <1> ;;    updates [swpd_first] pointer if it is required.
1712                    <1> ;;
1713                    <1> ;;    ............................................................
1714                    <1> ;;
1715                    <1> ;;    Note: If [swap_enabled] value is ZERO, that means there is not
1716                    <1> ;;    a swap disk or swap file in use... 'swap_in' and 'swap_out'
1717                    <1> ;;    procedures ans 'swap page que' procedures will not be active...
```

```
1718                            <1> ;;    'Insufficient memory' error will be returned by 'swap_out'
1719                            <1> ;;    and 'general protection fault' will be returned by 'swap_in'
1720                            <1> ;;    procedure, if it is called mistakenly (a wrong value in a PTE).
1721                            <1> ;;
1722                            <1>
1723                            <1> swap_in:
1724                            <1>       ; 31/08/2015
1725                            <1>       ; 20/07/2015
1726                            <1>       ; 28/04/2015
1727                            <1>       ; 18/04/2015
1728                            <1>       ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
1729                            <1>       ;
1730                            <1>       ; INPUT ->
1731                            <1>       ;     EBX = PHYSICAL (real/flat) ADDRESS OF THE MEMORY PAGE
1732                            <1>       ;     EBP = VIRTUAL (LINEAR) ADDRESS (page fault address)
1733                            <1>       ;     EAX = Offset Address for the swapped page on the
1734                            <1>       ;           swap disk or in the swap file.
1735                            <1>       ;
1736                            <1>       ; OUTPUT ->
1737                            <1>       ;     EAX = 0 if loading at memory has been successful
1738                            <1>       ;
1739                            <1>       ;     CF = 1 -> swap disk reading error (disk/file not present
1740                            <1>       ;               or sector not present or drive not ready
1741                            <1>       ;          EAX = Error code
1742                            <1>       ;          [u.error] = EAX
1743                            <1>       ;                  = The last error code for the process
1744                            <1>       ;                    (will be reset after returning to user)
1745                            <1>       ;
1746                            <1>       ; Modified Registers -> EAX
1747                            <1>       ;
1748                            <1>
1749 00006012 833D[62050300]00 <1>         cmp     dword [swp_drv], 0
1750 00006019 7646             <1>       jna     short swpin_dnp_err
1751                            <1>
1752 0000601B 3B05[66050300]   <1>       cmp     eax, [swpd_size]
1753 00006021 734A             <1>       jnb     short swpin_snp_err
1754                            <1>
1755 00006023 56               <1>       push    esi
1756 00006024 53               <1>       push    ebx
1757 00006025 51               <1>       push    ecx
1758 00006026 8B35[62050300]   <1>       mov     esi, [swp_drv]
1759 0000602C B908000000       <1>       mov     ecx, PAGE_SIZE / LOGIC_SECT_SIZE  ; 8 !
1760                            <1>               ; Note: Even if corresponding physical disk's sector
1761                            <1>               ; size different than 512 bytes, logical disk sector
1762                            <1>               ; size is 512 bytes and disk reading procedure
1763                            <1>               ; will be performed for reading 4096 bytes
1764                            <1>               ; (2*2048, 8*512).
1765                            <1>       ; ESI = Logical disk description table address
1766                            <1>       ; EBX = Memory page (buffer) address (physical!)
1767                            <1>       ; EAX = Sector adress (offset address, logical sector number)
1768                            <1>       ; ECX = Sector count ; 8 sectors
1769 00006031 50               <1>       push    eax
1770 00006032 E8AF020000       <1>       call    logical_disk_read
1771 00006037 58               <1>       pop     eax
1772 00006038 730C             <1>       jnc     short swpin_read_ok
1773                            <1>       ;
1774 0000603A B828000000       <1>       mov     eax, SWP_DISK_READ_ERR ; drive not ready or read error
1775 0000603F A3[C8030300]     <1>       mov     [u.error], eax
1776 00006044 EB17             <1>       jmp     short swpin_retn
1777                            <1>       ;
1778                            <1> swpin_read_ok:
1779                            <1>       ; EAX = Offset address (logical sector number)
1780 00006046 E80D020000       <1>       call    unlink_swap_block  ; Deallocate swap block
1781                            <1>       ;
1782                            <1>       ; EBX = Memory page (buffer) address (physical!)
1783                            <1>       ; 20/07/2015
1784 0000604B 89EB             <1>       mov     ebx, ebp ; virtual address (page fault address)
1785 0000604D 6681E300F0       <1>         and     bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
1786 00006052 8A1D[B3030300]   <1>       mov     bl, [u.uno] ; current process number
1787                            <1>       ; EBX = Virtual (Linear) address & process number combination
1788 00006058 E8DB000000       <1>       call    swap_queue_shift
1789                            <1>       ; eax = 0 ; 10/06/2016 (if ebx input > 0, eax output = 0)
1790                            <1>       ;sub   eax, eax  ; 0 ; Error Code = 0  (no error)
1791                            <1>       ; zf = 1
1792                            <1> swpin_retn:
1793 0000605D 59               <1>       pop     ecx
1794 0000605E 5B               <1>       pop     ebx
1795 0000605F 5E               <1>       pop     esi
1796 00006060 C3               <1>       retn
1797                            <1>
1798                            <1> swpin_dnp_err:
1799 00006061 B829000000       <1>       mov     eax, SWP_DISK_NOT_PRESENT_ERR
1800                            <1> swpin_err_retn:
1801 00006066 A3[C8030300]     <1>       mov     [u.error], eax
1802 0000606B F9               <1>       stc
1803 0000606C C3               <1>       retn
1804                            <1>
1805                            <1> swpin_snp_err:
1806 0000606D B82A000000       <1>       mov     eax, SWP_SECTOR_NOT_PRESENT_ERR
1807 00006072 EBF2             <1>       jmp     short swpin_err_retn
1808                            <1>
1809                            <1> swap_out:
1810                            <1>       ; 10/06/2016
1811                            <1>       ; 07/06/2016
1812                            <1>         ; 23/05/2016
1813                            <1>       ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
1814                            <1>       ; 24/10/2014 - 31/08/2015 (Retro UNIX 386 v1)
1815                            <1>       ;
1816                            <1>       ; INPUT ->
1817                            <1>       ;     none
1818                            <1>       ;
1819                            <1>       ; OUTPUT ->
1820                            <1>       ;     EAX = Physical page address (which is swapped out
1821                            <1>       ;           for allocating a new page)
1822                            <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>       ;
00006074 66833D[60050300]01     <1>       cmp    word [swpq_count], 1
0000607C 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>
00006082 833D[6A050300]01       <1>         cmp      dword [swpd_free], 1
00006089 0F828F000000           <1>         jc       swpout_nfspc_err ; 'no free space on swap disk'
                                <1>
0000608F 53                     <1>       push   ebx ; *
                                <1> swpout_1:
                                <1>       ; 10/06/2016
00006090 31DB                   <1>       xor    ebx, ebx ; shift the queue and return a PTE value
00006092 E8A1000000             <1>       call   swap_queue_shift
00006097 21C0                   <1>       and    eax, eax      ; 0 = empty queue (improper entries)
00006099 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
0000609F 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:
000060A3 52                     <1>       push   edx ; **
                                <1>
                                <1>       ; 07/06/2016
000060A4 803D[53960100]00       <1>       cmp    byte [timer_events], 0
000060AB 762F                   <1>       jna    short swpout_2
                                <1>       ;
000060AD 8A15[53960100]         <1>       mov    dl, [timer_events]
                                <1>
000060B3 51                     <1>       push   ecx ; ***
000060B4 53                     <1>       push   ebx ; ****
000060B5 BB[60040300]           <1>       mov    ebx, timer_set ; beginning address of timer event
                                <1>                             ; structures
                                <1> swpout_timer_page_1:
000060BA 8A0B                   <1>       mov    cl, [ebx]
000060BC 08C9                   <1>       or     cl, cl ; 0 = free, >0 = process number
000060BE 7415                   <1>       jz     short swpout_timer_page_3
000060C0 8B4B0C                 <1>       mov    ecx, [ebx+12] ; response (signal return) address
000060C3 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)
000060C8 39C8                   <1>       cmp    eax, ecx
000060CA 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>       ;
000060CC 5B                     <1>       pop    ebx ; ****
000060CD 59                     <1>       pop    ecx ; ***
000060CE 5A                     <1>       pop    edx ; **
000060CF EBBF                   <1>       jmp    short swpout_1      ; do not swap out this page !
                                <1>
                                <1> swpout_timer_page_2:
                                <1>       ; 07/06/2016
000060D1 FECA                   <1>       dec    dl
000060D3 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
000060D5 83C310                 <1>       add    ebx, 16
000060D8 EBE0                   <1>       jmp    short swpout_timer_page_1
                                <1>
                                <1> swpout_timer_page_4:
000060DA 5B                     <1>       pop    ebx ; ****
000060DB 59                     <1>       pop    ecx ; ***
                                <1> swpout_2:
000060DC 89DA                   <1>       mov    edx, ebx            ; Page table entry address
000060DE 89C3                   <1>       mov    ebx, eax            ; Buffer (Page) Address
                                <1>       ;
000060E0 E8A6010000             <1>       call   link_swap_block
000060E5 7304                   <1>       jnc    short swpout_3             ; It may not be needed here
                                <1>                                        ; because [swpd_free] value
                                <1>                                        ; was checked at the beginging.
000060E7 5A                     <1>       pop    edx ; **
000060E8 5B                     <1>       pop    ebx ; *
000060E9 EB33                   <1>       jmp    short swpout_nfspc_err
                                <1> swpout_3:
000060EB A900000080             <1>       test   eax, 80000000h ; test bit 31 (this may not be needed!)
000060F0 752C                   <1>       jnz    short swpout_nfspc_err  ; 10/06/2016 (bit 31 = 1 !)
                                <1>       ;
000060F2 56                     <1>       push   esi ; **
000060F3 51                     <1>       push   ecx ; ***
000060F4 50                     <1>       push   eax ; sector address ; (31 bit !, bit 31 = 0)
000060F5 8B35[62050300]         <1>       mov    esi, [swp_drv]
000060FB B908000000             <1>       mov    ecx, PAGE_SIZE / LOGIC_SECT_SIZE  ; 8 !
                                <1>              ; Note: Even if corresponding physical disk's sector
```

```
1928                                    <1>             ; size different than 512 bytes, logical disk sector
1929                                    <1>             ; size is 512 bytes and disk writing procedure
1930                                    <1>             ; will be performed for writing 4096 bytes
1931                                    <1>             ; (2*2048, 8*512).
1932                                    <1>       ; ESI = Logical disk description table address
1933                                    <1>       ; EBX = Buffer (Page) address
1934                                    <1>       ; EAX = Sector adress (offset address, logical sector number)
1935                                    <1>       ; ECX = Sector count ; 8 sectors
1936                                    <1>       ; edx = PTE address
1937 00006100 E8E2010000                 <1>       call   logical_disk_write
1938                                    <1>       ; edx = PTE address
1939 00006105 59                       <1>       pop    ecx ; sector address
1940 00006106 730C                     <1>       jnc    short swpout_write_ok
1941                                    <1>       ;
1942                                    <1>       ;; call      unlink_swap_block ; this block must be left as 'in use'
1943                                    <1> swpout_dw_err:
1944 00006108 B82C000000               <1>       mov    eax, SWP_DISK_WRITE_ERR ; drive not ready or write error
1945 0000610D A3[C8030300]             <1>       mov    [u.error], eax
1946 00006112 EB06                     <1>       jmp    short swpout_retn
1947                                    <1>       ;
1948                                    <1> swpout_write_ok:
1949                                    <1>       ; EBX = Buffer (page) address
1950                                    <1>       ; EDX = Page Table Entry address
1951                                    <1>       ; ECX = Swap disk sector (file block) address (31 bit)
1952 00006114 D1E1                     <1>       shl    ecx, 1  ; 31 bit sector address from bit 1 to bit 31
1953 00006116 890A                     <1>       mov    [edx], ecx
1954                                    <1>              ; bit 0 = 0 (swapped page)
1955 00006118 89D8                     <1>       mov    eax, ebx
1956                                    <1> swpout_retn:
1957 0000611A 59                       <1>       pop    ecx ; ***
1958 0000611B 5E                       <1>       pop    esi ; **
1959 0000611C 5B                       <1>       pop    ebx ; *
1960 0000611D C3                       <1>       retn
1961                                    <1>
1962                                    <1> ;swpout_dnp_err:
1963                                    <1> ;     mov    eax, SWP_DISK_NOT_PRESENT_ERR ; disk not present
1964                                    <1> ;     jmp    short swpout_err_retn
1965                                    <1> swpout_nfspc_err:
1966 0000611E B82B000000               <1>       mov    eax, SWP_NO_FREE_SPACE_ERR ; no free space
1967                                    <1> swpout_err_retn:
1968 00006123 A3[C8030300]             <1>       mov    [u.error], eax
1969                                    <1>       ;stc
1970 00006128 C3                       <1>       retn
1971                                    <1> swpout_npts_err:
1972 00006129 B82D000000               <1>       mov    eax, SWP_NO_PAGE_TO_SWAP_ERR
1973 0000612E 5B                       <1>       pop    ebx
1974 0000612F EBF2                     <1>       jmp    short swpout_err_retn
1975                                    <1> swpout_im_err:
1976 00006131 B804000000               <1>       mov    eax, ERR_MINOR_IM ; insufficient (out of) memory
1977 00006136 EBEB                     <1>       jmp    short swpout_err_retn
1978                                    <1>
1979                                    <1> swap_queue_shift:
1980                                    <1>       ; 26/03/2017
1981                                    <1>       ; 10/06/2016
1982                                    <1>       ; 09/06/2016 - TRDOS 386 (TRDOS v2.0)
1983                                    <1>       ; 23/10/2014 - 20/07/2015 (Retro UNIX 386 v1)
1984                                    <1>       ;
1985                                    <1>       ; INPUT ->
1986                                    <1>       ;     EBX = Virtual (linear) address (bit 12 to 31)
1987                                    <1>       ;           and process number combination (bit 0 to 11)
1988                                    <1>       ;     EBX = 0 -> shift/drop from the head (offset 0)
1989                                    <1>       ;
1990                                    <1>       ; OUTPUT ->
1991                                    <1>       ;     If EBX input > 0
1992                                    <1>       ;        the queue will be shifted 4 bytes (dword),
1993                                    <1>       ;        from the tail to the head, up to entry offset
1994                                    <1>       ;        which points to EBX input value or nothing
1995                                    <1>       ;        to do if EBX value is not found on the queue.
1996                                    <1>       ;        (The entry -with EBX value- will be removed
1997                                    <1>       ;        from the queue if it is found.)
1998                                    <1>       ;
1999                                    <1>       ;        EAX = 0
2000                                    <1>       ;
2001                                    <1>       ;     If EBX input = 0
2002                                    <1>       ;        the queue will be shifted 4 bytes (dword),
2003                                    <1>       ;        from the tail to the head, if the PTE address
2004                                    <1>       ;        which is pointed in head of the queue is marked
2005                                    <1>       ;        as "accessed" or it is marked as "non present".
2006                                    <1>       ;        (If "accessed" flag of the PTE -which is pointed
2007                                    <1>       ;        in the head- is set -to 1-, it will be reset
2008                                    <1>       ;        -to 0- and then, the queue will be rotated
2009                                    <1>       ;        -without dropping pointer of the PTE from
2010                                    <1>       ;        the queue- for 4 bytes on head to tail direction.
2011                                    <1>       ;        Pointer in the head will be moved into the tail,
2012                                    <1>       ;        other PTEs will be shifted on head direction.)
2013                                    <1>       ;
2014                                    <1>       ;        Swap queue will be shifted up to the first
2015                                    <1>       ;        'present' or 'non accessed' page will be found
2016                                    <1>       ;        (as pointed) on the queue head (then it will be
2017                                    <1>         ;          removed/dropped from the queue).
2018                                    <1>       ;
2019                                    <1>       ;        EAX (> 0) = PTE value of the page which is
2020                                    <1>       ;             (it's pointer -virtual address-) dropped
2021                                    <1>       ;             (removed) from swap queue.
2022                                    <1>       ;        EBX = PTE address of the page (if EAX > 0)
2023                                    <1>       ;              which is (it's pointer -virtual address-)
2024                                    <1>       ;             dropped (removed) from swap queue.
2025                                    <1>       ;
2026                                    <1>       ;        EAX = 0 -> empty swap queue !
2027                                    <1>       ;
2028                                    <1>       ; Modified Registers -> EAX, EBX
2029                                    <1>       ;
2030 00006138 0FB705[60050300]         <1>       movzx   eax, word [swpq_count]  ; Max. 1024
2031 0000613F 6621C0                   <1>       and    ax, ax
2032 00006142 7431                     <1>       jz     short swpqs_retn
```

```
2033 00006144 57                <1>       push   edi
2034 00006145 56                <1>       push   esi
2035 00006146 51                <1>       push   ecx
2036 00006147 BE00E00800        <1>       mov    esi, swap_queue
2037 0000614C 89C1              <1>       mov    ecx, eax
2038 0000614E 09DB              <1>       or     ebx, ebx
2039 00006150 7424              <1>       jz     short swpqs_7
2040                            <1> swpqs_1:
2041 00006152 AD                <1>       lodsd
2042 00006153 39D8              <1>       cmp    eax, ebx
2043 00006155 7406              <1>       je     short swpqs_2
2044 00006157 E2F9              <1>       loop   swpqs_1
2045                            <1>       ; 10/06/2016
2046 00006159 29C0              <1>       sub    eax, eax
2047 0000615B EB15              <1>       jmp    short swpqs_6
2048                            <1> swpqs_2:
2049 0000615D 89F7              <1>       mov    edi, esi
2050 0000615F 83EF04            <1>       sub    edi, 4
2051                            <1> swpqs_3:
2052 00006162 66FF0D[60050300]  <1>       dec    word [swpq_count]
2053 00006169 7403              <1>       jz     short swpqs_5
2054                            <1> swpqs_4:
2055 0000616B 49                <1>       dec    ecx
2056 0000616C F3A5              <1>       rep    movsd  ; shift up (to the head)
2057                            <1> swpqs_5:
2058 0000616E 31C0              <1>       xor    eax, eax
2059 00006170 8907              <1>       mov    [edi], eax
2060                            <1> swpqs_6:
2061 00006172 59                <1>       pop    ecx
2062 00006173 5E                <1>       pop    esi
2063 00006174 5F                <1>       pop    edi
2064                            <1> swpqs_retn:
2065 00006175 C3                <1>       retn
2066                            <1> swpqs_7:
2067 00006176 89F7              <1>       mov    edi, esi ; head
2068 00006178 AD                <1>       lodsd
2069                            <1>       ; 20/07/2015
2070 00006179 89C3              <1>       mov    ebx, eax
2071 0000617B 81E300F0FFFF      <1>       and    ebx, ~PAGE_OFF ; ~0FFFh
2072                            <1>                     ; ebx = virtual address (at page boundary)
2073 00006181 25FF0F0000        <1>       and    eax, PAGE_OFF ; 0FFFh
2074                            <1>                     ; ax = process number (1 to 4095)
2075 00006186 3A05[B3030300]    <1>       cmp    al, [u.uno]
2076                            <1>                     ; Max. 16 (nproc) processes for Retro UNIX 386 v1
2077 0000618C 7507              <1>       jne    short swpqs_8
2078 0000618E A1[B8030300]      <1>       mov    eax, [u.pgdir]
2079 00006193 EB28              <1>       jmp    short swpqs_9
2080                            <1> swpqs_8:
2081                            <1>       ; 09/06/2016
2082 00006195 80B8[AF000300]00  <1>       cmp    byte [eax+p.stat-1], 0
2083 0000619C 76C4              <1>       jna    short swpqs_3     ; free (or terminated) process
2084 0000619E 80B8[AF000300]02  <1>       cmp    byte [eax+p.stat-1], 2 ; waiting
2085 000061A5 77BB              <1>       ja     short swpqs_3         ; zombie (3) or undefined ?
2086                            <1>
2087                            <1>       ;shl   ax, 2
2088 000061A7 C0E002            <1>       shl    al, 2
2089 000061AA 8B80[BC000300]    <1>       mov    eax, [eax+p.upage-4]
2090 000061B0 09C0              <1>       or     eax, eax
2091 000061B2 74AE              <1>       jz     short swpqs_3 ; invalid upage
2092 000061B4 83C05C            <1>       add    eax, u.pgdir - user
2093                            <1>                     ; u.pgdir value for the process
2094                            <1>                     ; is in [eax]
2095 000061B7 8B00              <1>       mov    eax, [eax]
2096 000061B9 21C0              <1>       and    eax, eax
2097 000061BB 74A5              <1>       jz     short swpqs_3 ; invalid page directory
2098                            <1> swpqs_9:
2099 000061BD 52                <1>       push   edx
2100                            <1>       ; eax = page directory
2101                            <1>       ; ebx = virtual address
2102 000061BE E82BFBFFFF        <1>       call   get_pte
2103 000061C3 89D3              <1>       mov    ebx, edx     ; PTE address
2104 000061C5 5A                <1>       pop    edx
2105                            <1>       ; 10/06/2016
2106 000061C6 723A              <1>       jc     short swpqs_13 ; empty PDE
2107                            <1>       ; EAX = PTE value
2108 000061C8 A801              <1>       test   al, PTE_A_PRESENT ; bit 0 = 1
2109 000061CA 7436              <1>       jz     short swpqs_13  ; Drop non-present page
2110                            <1>                              ; from the queue (head)
2111 000061CC A802              <1>       test   al, PTE_A_WRITE     ; bit 1 = 0 (read only)
2112 000061CE 7432              <1>       jz     short swpqs_13  ; Drop read only page
2113                            <1>                              ; from the queue (head)
2114                            <1>       ;test  al, PTE_A_ACCESS ; bit 5 = 1 (Accessed)
2115                            <1>       ;jnz   short swpqs_11  ; present
2116                            <1>                              ; accessed page
2117 000061D0 0FBAF005          <1>         btr     eax, PTE_A_ACCESS_BIT ; reset 'accessed' bit
2118 000061D4 7210              <1>       jc     short swpqs_11  ; accessed page
2119                            <1>
2120 000061D6 49                <1>       dec    ecx
2121 000061D7 66890D[60050300]  <1>       mov    [swpq_count], cx
2122 000061DE 7402              <1>         jz      short swpqs_10
2123                            <1>             ; esi = head + 4
2124                            <1>             ; edi = head
2125 000061E0 F3A5              <1>       rep    movsd   ; n = 1 to k-1, [n - 1] = [n]
2126                            <1> swpqs_10:
2127 000061E2 890F              <1>       mov    [edi], ecx ; 0
2128 000061E4 EB8C              <1>       jmp    short swpqs_6 ; 26/03/2017
2129                            <1>
2130                            <1> swpqs_11:
2131 000061E6 8903              <1>       mov    [ebx], eax     ; save changed attribute
2132                            <1>       ; Rotation (head -> tail)
2133 000061E8 49                <1>       dec    ecx     ; entry count -> last entry number
2134 000061E9 74F7              <1>       jz     short swpqs_10
2135                            <1>             ; esi = head + 4
2136                            <1>             ; edi = head
2137 000061EB 8B07              <1>       mov    eax, [edi] ; 20/07/2015
```

```
2138 000061ED F3A5               <1>       rep    movsd   ; n = 1 to k-1, [n - 1] = [n]
2139 000061EF 8907               <1>       mov    [edi], eax ; head -> tail ; [k] = [1]
2140                             <1>
2141 000061F1 668B0D[60050300]   <1>       mov    cx, [swpq_count]
2142                             <1>
2143                             <1> swpqs_12:
2144 000061F8 BE00E00800         <1>       mov    esi, swap_queue ; head
2145 000061FD E974FFFFFF         <1>         jmp     swpqs_7
2146                             <1>
2147                             <1> swpqs_13:
2148 00006202 49                 <1>       dec    ecx
2149 00006203 66890D[60050300]   <1>       mov    [swpq_count], cx
2150 0000620A 0F845EFFFFFF       <1>         jz      swpqs_5
2151 00006210 EBE6               <1>       jmp    short swpqs_12
2152                             <1>
2153                             <1> add_to_swap_queue:
2154                             <1> ; temporary - 16/09/2015
2155 00006212 C3                 <1> retn
2156                             <1>       ; 20/02/2017
2157                             <1>       ; 20/07/2015
2158                             <1>       ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
2159                             <1>       ;
2160                             <1>       ; Adds new page to swap queue
2161                             <1>       ; (page directories and page tables must not be added
2162                             <1>       ; to swap queue)
2163                             <1>       ;
2164                             <1>       ; INPUT ->
2165                             <1>       ;      EBX = Linear (Virtual) addr for current process
2166                             <1>       ;      [u.uno]
2167                             <1>       ;      20/02/2017
2168                             <1>       ;      (Linear address = CORE + user's virtual address)
2169                             <1>       ;
2170                             <1>       ; OUTPUT ->
2171                             <1>       ;      EAX = [swpq_count]
2172                             <1>       ;            (after the PTE has been added)
2173                             <1>       ;      EAX = 0 -> Swap queue is full, (1024 entries)
2174                             <1>       ;            the PTE could not be added.
2175                             <1>       ;
2176                             <1>       ; Modified Registers -> EAX
2177                             <1>       ;
2178 00006213 53                 <1>       push   ebx
2179 00006214 6681E300F0         <1>         and     bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
2180 00006219 8A1D[B3030300]     <1>       mov    bl, [u.uno] ; current process number
2181 0000621F E814FFFFFF         <1>       call   swap_queue_shift ; drop from the queue if
2182                             <1>                               ; it is already on the queue
2183                             <1>             ; then add it to the tail of the queue
2184 00006224 0FB705[60050300]   <1>       movzx  eax, word [swpq_count]
2185 0000622B 663D0004           <1>       cmp    ax, 1024
2186 0000622F 7205               <1>       jb     short atsq_1
2187 00006231 6629C0             <1>       sub    ax, ax
2188 00006234 5B                 <1>       pop    ebx
2189 00006235 C3                 <1>       retn
2190                             <1> atsq_1:
2191 00006236 56                 <1>       push   esi
2192 00006237 BE00E00800         <1>       mov    esi, swap_queue
2193 0000623C 6621C0             <1>       and    ax, ax
2194 0000623F 740A               <1>       jz     short atsq_2
2195 00006241 66C1E002           <1>       shl    ax, 2  ; convert to offset
2196 00006245 01C6               <1>       add    esi, eax
2197 00006247 66C1E802           <1>       shr    ax, 2
2198                             <1> atsq_2:
2199 0000624B 6640               <1>       inc    ax
2200 0000624D 891E               <1>       mov    [esi], ebx ; Virtual address + [u.uno] combination
2201 0000624F 66A3[60050300]     <1>       mov    [swpq_count], ax
2202 00006255 5E                 <1>       pop    esi
2203 00006256 5B                 <1>       pop    ebx
2204 00006257 C3                 <1>       retn
2205                             <1>
2206                             <1> unlink_swap_block:
2207                             <1>       ; 15/09/2015
2208                             <1>       ; 30/04/2015
2209                             <1>       ; 18/04/2015
2210                             <1>       ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
2211                             <1>       ;
2212                             <1>       ; INPUT ->
2213                             <1>       ;      EAX = swap disk/file offset address
2214                             <1>       ;            (bit 1 to bit 31)
2215                             <1>       ; OUTPUT ->
2216                             <1>       ;      [swpd_free] is increased
2217                             <1>       ;      (corresponding SWAP DISK ALLOC. TABLE bit is SET)
2218                             <1>       ;
2219                             <1>       ; Modified Registers -> EAX
2220                             <1>       ;
2221 00006258 53                 <1>       push   ebx
2222 00006259 52                 <1>       push   edx
2223                             <1>       ;
2224 0000625A C1E804             <1>       shr    eax, SECTOR_SHIFT+1  ;3+1 ; shift sector address to
2225                             <1>                                   ; 3 bits right
2226                             <1>                                   ; to get swap block/page number
2227 0000625D 89C2               <1>       mov    edx, eax
2228                             <1>       ; 15/09/2015
2229 0000625F C1EA03             <1>       shr    edx, 3            ; to get offset to S.A.T.
2230                             <1>                                   ; (1 allocation bit = 1 page)
2231                             <1>                                   ; (1 allocation bytes = 8 pages)
2232 00006262 80E2FC             <1>       and    dl, 0FCh          ; clear lower 2 bits
2233                             <1>                                   ; (to get 32 bit position)
2234                             <1>       ;
2235 00006265 BB00000D00         <1>       mov    ebx, swap_alloc_table ; Swap Allocation Table address
2236 0000626A 01D3               <1>       add    ebx, edx
2237 0000626C 83E01F             <1>       and    eax, 1Fh          ; lower 5 bits only
2238                             <1>                                   ; (allocation bit position)
2239 0000626F 3B05[6E050300]     <1>       cmp    eax, [swpd_next]     ; is the new free block addr. lower
2240                             <1>                                   ; than the address in 'swpd_next' ?
2241                             <1>                                   ; (next/first free block value)
2242 00006275 7305               <1>       jnb    short uswpbl_1           ; no
```

```
2243 00006277 A3[6E050300]        <1>       mov   [swpd_next], eax     ; yes
2244                              <1> uswpbl_1:
2245 0000627C 0FAB03              <1>       bts   [ebx], eax        ; unlink/release/deallocate block
2246                              <1>                               ; set relevant bit to 1.
2247                              <1>                               ; set CF to the previous bit value
2248 0000627F F5                  <1>       cmc                     ; complement carry flag
2249 00006280 7206                <1>       jc    short uswpbl_2          ; do not increase swfd_free count
2250                              <1>                               ; if the block is already deallocated
2251                              <1>                               ; before.
2252 00006282 FF05[6A050300]      <1>         inc     dword [swpd_free]
2253                              <1> uswpbl_2:
2254 00006288 5A                  <1>       pop   edx
2255 00006289 5B                  <1>       pop   ebx
2256 0000628A C3                  <1>       retn
2257                              <1>
2258                              <1> link_swap_block:
2259                              <1>       ; 01/07/2015
2260                              <1>       ; 18/04/2015
2261                              <1>       ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
2262                              <1>       ;
2263                              <1>       ; INPUT -> none
2264                              <1>       ;
2265                              <1>       ; OUTPUT ->
2266                              <1>       ;     EAX = OFFSET ADDRESS OF THE ALLOCATED BLOCK (4096 bytes)
2267                              <1>       ;           in sectors (corresponding
2268                              <1>       ;           SWAP DISK ALLOCATION TABLE bit is RESET)
2269                              <1>       ;
2270                              <1>       ;     CF = 1 and EAX = 0
2271                              <1>       ;               if there is not a free block to be allocated
2272                              <1>       ;
2273                              <1>       ; Modified Registers -> none (except EAX)
2274                              <1>       ;
2275                              <1>
2276                              <1>       ;mov   eax, [swpd_free]
2277                              <1>       ;and   eax, eax
2278                              <1>       ;jz    short out_of_swpspc
2279                              <1>       ;
2280 0000628B 53                  <1>       push  ebx
2281 0000628C 51                  <1>       push  ecx
2282                              <1>       ;
2283 0000628D BB00000D00          <1>       mov   ebx, swap_alloc_table ; Swap Allocation Table offset
2284 00006292 89D9                <1>       mov   ecx, ebx
2285 00006294 031D[6E050300]      <1>       add   ebx, [swpd_next] ; Free block searching starts from here
2286                              <1>                              ; next_free_swap_block >> 5
2287 0000629A 030D[72050300]      <1>       add   ecx, [swpd_last] ; Free block searching ends here
2288                              <1>                              ; (total_swap_blocks - 1) >> 5
2289                              <1> lswbl_scan:
2290 000062A0 39CB                <1>       cmp   ebx, ecx
2291 000062A2 770A                <1>       ja    short lswbl_notfound
2292                              <1>       ;
2293 000062A4 0FBC03              <1>       bsf   eax, [ebx] ; Scans source operand for first bit set (1).
2294                              <1>                        ; Clears ZF if a bit is found set (1) and
2295                              <1>                        ; loads the destination with an index to
2296                              <1>                        ; first set bit. (0 -> 31)
2297                              <1>                        ; Sets ZF to 1 if no bits are found set.
2298                              <1>       ; 01/07/2015
2299 000062A7 751C                <1>       jnz   short lswbl_found ; ZF = 0 -> a free block has been found
2300                              <1>                       ;
2301                              <1>                       ; NOTE:  a Swap Disk Allocation Table bit
2302                              <1>                       ;       with value of 1 means
2303                              <1>                       ;       the corresponding page is free
2304                              <1>                       ;       (Retro UNIX 386 v1 feaure only!)
2305 000062A9 83C304              <1>       add   ebx, 4
2306                              <1>                       ; We return back for searching next page block
2307                              <1>                       ; NOTE: [swpd_free] is not ZERO; so,
2308                              <1>                       ;     we always will find at least 1 free block here.
2309 000062AC EBF2                <1>       jmp          short lswbl_scan
2310                              <1>       ;
2311                              <1> lswbl_notfound:
2312 000062AE 81E900000D00        <1>       sub   ecx, swap_alloc_table
2313 000062B4 890D[6E050300]      <1>       mov   [swpd_next], ecx ; next/first free page = last page
2314                              <1>                              ; (unlink_swap_block procedure will change it)
2315 000062BA 31C0                <1>       xor   eax, eax
2316 000062BC A3[6A050300]        <1>       mov   [swpd_free], eax
2317 000062C1 F9                  <1>       stc
2318                              <1> lswbl_ok:
2319 000062C2 59                  <1>       pop   ecx
2320 000062C3 5B                  <1>       pop   ebx
2321 000062C4 C3                  <1>       retn
2322                              <1>       ;
2323                              <1> ;out_of_swpspc:
2324                              <1> ;     stc
2325                              <1> ;     retn
2326                              <1>
2327                              <1> lswbl_found:
2328 000062C5 89D9                <1>       mov   ecx, ebx
2329 000062C7 81E900000D00        <1>       sub   ecx, swap_alloc_table
2330 000062CD 890D[6E050300]      <1>       mov   [swpd_next], ecx ; Set first free block searching start
2331                              <1>                              ; address/offset (to the next)
2332 000062D3 FF0D[6A050300]      <1>         dec     dword [swpd_free] ; 1 block has been allocated (X = X-1)
2333                              <1>       ;
2334 000062D9 0FB303              <1>       btr   [ebx], eax    ; The destination bit indexed by the source value
2335                              <1>                           ; is copied into the Carry Flag and then cleared
2336                              <1>                           ; in the destination.
2337                              <1>                           ;
2338                              <1>                           ; Reset the bit which is corresponding to the
2339                              <1>                           ; (just) allocated block.
2340 000062DC C1E105              <1>       shl   ecx, 5        ; (block offset * 32) + block index
2341 000062DF 01C8                <1>       add   eax, ecx      ; = block number
2342 000062E1 C1E003              <1>       shl   eax, SECTOR_SHIFT ; 3, sector (offset) address of the block
2343                              <1>                           ; 1 block =  8 sectors
2344                              <1>       ;
2345                              <1>       ; EAX = offset address of swap disk/file sector (beginning of the block)
2346                              <1>       ;
2347                              <1>       ; NOTE: The relevant page table entry will be updated
```

```
2348                                    <1>       ;       according to this EAX value...
2349                                    <1>       ;
2350 000062E4 EBDC                     <1>       jmp    short lswbl_ok
2351                                    <1>
2352                                    <1> logical_disk_read:
2353                                    <1>       ; 20/07/2015
2354                                    <1>       ; 09/03/2015 (temporary code here)
2355                                    <1>       ;
2356                                    <1>       ; INPUT ->
2357                                    <1>       ;      ESI = Logical disk description table address
2358                                    <1>       ;      EBX = Memory page (buffer) address (physical!)
2359                                    <1>       ;      EAX = Sector adress (offset address, logical sector number)
2360                                    <1>       ;      ECX = Sector count
2361                                    <1>       ;
2362                                    <1>       ;
2363 000062E6 C3                     <1>       retn
2364                                    <1>
2365                                    <1> logical_disk_write:
2366                                    <1>       ; 20/07/2015
2367                                    <1>       ; 09/03/2015 (temporary code here)
2368                                    <1>       ;
2369                                    <1>       ; INPUT ->
2370                                    <1>       ;      ESI = Logical disk description table address
2371                                    <1>       ;      EBX = Memory page (buffer) address (physical!)
2372                                    <1>       ;      EAX = Sector adress (offset address, logical sector number)
2373                                    <1>       ;      ECX = Sector count
2374                                    <1>       ;
2375 000062E7 C3                     <1>       retn
2376                                    <1>
2377                                    <1> get_physical_addr:
2378                                    <1>       ; 26/03/2017
2379                                    <1>       ; 20/02/2017
2380                                    <1>       ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)
2381                                    <1>       ; 18/10/2015
2382                                    <1>       ; 29/07/2015
2383                                    <1>       ; 20/07/2015
2384                                    <1>       ; 04/06/2015
2385                                    <1>       ; 20/05/2015
2386                                    <1>       ; 28/04/2015
2387                                    <1>       ; 18/04/2015
2388                                    <1>       ; Get physical address
2389                                    <1>       ;     (allocates a new page for user if it is not present)
2390                                    <1>       ;
2391                                    <1>       ; (This subroutine is needed for mapping user's virtual
2392                                    <1>       ; (buffer) address to physical address (of the buffer).)
2393                                    <1>       ; ('sys write', 'sys read' system calls...)
2394                                    <1>       ;
2395                                    <1>       ; INPUT ->
2396                                    <1>       ;     EBX = virtual address
2397                                    <1>       ;     u.pgdir = page directory (physical) address
2398                                    <1>       ;
2399                                    <1>       ; OUTPUT ->
2400                                    <1>       ;     EAX = physical address
2401                                    <1>       ;     EBX = linear address
2402                                    <1>       ;     EDX = physical address of the page frame
2403                                    <1>       ;           (with attribute bits)
2404                                    <1>       ;     ECX = byte count within the page frame
2405                                    <1>       ;
2406                                    <1>       ; Modified Registers -> EAX, EBX, ECX, EDX
2407                                    <1>       ;
2408 000062E8 81C300004000             <1>       add    ebx, CORE ; 18/10/2015
2409                                    <1> get_physical_addr_x: ; 27/05/2016
2410 000062EE A1[B8030300]             <1>       mov    eax, [u.pgdir]
2411 000062F3 E8F6F9FFFF               <1>       call   get_pte
2412                                    <1>              ; EDX = Page table entry address (if CF=0)
2413                                    <1>              ;       Page directory entry address (if CF=1)
2414                                    <1>              ;       (Bit 0 value is 0 if PT is not present)
2415                                    <1>              ; EAX = Page table entry value (page address)
2416                                    <1>              ;     CF = 1 -> PDE not present or invalid ?
2417 000062F8 731C                     <1>       jnc    short gpa_1
2418                                    <1>       ;
2419 000062FA E8D4F8FFFF               <1>       call   allocate_page
2420 000062FF 7248                     <1>       jc     short gpa_im_err  ; 'insufficient memory' error
2421                                    <1> gpa_0:
2422 00006301 E847F9FFFF               <1>       call   clear_page
2423                                    <1>       ; EAX = Physical (base) address of the allocated (new) page
2424 00006306 0C07                     <1>       or     al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER ; 4+2+1 = 7
2425                                    <1>                        ; lower 3 bits are used as U/S, R/W, P flags
2426                                    <1>                        ; (user, writable, present page)
2427 00006308 8902                     <1>       mov    [edx], eax ; Let's put the new page directory entry here !
2428 0000630A A1[B8030300]             <1>       mov    eax, [u.pgdir]
2429 0000630F E8DAF9FFFF               <1>       call   get_pte
2430 00006314 7233                     <1>       jc     short gpa_im_err ; 'insufficient memory' error
2431                                    <1> gpa_1:
2432                                    <1>       ; EAX = PTE value, EDX = PTE address
2433 00006316 A801                     <1>       test   al, PTE_A_PRESENT
2434 00006318 751F                     <1>       jnz    short gpa_3 ; 26/03/2017
2435 0000631A 09C0                     <1>       or     eax, eax
2436 0000631C 7456                     <1>       jz     short gpa_7  ; Allocate a new page
2437                                    <1>       ; 20/07/2015
2438 0000631E 55                       <1>       push   ebp
2439 0000631F 89DD                     <1>       mov    ebp, ebx ; virtual (linear) address
2440                                    <1>       ; reload swapped page
2441 00006321 E878000000               <1>       call   reload_page ; 28/04/2015
2442 00006326 5D                       <1>       pop    ebp
2443 00006327 724A                     <1>       jc     short gpa_retn
2444                                    <1> gpa_2:
2445                                    <1>       ; 26/03/2017
2446                                    <1>       ; 20/02/2017
2447                                    <1>       ; If a page will contain a Signal Response Byte
2448                                    <1>       ; it must not be swapped out, because
2449                                    <1>       ; timer service or irq callback service
2450                                    <1>       ; will write a signal return/response byte
2451                                    <1>       ; directly by using physical address of Signal
2452                                    <1>       ; Response Byte.(Even if process is not running,
```

```
2453                            <1>       ; or it is running with swapped out pages.)
2454                            <1>       ;
2455                            <1>       ; 'no_page_swap' will be set by 'systimer' or
2456                            <1>       ; 'syscalbac' sistem functions/calls. (*)
2457                            <1>       ;
2458 00006329 803D[929B0100]00   <1>       cmp    byte [no_page_swap], 0
2459 00006330 761D             <1>       jna    short gpa_4 ; this page can be swapped out
2460                            <1>       ; this page must not be swapped out
2461                            <1>       ; but 'no_page_swap' must be reset here
2462                            <1>       ; imediately for other callers (*)
2463                            <1>       ; (otherwise, swap queue would not be long enough)
2464 00006332 E84B000000         <1>       call   gpa_8 ; 26/03/2017
2465 00006337 EB1D             <1>       jmp    short gpa_5
2466                            <1> gpa_3:
2467                            <1>       ; 26/03/2017
2468 00006339 803D[929B0100]00   <1>       cmp    byte [no_page_swap], 0
2469 00006340 7618             <1>       jna    short gpa_6 ; this page can be swapped out
2470 00006342 E83B000000         <1>       call   gpa_8
2471 00006347 EB11             <1>       jmp    short gpa_6
2472                            <1>
2473                            <1> gpa_im_err:
2474 00006349 B804000000         <1>       mov    eax, ERR_MINOR_IM ; Insufficient memory (minor) error!
2475                            <1>                                ; Major error = 0 (No protection fault)
2476 0000634E C3               <1>       retn
2477                            <1> gpa_4:
2478                            <1>       ; 20/07/2015
2479                            <1>       ; 20/05/2015
2480                            <1>       ; add this page to swap queue
2481 0000634F 50               <1>       push   eax
2482                            <1>       ; EBX = Linear (CORE+virtual) address ; 20/02/2017
2483 00006350 E8BDFEFFFF         <1>       call   add_to_swap_queue
2484 00006355 58               <1>       pop    eax
2485                            <1> gpa_5:
2486                            <1>             ; PTE address in EDX
2487                            <1>             ; virtual address in EBX
2488                            <1>       ; EAX = memory page address
2489 00006356 0C07             <1>       or     al, PTE_A_PRESENT + PTE_A_USER + PTE_A_WRITE
2490                            <1>                               ; present flag, bit 0 = 1
2491                            <1>                               ; user flag, bit 2 = 1
2492                            <1>                               ; writable flag, bit 1 = 1
2493 00006358 8902             <1>       mov    [edx], eax  ; Update PTE value
2494                            <1> gpa_6:
2495                            <1>       ; 18/10/2015
2496 0000635A 89D9             <1>       mov    ecx, ebx
2497 0000635C 81E1FF0F0000       <1>       and    ecx, PAGE_OFF
2498 00006362 89C2             <1>       mov    edx, eax
2499 00006364 662500F0           <1>       and    ax, PTE_A_CLEAR
2500 00006368 01C8             <1>       add    eax, ecx
2501 0000636A F7D9             <1>       neg    ecx ; 1 -> -1 (0FFFFFFFFh), 4095 (0FFFh) -> -4095
2502 0000636C 81C100100000       <1>       add    ecx, PAGE_SIZE
2503 00006372 F8               <1>       clc
2504                            <1> gpa_retn:
2505 00006373 C3               <1>       retn
2506                            <1> gpa_7:
2507 00006374 E85AF8FFFF         <1>       call   allocate_page
2508 00006379 72CE             <1>       jc     short gpa_im_err ; 'insufficient memory' error
2509 0000637B E8CDF8FFFF         <1>       call   clear_page
2510 00006380 EBA7             <1>       jmp    short gpa_2
2511                            <1>
2512                            <1> gpa_8: ; 26/03/2017
2513 00006382 C605[929B0100]00   <1>       mov    byte [no_page_swap], 0
2514 00006389 53               <1>       push   ebx
2515 0000638A 50               <1>       push   eax ; 26/03/2017
2516 0000638B 6681E300F0         <1>         and     bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
2517 00006390 8A1D[B3030300]     <1>       mov    bl, [u.uno] ; current process number
2518 00006396 E89DFDFFFF         <1>       call   swap_queue_shift ; drop from the queue if
2519                            <1>                               ; it is already on the queue
2520 0000639B 58               <1>       pop    eax ; 26/03/2017
2521 0000639C 5B               <1>       pop    ebx
2522 0000639D C3               <1>       retn
2523                            <1>
2524                            <1> reload_page:
2525                            <1>       ; 20/07/2015
2526                            <1>       ; 28/04/2015 (Retro UNIX 386 v1 - beginning)
2527                            <1>       ;
2528                            <1>       ; Reload (Restore) swapped page at memory
2529                            <1>       ;
2530                            <1>       ; INPUT ->
2531                            <1>       ;     EBP = Virtual (linear) memory address
2532                            <1>       ;     EAX = PTE value (swap disk sector address)
2533                            <1>       ;     (Swap disk sector address = bit 1 to bit 31 of EAX)
2534                            <1>       ; OUTPUT ->
2535                            <1>       ;     EAX = PHYSICAL (real/flat) ADDRESS OF RELOADED PAGE
2536                            <1>       ;
2537                            <1>       ;     CF = 1 and EAX = error code
2538                            <1>       ;
2539                            <1>       ; Modified Registers -> none (except EAX)
2540                            <1>       ;
2541 0000639E D1E8             <1>       shr    eax, 1   ; Convert PTE value to swap disk address
2542 000063A0 53               <1>       push   ebx      ;
2543 000063A1 89C3             <1>       mov    ebx, eax ; Swap disk (offset) address
2544 000063A3 E82BF8FFFF         <1>       call   allocate_page
2545 000063A8 720C             <1>       jc     short rlp_im_err
2546 000063AA 93               <1>       xchg   eax, ebx
2547                            <1>       ; EBX = Physical memory (page) address
2548                            <1>       ; EAX = Swap disk (offset) address
2549                            <1>       ; EBP = Virtual (linear) memory address
2550 000063AB E862FCFFFF         <1>       call   swap_in
2551 000063B0 720B             <1>       jc     short rlp_swp_err  ; (swap disk/file read error)
2552 000063B2 89D8             <1>       mov    eax, ebx
2553                            <1> rlp_retn:
2554 000063B4 5B               <1>       pop    ebx
2555 000063B5 C3               <1>       retn
2556                            <1>
2557                            <1> rlp_im_err:
```

```
2558 000063B6 B804000000          <1>       mov    eax, ERR_MINOR_IM ; Insufficient memory (minor) error!
2559                              <1>                                ; Major error = 0 (No protection fault)
2560 000063BB EBF7                <1>       jmp    short rlp_retn
2561                              <1>
2562                              <1> rlp_swp_err:
2563 000063BD B828000000          <1>       mov    eax, SWP_DISK_READ_ERR ; Swap disk read error !
2564 000063C2 EBF0                <1>       jmp    short rlp_retn
2565                              <1>
2566                              <1>
2567                              <1> copy_page_dir:
2568                              <1>       ; 19/09/2015
2569                              <1>       ; temporary - 07/09/2015
2570                              <1>       ; 07/09/2015 (Retro UNIX 386 v1 - beginning)
2571                              <1>       ;
2572                              <1>       ; INPUT ->
2573                              <1>       ;      [u.pgdir] = PHYSICAL (real/flat) ADDRESS of the parent's
2574                              <1>       ;                  page directory.
2575                              <1>       ; OUTPUT ->
2576                              <1>       ;      EAX =  PHYSICAL (real/flat) ADDRESS of the child's
2577                              <1>       ;             page directory.
2578                              <1>       ;      (New page directory with new page table entries.)
2579                              <1>       ;      (New page tables with read only copies of the parent's
2580                              <1>       ;      pages.)
2581                              <1>       ;      EAX = 0 -> Error (CF = 1)
2582                              <1>       ;
2583                              <1>       ; Modified Registers -> none (except EAX)
2584                              <1>       ;
2585 000063C4 E80AF8FFFF          <1>       call   allocate_page
2586 000063C9 723E                <1>       jc     short cpd_err
2587                              <1>       ;
2588 000063CB 55                  <1>       push   ebp ; 20/07/2015
2589 000063CC 56                  <1>       push   esi
2590 000063CD 57                  <1>       push   edi
2591 000063CE 53                  <1>       push   ebx
2592 000063CF 51                  <1>       push   ecx
2593 000063D0 8B35[B8030300]      <1>       mov    esi, [u.pgdir]
2594 000063D6 89C7                <1>       mov    edi, eax
2595 000063D8 50                  <1>       push   eax ; save child's page directory address
2596                              <1>       ; copy PDE 0 from the parent's page dir to the child's page dir
2597                              <1>       ; (use same system space for all user page tables)
2598 000063D9 A5                  <1>       movsd
2599 000063DA BD00004000          <1>       mov    ebp, 1024*4096 ; pass the 1st 4MB (system space)
2600 000063DF B9FF030000          <1>       mov    ecx, (PAGE_SIZE / 4) - 1 ; 1023
2601                              <1> cpd_0:
2602 000063E4 AD                  <1>       lodsd
2603                              <1>       ;or    eax, eax
2604                              <1>         ;jnz     short cpd_1
2605 000063E5 A801                <1>       test   al, PDE_A_PRESENT ;  bit 0 =  1
2606 000063E7 7508                <1>       jnz    short cpd_1
2607                              <1>       ; (virtual address at the end of the page table)
2608 000063E9 81C500004000        <1>       add    ebp, 1024*4096 ; page size * PTE count
2609 000063EF EB0F                <1>       jmp    short cpd_2
2610                              <1> cpd_1:
2611 000063F1 662500F0            <1>       and    ax, PDE_A_CLEAR ; 0F000h ; clear attribute bits
2612 000063F5 89C3                <1>       mov    ebx, eax
2613                              <1>       ; EBX = Parent's page table address
2614 000063F7 E81F000000          <1>       call   copy_page_table
2615 000063FC 720C                <1>       jc     short cpd_p_err
2616                              <1>       ; EAX = Child's page table address
2617 000063FE 0C07                <1>       or     al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
2618                              <1>                      ; set bit 0, bit 1 and bit 2 to 1
2619                              <1>                      ; (present, writable, user)
2620                              <1> cpd_2:
2621 00006400 AB                  <1>       stosd
2622 00006401 E2E1                <1>       loop   cpd_0
2623                              <1>       ;
2624 00006403 58                  <1>       pop    eax  ; restore child's page directory address
2625                              <1> cpd_3:
2626 00006404 59                  <1>       pop    ecx
2627 00006405 5B                  <1>       pop    ebx
2628 00006406 5F                  <1>       pop    edi
2629 00006407 5E                  <1>       pop    esi
2630 00006408 5D                  <1>       pop    ebp
2631                              <1> cpd_err:
2632 00006409 C3                  <1>       retn
2633                              <1> cpd_p_err:
2634                              <1>       ; release the allocated pages missing (recover free space)
2635 0000640A 58                  <1>       pop    eax  ; the new page directory address (physical)
2636 0000640B 8B1D[B8030300]      <1>       mov    ebx, [u.pgdir] ; parent's page directory address
2637 00006411 E8F6F8FFFF          <1>       call   deallocate_page_dir
2638 00006416 29C0                <1>       sub    eax, eax ; 0
2639 00006418 F9                  <1>       stc
2640 00006419 EBE9                <1>       jmp    short cpd_3
2641                              <1>
2642                              <1> copy_page_table:
2643                              <1>       ; 19/09/2015
2644                              <1>       ; temporary - 07/09/2015
2645                              <1>       ; 07/09/2015 (Retro UNIX 386 v1 - beginning)
2646                              <1>       ;
2647                              <1>       ; INPUT ->
2648                              <1>       ;      EBX = PHYSICAL (real/flat) ADDRESS of the parent's page table.
2649                              <1>       ;      EBP = page table entry index (from 'copy_page_dir')
2650                              <1>       ; OUTPUT ->
2651                              <1>       ;      EAX = PHYSICAL (real/flat) ADDRESS of the child's page table.
2652                              <1>       ;      EBP = (recent) page table index (for 'add_to_swap_queue')
2653                              <1>       ;      CF = 1 -> error
2654                              <1>       ;
2655                              <1>       ; Modified Registers -> EBP (except EAX)
2656                              <1>       ;
2657 0000641B E8B3F7FFFF          <1>       call   allocate_page
2658 00006420 725A                <1>       jc     short cpt_err
2659                              <1>       ;
2660 00006422 50                  <1>       push   eax ; *
2661                              <1>       ;push  ebx
2662 00006423 56                  <1>       push   esi
```

```
2663 00006424 57                <1>       push   edi
2664 00006425 52                <1>       push   edx
2665 00006426 51                <1>       push   ecx
2666                            <1>       ;
2667 00006427 89DE              <1>       mov    esi, ebx
2668 00006429 89C7              <1>       mov    edi, eax
2669 0000642B 89C2              <1>       mov    edx, eax
2670 0000642D 81C200100000      <1>       add    edx, PAGE_SIZE
2671                            <1> cpt_0:
2672 00006433 AD                <1>       lodsd
2673 00006434 A801              <1>       test   al, PTE_A_PRESENT ;  bit 0 =  1
2674 00006436 750B              <1>       jnz    short cpt_1
2675 00006438 21C0              <1>       and    eax, eax
2676 0000643A 7430              <1>       jz     short cpt_2
2677                            <1>       ; ebp = virtual (linear) address of the memory page
2678 0000643C E85DFFFFFF        <1>       call   reload_page ; 28/04/2015
2679 00006441 7234              <1>       jc     short cpt_p_err
2680                            <1> cpt_1:
2681 00006443 662500F0          <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
2682 00006447 89C1              <1>       mov    ecx, eax
2683                            <1>       ; Allocate a new page for the child process
2684 00006449 E885F7FFFF        <1>       call   allocate_page
2685 0000644E 7227              <1>       jc     short cpt_p_err
2686 00006450 57                <1>       push   edi
2687 00006451 56                <1>       push   esi
2688 00006452 89CE              <1>       mov    esi, ecx
2689 00006454 89C7              <1>       mov    edi, eax
2690 00006456 B900040000        <1>       mov    ecx, PAGE_SIZE/4
2691 0000645B F3A5              <1>       rep    movsd  ; copy page (4096 bytes)
2692 0000645D 5E                <1>       pop    esi
2693 0000645E 5F                <1>       pop    edi
2694                            <1>       ;
2695 0000645F 53                <1>       push   ebx
2696 00006460 50                <1>       push   eax
2697 00006461 89EB              <1>       mov    ebx, ebp
2698                            <1>       ; ebx = virtual address of the memory page
2699 00006463 E8AAFDFFFF        <1>       call   add_to_swap_queue
2700 00006468 58                <1>       pop    eax
2701 00006469 5B                <1>       pop    ebx
2702                            <1>       ;
2703                            <1>       ;or    ax, PTE_A_USER+PTE_A_PRESENT
2704 0000646A 0C07              <1>       or     al, PTE_A_USER+PTE_A_WRITE+PTE_A_PRESENT
2705                            <1> cpt_2:
2706 0000646C AB                <1>       stosd  ; EDI points to child's PTE
2707                            <1>       ;
2708 0000646D 81C500100000      <1>       add    ebp, 4096 ; 20/07/2015 (next page)
2709                            <1>       ;
2710 00006473 39D7              <1>       cmp    edi, edx
2711 00006475 72BC              <1>       jb     short cpt_0
2712                            <1> cpt_p_err:
2713 00006477 59                <1>       pop    ecx
2714 00006478 5A                <1>       pop    edx
2715 00006479 5F                <1>       pop    edi
2716 0000647A 5E                <1>       pop    esi
2717                            <1>       ;pop   ebx
2718 0000647B 58                <1>       pop    eax ; *
2719                            <1> cpt_err:
2720 0000647C C3                <1>       retn
2721                            <1>
2722                            <1> allocate_memory_block:
2723                            <1>       ; 01/05/2017
2724                            <1>       ; 28/04/2017
2725                            <1>       ; 25/04/2017
2726                            <1>       ; 01/04/2016, 02/04/2016, 03/04/2016
2727                            <1>       ; 13/03/2016, 14/03/2016
2728                            <1>       ; 12/03/2016 (TRDOS 386 = TRDOS v2.0)
2729                            <1>       ; Allocating contiguous memory pages (in the kernel's memory space)
2730                            <1>       ;
2731                            <1>       ; INPUT ->
2732                            <1>       ;      EAX = Beginning address (physical)
2733                            <1>       ;      EAX = 0 -> Allocate memory block from the first proper aperture
2734                            <1>       ;      ECX = Number of bytes to be allocated
2735                            <1>       ;
2736                            <1>       ; OUTPUT ->
2737                            <1>       ;      1) cf = 0 -> successful
2738                            <1>       ;      EAX = Beginning (physical) address of the allocated memory block
2739                            <1>       ;      ECX = Number of allocated bytes (rounded up to page borders)
2740                            <1>       ;      2) cf = 1 -> unsuccessful
2741                            <1>       ;       2.1) If EAX > 0 ->
2742                            <1>       ;            (Number of requested pages is more than # of free pages
2743                            <1>       ;             but contiguous free pages -the aperture- is not enough!)
2744                            <1>       ;            EAX = Beginning address of available aperture
2745                            <1>       ;                 (one of all aperture with max. aperture size/length)
2746                            <1>       ;            ECX = Size of available aperture (memory block) in bytes
2747                            <1>       ;       2.2) If EAX = 0 -> Out of memory error
2748                            <1>       ;                  (number of free pages is less than requested number)
2749                            <1>       ;            ECX = Total number of free bytes (free pages * 4096)
2750                            <1>       ;                 (It is not number of contiguous free bytes)
2751                            <1>       ;
2752                            <1>       ; (Modified Registers -> EAX, ECX)
2753                            <1>       ;
2754                            <1>       ; PURPOSE: Loading a file at memory for copying or running etc.
2755                            <1>       ; If this procedure returns with cf is set, ECX contains maximum
2756                            <1>       ; available space and EAX contains the beginning address of it.
2757                            <1>       ; If EAX has zero, ECX contains total number of free bytes.
2758                            <1>       ; If requested block has been successfully allocated (by rounding up to
2759                            <1>       ; the last page border), it must be deallocated later by using
2760                            <1>       ; 'deallocate_memory_block' procedure.
2761                            <1>
2762 0000647D 52                <1>       push   edx ; *
2763 0000647E BAFF0F0000        <1>       mov    edx, PAGE_SIZE - 1   ; 4095
2764 00006483 01D0              <1>       add    eax, edx
2765 00006485 01D1              <1>       add    ecx, edx
2766 00006487 C1E90C            <1>       shr    ecx, PAGE_SHIFT         ; 12
2767                            <1>
```

```
2768                                    <1>       ; ECX = number of contiguous pages to be allocated
2769 0000648A 8B15[C8890100]           <1>       mov    edx, [free_pages]
2770                                    <1>       ; 01/05/2017
2771                                    <1>       ;or    ecx, ecx
2772                                    <1>       ;jz    short amb3
2773                                    <1>       ; If ECX=0, set cf to 1 and return with max. available mem block size
2774                                    <1>
2775 00006490 39D1                     <1>       cmp    ecx, edx
2776 00006492 7760                     <1>       ja     short amb_3
2777                                    <1>
2778 00006494 C1E80C                   <1>       shr    eax, PAGE_SHIFT      ; 12
2779                                    <1>
2780 00006497 89C2                     <1>       mov    edx, eax          ; page number
2781 00006499 C1EA03                   <1>       shr    edx, 3            ; to get offset to M.A.T.
2782                                    <1>                                ; (1 allocation bit = 1 page)
2783                                    <1>                                ; (1 allocation bytes = 8 pages)
2784 0000649C 80E2FC                   <1>       and    dl, 0FCh          ; clear lower 2 bits
2785                                    <1>                                ; (to get 32 bit position)
2786 0000649F 53                       <1>       push   ebx ; **
2787                                    <1> amb_0:
2788 000064A0 890D[7C950100]           <1>       mov    [mem_ipg_count], ecx ; initial (reset) value of page count
2789 000064A6 890D[80950100]           <1>       mov    [mem_pg_count], ecx
2790 000064AC 31C9                     <1>       xor    ecx, ecx ; 0
2791 000064AE 890D[84950100]           <1>       mov    [mem_aperture], ecx ; 0
2792 000064B4 890D[88950100]           <1>       mov    [mem_max_aperture], ecx ; 0
2793                                    <1>
2794 000064BA BB00001000               <1>       mov    ebx, MEM_ALLOC_TBL   ; Memory Allocation Table address.
2795 000064BF 3B15[CC890100]           <1>       cmp    edx, [next_page]     ; Is the beginning page address lower
2796                                    <1>                                ; than the address in 'next_page' ?
2797                                    <1>                                ; (the first/next free page of user space)
2798 000064C5 7208                     <1>       jb     short amb_1
2799 000064C7 3B15[D0890100]           <1>       cmp    edx, [last_page]     ; is the beginning page address higher
2800                                    <1>                                ; than the address in 'last_page' ?
2801                                    <1>                                ; (end of the memory)
2802 000064CD 7606                     <1>       jna    short amb_2       ; no
2803                                    <1> amb_1:
2804 000064CF 8B15[CC890100]           <1>       mov    edx, [next_page]     ; M.A.T. offset (1 M.A.T. byte = 8 pages)
2805                                    <1> amb_2:
2806 000064D5 01D3                     <1>       add    ebx, edx
2807                                    <1>
2808                                    <1>       ; 28/04/2017
2809                                    <1>       ;xor   ecx, ecx
2810 000064D7 0FBC0B                   <1>       bsf    ecx, [ebx]        ; 0 to 31
2811 000064DA 89D0                     <1>       mov    eax, edx
2812 000064DC C1E003                   <1>       shl    eax, 3            ; *8
2813 000064DF 01C8                     <1>       add    eax, ecx          ; beginning page number
2814                                    <1>
2815 000064E1 A3[8C950100]             <1>       mov    [mem_pg_pos], eax    ; beginning page no (for curr. mem. aperture)
2816 000064E6 A3[90950100]             <1>       mov    [mem_max_pg_pos], eax ; beginning page no for max. mem. aperture
2817                                    <1>
2818 000064EB 83E01F                   <1>       and    eax, 1Fh          ; lower 5 bits only (0 to 31)
2819                                    <1>                                ; (allocation bit position)
2820 000064EE 750E                     <1>       jnz    short amb_4       ; 0
2821 000064F0 B120                     <1>       mov    cl, 32
2822 000064F2 EB4B                     <1>       jmp    short amb_10
2823                                    <1>
2824                                    <1> amb_3:       ; out_of_memory
2825 000064F4 31C0                     <1>       xor    eax, eax ; 0
2826 000064F6 89D1                     <1>       mov    ecx, edx ; free pages
2827 000064F8 C1E10C                   <1>       shl    ecx, PAGE_SHIFT
2828 000064FB 5A                       <1>       pop    edx ; *
2829 000064FC F9                       <1>       stc
2830 000064FD C3                       <1>       retn
2831                                    <1> amb_4:
2832 000064FE 8B13                     <1>       mov    edx, [ebx]
2833 00006500 88C1                     <1>       mov    cl, al ; 1 to 31
2834 00006502 D3EA                     <1>       shr    edx, cl
2835 00006504 89D0                     <1>       mov    eax, edx
2836                                    <1> amb_5:
2837 00006506 D1E8                     <1>       shr    eax, 1 ; (***)
2838 00006508 7317                     <1>       jnc    short amb_7
2839 0000650A FF05[84950100]           <1>       inc    dword [mem_aperture]
2840 00006510 FF0D[80950100]           <1>       dec    dword [mem_pg_count]
2841 00006516 7470                     <1>       jz     short amb_15
2842                                    <1> amb_6:
2843                                    <1>       ; 28/04/2017
2844 00006518 FEC1                     <1>       inc    cl
2845 0000651A 80F920                   <1>       cmp    cl, 32
2846 0000651D 730D                     <1>       jnb    short amb_9
2847 0000651F EBE5                     <1>       jmp    short amb_5
2848                                    <1> amb_7:
2849 00006521 50                       <1>       push   eax ; (***) allocation bits (in shifted status)
2850 00006522 E81B010000               <1>       call   amb_26 ; set maximum memory aperture (free memory block size)
2851 00006527 58                       <1>       pop    eax ; (***)
2852 00006528 EBEE                     <1>       jmp    short amb_6
2853                                    <1> amb_8:
2854                                    <1>       ; 28/04/2017
2855 0000652A B120                     <1>       mov    cl, 32
2856                                    <1> amb_9:
2857 0000652C 89DA                     <1>       mov    edx, ebx
2858 0000652E 81EA00001000             <1>       sub    edx, MEM_ALLOC_TBL
2859 00006534 3B15[D0890100]           <1>       cmp    edx, [last_page]
2860 0000653A 7336                     <1>       jnb    short amb_14 ; contiguous pages not enough
2861 0000653C 83C304                   <1>       add    ebx, 4
2862                                    <1> amb_10:
2863 0000653F 8B03                     <1>       mov    eax, [ebx]
2864 00006541 21C0                     <1>       and    eax, eax
2865 00006543 7408                     <1>         jz     short amb_11 ; there is not a free page bit in this alloc dword
2866 00006545 40                       <1>       inc    eax ; 0FFFFFFFFh -> 0
2867 00006546 740C                     <1>       jz     short amb_12 ; all of bits are set (32 free pages)
2868 00006548 48                       <1>       dec    eax
2869 00006549 28C9                     <1>       sub    cl, cl ; 0
2870 0000654B EBB9                     <1>       jmp    short amb_5
2871                                    <1> amb_11:
2872 0000654D E8F0000000               <1>       call   amb_26 ; set maximum memory aperture (free memory block size)
```

```
2873 00006552 EBD8              <1>       jmp    short amb_9
2874                            <1> amb_12:
2875 00006554 390D[80950100]    <1>       cmp    [mem_pg_count], ecx ; 32
2876 0000655A 7306              <1>       jnb    short amb_13
2877 0000655C 8B0D[80950100]    <1>       mov    ecx, [mem_pg_count]
2878                            <1> amb_13:
2879 00006562 010D[84950100]    <1>       add    [mem_aperture], ecx
2880 00006568 290D[80950100]    <1>       sub    [mem_pg_count], ecx
2881 0000656E 7618              <1>       jna    short amb_15
2882 00006570 EBBA              <1>       jmp    short amb_9 ; 01/05/2017
2883                            <1> amb_14:
2884 00006572 E8CB000000        <1>       call   amb_26 ; 28/04/2017
2885 00006577 A1[90950100]      <1>       mov    eax, [mem_max_pg_pos] ; begin address of max. mem aperture
2886 0000657C 8B0D[88950100]    <1>       mov    ecx, [mem_max_aperture] ; max. (largest) memory aperture
2887 00006582 F9                <1>       stc
2888 00006583 E9AF000000        <1>         jmp     amb_25
2889                            <1>
2890                            <1> amb_15: ; OK !
2891 00006588 A1[8C950100]      <1>       mov    eax, [mem_pg_pos]    ; Beginning address as page number
2892 0000658D 8B0D[84950100]    <1>       mov    ecx, [mem_aperture]  ; Free contiguous page count (>=1)
2893                            <1> amb_16:
2894                            <1>       ; allocate contiguous memory pages (via memory allocation table bits)
2895 00006593 89C2              <1>       mov    edx, eax
2896                            <1>       ; 25/04/2017
2897 00006595 C1EA03            <1>       shr    edx, 3        ; 8 pages in one allocation byte
2898 00006598 80E2FC            <1>       and    dl, 0FCh      ; clear lower 2 bits
2899                            <1>                            ; (for dword/32bit positioning)
2900                            <1>
2901 0000659B BB00001000        <1>       mov    ebx, MEM_ALLOC_TBL
2902 000065A0 01D3              <1>       add    ebx, edx
2903 000065A2 83E01F            <1>       and    eax, 1Fh ; 31
2904                            <1>       ; 03/04/2016
2905 000065A5 BA20000000        <1>       mov    edx, 32
2906 000065AA 28C2              <1>       sub    dl, al
2907 000065AC 39CA              <1>       cmp    edx, ecx      ; ecx >= 1
2908 000065AE 7602              <1>       jna    short amb_17
2909 000065B0 89CA              <1>       mov    edx, ecx
2910                            <1> amb_17:
2911 000065B2 29D1              <1>       sub    ecx, edx
2912 000065B4 51                <1>       push   ecx ; ***
2913 000065B5 89D1              <1>       mov    ecx, edx
2914                            <1> amb_18:
2915 000065B7 0FB303            <1>       btr    [ebx], eax    ; The destination bit indexed by the source value
2916                            <1>                            ; is copied into the Carry Flag and then cleared
2917                            <1>                            ; in the destination.
2918 000065BA FF0D[C8890100]    <1>       dec     dword [free_pages] ; 1 page has been allocated (X = X-1)
2919 000065C0 49                <1>       dec    ecx
2920 000065C1 7404              <1>       jz     short amb_19
2921 000065C3 FEC0              <1>       inc    al
2922 000065C5 EBF0              <1>       jmp    short amb_18
2923                            <1> amb_19:
2924 000065C7 59                <1>       pop    ecx ; ***
2925 000065C8 21C9              <1>       and    ecx, ecx ; 0 ?
2926 000065CA 741E              <1>       jz     short amb_22
2927                            <1>       ; 01/04/2016
2928 000065CC B020              <1>       mov    al, 32
2929                            <1> amb_20:
2930 000065CE 83C304            <1>       add    ebx, 4
2931 000065D1 39C1              <1>       cmp    ecx, eax ; 32
2932 000065D3 7305              <1>       jnb    short amb_21
2933                            <1>       ; ECX < 32
2934 000065D5 28C0              <1>       sub    al, al ; 0
2935 000065D7 50                <1>       push   eax ; 0 ***
2936 000065D8 EBDD              <1>       jmp    short amb_18
2937                            <1> amb_21:
2938 000065DA 2905[C8890100]    <1>       sub    [free_pages], eax   ; [free_pages] = [free_pages] - 32
2939 000065E0 C70300000000      <1>       mov    dword [ebx], 0          ; reset 32 bits
2940 000065E6 29C1              <1>       sub    ecx, eax ; 32
2941 000065E8 75E4              <1>       jnz    short amb_20
2942                            <1> amb_22:
2943 000065EA A1[8C950100]      <1>       mov    eax, [mem_pg_pos]   ; Beginning address as page number
2944 000065EF 8B0D[84950100]    <1>       mov    ecx, [mem_aperture] ; Free contiguous page count
2945                            <1>       ; [next_page] update
2946 000065F5 89C2              <1>       mov    edx, eax
2947                            <1>       ; 03/04/2016
2948 000065F7 C1EA03            <1>       shr    edx, 3            ; to get offset to M.A.T.
2949                            <1>                                ; (1 allocation bit = 1 page)
2950                            <1>                                ; (1 allocation bytes = 8 pages)
2951 000065FA 80E2FC            <1>       and    dl, 0FCh          ; clear lower 2 bits
2952                            <1>                                ; (to get 32 bit position)
2953 000065FD 3B15[CC890100]    <1>       cmp    edx, [next_page] ; first free page pointer offset
2954 00006603 7732              <1>       ja     short amb_25
2955 00006605 BB00001000        <1>       mov    ebx, MEM_ALLOC_TBL
2956 0000660A 833C1300          <1>       cmp    dword [ebx+edx], 0
2957 0000660E 7721              <1>       ja     short amb_24
2958 00006610 89C2              <1>       mov    edx, eax
2959 00006612 01CA              <1>       add    edx, ecx
2960 00006614 C1EA03            <1>       shr    edx, 3
2961 00006617 80E2FC            <1>       and    dl, 0FCh
2962                            <1> amb_23:
2963 0000661A 833C1300          <1>       cmp    dword [ebx+edx], 0
2964 0000661E 7711              <1>       ja     short amb_24
2965 00006620 83C204            <1>       add    edx, 4
2966 00006623 3B15[D0890100]    <1>       cmp    edx, [last_page]    ; last page pointer offset
2967 00006629 76EF              <1>       jna    short amb_23
2968 0000662B 8B15[D4890100]    <1>       mov    edx, [first_page]   ; (for) beginning of user's space
2969                            <1> amb_24:
2970 00006631 8915[CC890100]    <1>       mov    [next_page], edx
2971                            <1> amb_25:
2972 00006637 9C                <1>       pushf
2973 00006638 C1E00C            <1>       shl    eax, PAGE_SHIFT         ; convert to phy. address in bytes
2974 0000663B C1E10C            <1>       shl    ecx, PAGE_SHIFT         ; convert to byte counts
2975 0000663E 9D                <1>       popf
2976 0000663F 5B                <1>       pop    ebx ; **
2977 00006640 5A                <1>       pop    edx ; *
```

```
2978 00006641 C3                  <1>       retn
2979                              <1>
2980                              <1> amb_26:     ; set maximum free memory aperture (free memory block size)
2981 00006642 89DA                <1>       mov    edx, ebx ; current address
2982 00006644 81EA00001000        <1>       sub    edx, MEM_ALLOC_TBL ; MAT beginning address
2983                              <1>       ; 02/04/2016
2984 0000664A C1E203              <1>       shl    edx, 3 ; MAT byte offset * 8 = page number base
2985 0000664D 01CA                <1>       add    edx, ecx ; current page number (ecx = 0 to 32)
2986                              <1>       ;
2987 0000664F A1[84950100]        <1>       mov    eax, [mem_aperture]
2988 00006654 21C0                <1>       and    eax, eax
2989 00006656 7421                <1>         jz      short amb_27
2990 00006658 C705[84950100]0000- <1>         mov     dword [mem_aperture], 0
2990 00006660 0000                <1>
2991 00006662 3B05[88950100]      <1>       cmp    eax, [mem_max_aperture]
2992 00006668 760F                <1>       jna    short amb_27
2993 0000666A A3[88950100]        <1>       mov    [mem_max_aperture], eax
2994                              <1>       ; 25/04/2017
2995 0000666F A1[8C950100]        <1>       mov    eax, [mem_pg_pos]
2996                              <1>       ; EAX = Beginning page number of the max. aperture
2997 00006674 A3[90950100]        <1>       mov    [mem_max_pg_pos], eax
2998                              <1> amb_27:
2999 00006679 8915[8C950100]      <1>       mov    [mem_pg_pos], edx ; current page
3000                              <1>
3001 0000667F A1[7C950100]        <1>       mov    eax, [mem_ipg_count] ; initial (reset) value of page count
3002 00006684 A3[80950100]        <1>       mov    [mem_pg_count], eax
3003                              <1>
3004 00006689 C3                  <1>       retn
3005                              <1>
3006                              <1> deallocate_memory_block:
3007                              <1>       ; 03/04/2016
3008                              <1>       ; 14/03/2016 (TRDOS 386 = TRDOS v2.0)
3009                              <1>       ; Deallocating contiguous memory pages (in the kernel's memory space)
3010                              <1>       ;
3011                              <1>       ; INPUT ->
3012                              <1>       ;     EAX = Beginning address (physical)
3013                              <1>       ;     ECX = Number of bytes to be deallocated
3014                              <1>       ;
3015                              <1>       ; OUTPUT ->
3016                              <1>       ;     Memory Allocation Table bits will be updated
3017                              <1>       ;     [free_pages] will be changed (increased)
3018                              <1>       ;
3019                              <1>       ; (Modified Registers -> EAX, ECX)
3020                              <1>       ;
3021                              <1>       ; PURPOSE: Unloading/Freeing a file -or an allocated memory block-
3022                              <1>       ; at memory after copying, running, saving, reading, writing etc.
3023                              <1>       ;
3024                              <1>
3025 0000668A 52                  <1>       push   edx ; *
3026 0000668B 53                  <1>       push   ebx ; **
3027                              <1>
3028 0000668C C1E80C              <1>       shr    eax, PAGE_SHIFT          ; 12
3029 0000668F C1E90C              <1>       shr    ecx, PAGE_SHIFT          ; 12
3030                              <1>
3031                              <1>       ; EAX = Beginning page number
3032                              <1>       ; ECX = Number of contiguous pages to be deallocated
3033                              <1> damb_0:
3034                              <1>       ; deallocate contiguous memory pages (via memory allocation table bits)
3035 00006692 89C2                <1>       mov    edx, eax
3036 00006694 C1EA03              <1>       shr    edx, 3            ; to get offset to M.A.T.
3037                              <1>                                ; (1 allocation bit = 1 page)
3038                              <1>                                ; (1 allocation bytes = 8 pages)
3039 00006697 80E2FC              <1>       and    dl, 0FCh          ; clear lower 2 bits
3040                              <1>                                ; (to get 32 bit position)
3041 0000669A 3B15[CC890100]      <1>       cmp    edx, [next_page] ; next free page
3042 000066A0 7306                <1>       jnb    short damb_1
3043 000066A2 8915[CC890100]      <1>       mov    [next_page], edx
3044                              <1> damb_1:
3045 000066A8 BB00001000          <1>       mov    ebx, MEM_ALLOC_TBL
3046 000066AD 01D3                <1>       add    ebx, edx
3047 000066AF 83E01F              <1>       and    eax, 1Fh ; 31
3048                              <1>
3049                              <1>       ; 03/04/2016
3050 000066B2 BA20000000          <1>       mov    edx, 32
3051 000066B7 28C2                <1>       sub    dl, al
3052 000066B9 39CA                <1>       cmp    edx, ecx
3053 000066BB 7602                <1>       jna    short damb_2
3054 000066BD 89CA                <1>       mov    edx, ecx
3055                              <1> damb_2:
3056 000066BF 29D1                <1>       sub    ecx, edx
3057 000066C1 51                  <1>       push   ecx ; ***
3058 000066C2 89D1                <1>       mov    ecx, edx
3059                              <1> damb_3:
3060 000066C4 0FAB03              <1>       bts    [ebx], eax        ; unlink/release/deallocate page
3061                              <1>                                ; set relevant bit to 1.
3062                              <1>                                ; set CF to the previous bit value
3063 000066C7 FF05[C8890100]      <1>       inc     dword [free_pages]   ; 1 page has been deallocated (X = X+1)
3064 000066CD 49                  <1>       dec    ecx
3065 000066CE 7404                <1>       jz     short damb_4
3066 000066D0 FEC0                <1>       inc    al
3067 000066D2 EBF0                <1>       jmp    short damb_3
3068                              <1> damb_4:
3069 000066D4 59                  <1>       pop    ecx ; ***
3070 000066D5 21C9                <1>       and    ecx, ecx ; 0 ?
3071 000066D7 741E                <1>       jz     short damb_7
3072                              <1>       ; 03/04/2016
3073 000066D9 B020                <1>       mov    al, 32
3074                              <1> damb_5:
3075 000066DB 83C304              <1>       add    ebx, 4
3076 000066DE 39C1                <1>       cmp    ecx, eax ; 32
3077 000066E0 7305                <1>       jnb    short damb_6
3078                              <1>       ; ECX < 32
3079 000066E2 28C0                <1>       sub    al, al ; 0
3080 000066E4 50                  <1>       push   eax ; 0 ***
3081 000066E5 EBDD                <1>       jmp    short damb_3
```

```
                                <1> damb_6:
000066E7 0105[C8890100]         <1>       add   [free_pages], eax ; [free_pages] = [free_pages] + 32
000066ED C703FFFFFFFF           <1>       mov   dword [ebx], 0FFFFFFFFh ; set 32 bits
000066F3 29C1                   <1>       sub   ecx, eax ; 32
000066F5 75E4                   <1>       jnz   short damb_5
                                <1> damb_7:
000066F7 5B                     <1>       pop   ebx ; **
000066F8 5A                     <1>       pop   edx ; *
000066F9 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
000066FA 662500F0               <1>       and   ax, PTE_A_CLEAR ; clear page offset
000066FE 50                     <1>       push  eax
                                <1>       ;and  ecx, ecx ; page count
                                <1>       ;jz   dmem_acc_7  ; 'insufficient memory' error
000066FF 89C5                   <1>       mov   ebp, eax
00006701 81C300004000           <1>       add   ebx, CORE ; 12/05/2017
                                <1> dmem_acc_0:
00006707 891D[7CA00100]         <1>       mov   [base_addr], ebx ; 12/05/2017
0000670D A1[B8030300]           <1>       mov   eax, [u.pgdir] ; page dir address (physical)
00006712 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 ?
00006717 7324                   <1>       jnc   short dmem_acc_1
                                <1>       ;
00006719 E8B5F4FFFF             <1>       call  allocate_page
0000671E 0F82AB000000           <1>         jc      dmem_acc_7  ; 'insufficient memory' error
                                <1>       ;
00006724 E824F5FFFF             <1>       call  clear_page
                                <1>       ; EAX = Physical (base) address of the allocated (new) page
00006729 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)
0000672B 8902                   <1>       mov   [edx], eax ; Let's put the new page directory entry here !
0000672D A1[B8030300]           <1>       mov   eax, [u.pgdir]
00006732 E8B7F5FFFF             <1>       call  get_pte
00006737 0F8292000000           <1>         jc      dmem_acc_7 ; 'insufficient memory' error
                                <1> dmem_acc_1:
                                <1>       ; EAX = PTE value, EDX = PTE address
0000673D A801                   <1>       test  al, PTE_A_PRESENT
0000673F 750D                   <1>       jnz   short dmem_acc_2
00006741 09C0                   <1>       or    eax, eax
00006743 7468                   <1>       jz    short dmem_acc_6   ; Change PTE
00006745 D1E8                   <1>       shr   eax, 1       ; swap disk block (8 sectors) address
                                <1>       ; unlink swap disk block
00006747 E80CFBFFFF             <1>       call  unlink_swap_block
0000674C EB5F                   <1>       jmp   short dmem_acc_6
                                <1>
                                <1> dmem_acc_2:
0000674E A802                   <1>       test  al, PTE_A_WRITE   ; bit 1, writable (r/w) flag
                                <1>                               ; (must be 1)
00006750 7550                   <1>       jnz   short dmem_acc_4
                                <1>       ; Read only -duplicated- page (belongs to a parent or a child)
00006752 66A90002               <1>         test    ax, PTE_DUPLICATED ; Was this page duplicated
                                <1>                                 ; as child's page ?
00006756 7455                   <1>       jz    short dmem_acc_5 ; Change PTE but don't deallocate the page!
```

```
3187                                    <1>
3188                                    <1>       ;push  edi
3189                                    <1>       ;push  esi
3190                                    <1>
3191 00006758 51                       <1>       push   ecx
3192                                    <1>       ;push  ebx
3193 00006759 8B1D[BC030300]            <1>       mov    ebx, [u.ppgdir] ; parent's page dir address (physical)
3194                                    <1>
3195                                    <1>       ; check the parent's PTE value is read only & same page or not..
3196 0000675F 89EF                     <1>       mov    edi, ebp
3197 00006761 C1EF16                   <1>       shr    edi, PAGE_D_SHIFT ; 22
3198                                    <1>       ; EDI = page directory entry index (0-1023)
3199 00006764 89EE                     <1>       mov    esi, ebp
3200 00006766 C1EE0C                   <1>       shr    esi, PAGE_SHIFT ; 12
3201 00006769 81E6FF030000             <1>       and    esi, PTE_MASK
3202                                    <1>       ; ESI = page table entry index (0-1023)
3203                                    <1>
3204 0000676F 66C1E702                 <1>       shl    di, 2 ; * 4
3205 00006773 01FB                     <1>       add    ebx, edi ; PDE offset (for the parent)
3206 00006775 8B0F                     <1>       mov    ecx, [edi]
3207 00006777 F6C101                   <1>       test   cl, PDE_A_PRESENT ; present (valid) or not ?
3208 0000677A 7425                     <1>       jz     short dmem_acc_3    ; parent process does not use this page
3209 0000677C 6681E100F0               <1>       and    cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
3210 00006781 66C1E602                 <1>       shl    si, 2 ; *4
3211 00006785 01CE                     <1>       add    esi, ecx ; PTE offset (for the parent)
3212 00006787 8B1E                     <1>       mov    ebx, [esi]
3213 00006789 F6C301                   <1>       test   bl, PTE_A_PRESENT ; present or not ?
3214 0000678C 7413                     <1>       jz     short dmem_acc_3    ; parent process does not use this page
3215 0000678E 662500F0                 <1>       and    ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3216 00006792 6681E300F0               <1>       and    bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3217 00006797 39D8                     <1>       cmp    eax, ebx     ; parent's and child's pages are same ?
3218 00006799 7506                     <1>       jne    short dmem_acc_3    ; not same page
3219                                    <1>                           ; deallocate the child's page
3220 0000679B 800E02                   <1>         or     byte [esi], PTE_A_WRITE ; convert to writable page (parent)
3221                                    <1>       ;pop   ebx
3222 0000679E 59                       <1>       pop    ecx
3223 0000679F EB0C                     <1>       jmp    short dmem_acc_5
3224                                    <1> dmem_acc_3:
3225                                    <1>       ;pop   ebx
3226 000067A1 59                       <1>       pop    ecx
3227                                    <1> dmem_acc_4:
3228 000067A2 66A90004                 <1>       test   ax, PTE_SHARED ; shared or direct memory access indicator
3229 000067A6 7505                     <1>       jnz    short dmem_acc_5   ; AVL bit 1 = 1, do not deallocate this page!
3230                                    <1>       ;
3231                                    <1>       ;and   ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3232 000067A8 E804F6FFFF               <1>       call   deallocate_page
3233                                    <1> dmem_acc_5:
3234                                    <1>       ;pop   esi
3235                                    <1>       ;pop   edi
3236                                    <1> dmem_acc_6:
3237 000067AD 89E8                     <1>       mov    eax, ebp ; physical page (offset=0) address
3238                                    <1>       ; EAX = memory page address
3239                                    <1>       ; EDX = PTE entry address (physical)
3240 000067AF 660D0704                 <1>       or     ax, PTE_A_PRESENT+PTE_A_USER+PTE_A_WRITE+PTE_SHARED
3241                                    <1>                   ; present flag, bit 0 = 1
3242                                    <1>                   ; user flag, bit 2 = 1
3243                                    <1>                   ; writable flag, bit 1 = 1
3244                                    <1>                   ; direct memory access flag, bit 10 = 1
3245                                    <1>                   ; (This page must not be deallocated!)
3246 000067B3 8902                     <1>       mov    [edx], eax  ; Update PTE value
3247 000067B5 49                       <1>       dec    ecx ; remain count of contiguous pages
3248 000067B6 741E                     <1>       jz     short dmem_acc_8
3249 000067B8 81C500100000             <1>       add    ebp, PAGE_SIZE ; next physical page address
3250                                    <1>       ; 22/07/2017
3251                                    <1>       ;mov   eax, ebp
3252                                    <1>       ; 12/05/2017
3253 000067BE 8B1D[7CA00100]           <1>       mov    ebx, [base_addr] ; linear address (virtual+CORE)
3254 000067C4 81C300100000             <1>       add    ebx, PAGE_SIZE      ; next linear address
3255 000067CA E938FFFFFF               <1>         jmp     dmem_acc_0
3256                                    <1> dmem_acc_7:  ; ERROR !
3257 000067CF C7042404000000           <1>       mov    dword [esp], ERR_MINOR_IM
3258                                    <1>              ; Insufficient memory (minor) error!
3259                                    <1>              ; Major error = 0 (No protection fault)
3260                                    <1>       ; cf = 1
3261                                    <1> dmem_acc_8:
3262 000067D6 58                       <1>       pop    eax
3263                                    <1>       ;pop   edx
3264                                    <1>       ;pop   ecx
3265                                    <1>       ;pop   ebx
3266                                    <1>       ;pop   ebp
3267 000067D7 C3                       <1>       retn
3268                                    <1>
3269                                    <1> deallocate_user_pages:
3270                                    <1>       ; 20/05/2017
3271                                    <1>       ; 15/05/2017
3272                                    <1>       ; 20/02/2017
3273                                    <1>       ; 19/02/2017 (TRDOS 386 = TRDOS v2.0)
3274                                    <1>       ;
3275                                    <1>       ; Deallocate virtually contiguous user pages (memory block)
3276                                    <1>       ; (caller: 'sysdalloc' system call)
3277                                    <1>       ;
3278                                    <1>       ; INPUT ->
3279                                    <1>       ;     EBX = VIRTUAL ADDRESS (beginning address)
3280                                    <1>       ;     ECX = byte count
3281                                    <1>       ;     [u.pgdir] = user's page directory
3282                                    <1>       ;     [u.ppdir] = parent's page directory
3283                                    <1>       ;
3284                                    <1>       ; OUTPUT ->
3285                                    <1>       ;    If CF = 0
3286                                    <1>       ;     EAX = Deallocated memory bytes
3287                                    <1>       ;       (Even if shared or read only pages will not be
3288                                    <1>       ;        deallocated on M.A.T., this byte count will be
3289                                    <1>       ;        returned as virtually deallocated bytes; in fact
3290                                    <1>       ;        virtually deallocated user pages * 4096.)
3291                                    <1>       ;     EBX = Virtual address (as rounded up)
```

```
3292                                    <1>     ;    If CF = 1
3293                                    <1>     ;      EAX = 0 (there is not any deallocated pages)
3294                                    <1>     ;
3295                                    <1>     ; Note: Empty page tables will not be deallocated!!!
3296                                    <1>     ;     (they will be deallocated at process termination stage)
3297                                    <1>     ;
3298                                    <1>     ; Modified Registers -> EAX, EDX, ESI, EDI, EBX, ECX, EBP
3299                                    <1>     ;
3300 000067D8 89DE                     <1>     mov    esi, ebx
3301 000067DA 89F7                     <1>     mov    edi, esi
3302 000067DC 01CF                     <1>     add    edi, ecx
3303 000067DE 81C6FF0F0000             <1>     add    esi, PAGE_SIZE - 1  ; 4095 (round up)
3304 000067E4 C1EE0C                   <1>     shr    esi, PAGE_SHIFT
3305 000067E7 C1EF0C                   <1>     shr    edi, PAGE_SHIFT
3306 000067EA 89F8                     <1>     mov    eax, edi ; end page
3307 000067EC 29F0                     <1>     sub    eax, esi ; end page - start page
3308 000067EE 0F86D5000000             <1>     jna    da_u_pd_err  ; < 1
3309 000067F4 89F3                     <1>     mov    ebx, esi
3310 000067F6 C1E30C                   <1>     shl    ebx, PAGE_SHIFT ; virtual address (as rounded up)
3311 000067F9 53                       <1>     push   ebx ; *
3312 000067FA 89C1                     <1>     mov    ecx, eax ; page count
3313 000067FC C1E00C                   <1>     shl    eax, PAGE_SHIFT ; byte count as adjusted
3314 000067FF 50                       <1>     push   eax ; **
3315 00006800 8B1D[B8030300]           <1>     mov    ebx, [u.pgdir] ; physical addr of user's page dir
3316 00006806 81C600040000             <1>     add    esi, CORE/PAGE_SIZE
3317 0000680C 89F7                     <1>     mov    edi, esi
3318 0000680E 81E7FF030000             <1>     and    edi, PTE_MASK ; PTE entry in the page table
3319 00006814 57                       <1>     push   edi ; *** ; PTE index (of page directory)
3320 00006815 C1EE0A                   <1>     shr    esi, PAGE_D_SHIFT - PAGE_SHIFT ; 22-12=10
3321 00006818 89F2                     <1>     mov    edx, esi
3322                                    <1>     ; EDX = PDE index
3323 0000681A C1E602                   <1>     shl    esi, 2 ; convert PDE index to dword offset
3324 0000681D 01DE                     <1>     add    esi, ebx ; add page directory address
3325                                    <1> da_u_pd_1:
3326 0000681F AD                       <1>     lodsd
3327                                    <1>     ;
3328 00006820 89F5                     <1>     mov    ebp, esi ; 20/02/2017
3329                                    <1>     ; EBP = next PDE address
3330                                    <1>     ;
3331 00006822 A801                     <1>     test   al, PDE_A_PRESENT ; bit 0, present flag (must be 1)
3332 00006824 0F8494000000             <1>     jz     da_u_pd_3 ; 20/05/2017
3333 0000682A 662500F0                 <1>     and    ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3334                                    <1>     ; EAX = PHYSICAL (flat) ADDRESS OF THE PAGE TABLE
3335 0000682E 8B3C24                   <1>     mov    edi, [esp] ; ***
3336                                    <1>     ; EDI = PTE index (of complete page directory)
3337                                    <1>     ;and   edi, PTE_MASK ; PTE entry in the page table
3338 00006831 C1E702                   <1>     shl    edi, 2 ; convert PTE index to dword offset
3339 00006834 89FE                     <1>     mov    esi, edi ; PTE offset in page table (0-4092)
3340 00006836 01C6                     <1>     add    esi, eax ; now, esi points to requested PTE
3341                                    <1> da_u_pt_0:
3342 00006838 AD                       <1>     lodsd
3343 00006839 A801                     <1>     test   al, PTE_A_PRESENT ; bit 0, present flag (must be 1)
3344 0000683B 743F                     <1>     jz     short da_u_pt_1
3345                                    <1>     ;
3346 0000683D A802                     <1>     test   al, PTE_A_WRITE   ; bit 1, writable (r/w) flag
3347                                    <1>                              ; (must be 1)
3348 0000683F 7549                     <1>     jnz    short da_u_pt_3
3349                                    <1>     ; Read only -duplicated- page (belongs to a parent or a child)
3350 00006841 66A90002                 <1>       test    ax, PTE_DUPLICATED ; Was this page duplicated
3351                                    <1>                                ; as child's page ?
3352 00006845 744E                     <1>     jz     short da_u_pt_4 ; Clear PTE but don't deallocate the page!
3353                                    <1>     ;
3354                                    <1>     ; check the parent's PTE value is read only & same page or not..
3355                                    <1>     ; EDX = page directory entry index (0-1023)
3356 00006847 52                       <1>     push   edx ; ****
3357                                    <1>     ; EDI = page table entry offset (0-4092)
3358 00006848 8B1D[BC030300]           <1>     mov    ebx, [u.ppgdir] ; page directory of the parent process
3359 0000684E 66C1E202                 <1>     shl    dx, 2 ; *4
3360 00006852 01D3                     <1>     add    ebx, edx ; PDE address (for the parent)
3361 00006854 8B13                     <1>     mov    edx, [ebx] ; page table address
3362 00006856 F6C201                   <1>     test   dl, PDE_A_PRESENT ; present (valid) or not ?
3363 00006859 742E                     <1>     jz     short da_u_pt_2     ; parent process does not use this page
3364 0000685B 6681E200F0               <1>     and    dx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
3365                                    <1>     ; EDI = page table entry offset (0-4092)
3366 00006860 01D7                     <1>     add    edi, edx     ; PTE address (for the parent)
3367 00006862 8B1F                     <1>     mov    ebx, [edi]
3368 00006864 F6C301                   <1>     test   bl, PTE_A_PRESENT ; present or not ?
3369 00006867 7420                     <1>     jz     short da_u_pt_2     ; parent process does not use this page
3370 00006869 662500F0                 <1>     and    ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3371 0000686D 6681E300F0               <1>     and    bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3372 00006872 39D8                     <1>     cmp    eax, ebx     ; parent's and child's pages are same ?
3373 00006874 7513                     <1>     jne    short da_u_pt_2     ; not same page
3374                                    <1>                                ; deallocate the child's page
3375 00006876 800F02                   <1>       or      byte [edi], PTE_A_WRITE ; convert to writable page (parent)
3376 00006879 5A                       <1>     pop    edx ; ****
3377 0000687A EB19                     <1>     jmp    short da_u_pt_4
3378                                    <1> da_u_pt_1:
3379 0000687C 09C0                     <1>     or     eax, eax     ; swapped page ?
3380 0000687E 741C                     <1>     jz     short da_u_pt_5     ; no
3381                                    <1>                                ; yes
3382 00006880 D1E8                     <1>     shr    eax, 1
3383 00006882 E8D1F9FFFF               <1>     call   unlink_swap_block ; Deallocate swapped page block
3384                                    <1>                              ; on the swap disk (or in file)
3385 00006887 EB13                     <1>     jmp    short da_u_pt_5
3386                                    <1> da_u_pt_2:
3387 00006889 5A                       <1>     pop    edx ; ****
3388                                    <1> da_u_pt_3:
3389 0000688A 66A90004                 <1>     test   ax, PTE_SHARED      ; shared or direct memory access indicator
3390 0000688E 7505                     <1>     jnz    short da_u_pt_4     ; AVL bit 1 = 1, do not deallocate this page!
3391                                    <1>     ;
3392                                    <1>     ;and   ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3393 00006890 E81CF5FFFF               <1>     call   deallocate_page ; set the mem allocation bit of this page
3394                                    <1> da_u_pt_4:
3395 00006895 C746FC00000000           <1>     mov    dword [esi-4], 0 ; clear/reset PTE (child, dupl. as parent)
3396                                    <1> da_u_pt_5:
```

```
3397                                    <1>       ; 20/05/2017
3398 0000689C 58                        <1>       pop    eax ; *** PTE index (of page directory)
3399 0000689D 49                        <1>       dec    ecx ; remain page count
3400 0000689E 7426                      <1>       jz     short da_u_pd_4
3401 000068A0 40                        <1>       inc    eax ; next PTE
3402 000068A1 6625FF03                  <1>       and    ax, PTE_MASK ; PTE entry index in the page table
3403 000068A5 50                        <1>       push   eax ; *** (save again)
3404                                    <1>       ;mov   edi, eax
3405                                    <1>       ;and   di, PTE_MASK
3406                                    <1>       ;cmp   edi, PAGE_SIZE / 4 ; 1024
3407                                    <1>       ;jnb   short da_u_pd_2
3408 000068A6 89C7                      <1>       mov    edi, eax
3409 000068A8 C1E702                    <1>       shl    edi, 2 ; convert index to dword offset
3410                                    <1>       ;test  ax, PTE_MASK ; 3FFh
3411 000068AB 09C0                      <1>       or     eax, eax
3412 000068AD 7589                      <1>       jnz    short da_u_pt_0 ; 1-1023
3413                                    <1> da_u_pd_2:
3414 000068AF 42                        <1>       inc    edx
3415                                    <1>       ; 20/05/2017
3416 000068B0 6681E2FF03                <1>       and    dx, PTE_MASK  ; 3FFh
3417 000068B5 740F                      <1>       jz     short da_u_pd_4  ; 0 (1024)
3418                                    <1>       ;cmp   edx, 1024
3419                                    <1>       ;jnb   short da_u_pd_4
3420 000068B7 89EE                      <1>       mov    esi, ebp ; 20/02/2017
3421 000068B9 E961FFFFFF                <1>       jmp    da_u_pd_1
3422                                    <1> da_u_pd_3:
3423                                    <1>       ; 15/05/2017 (empty page directory entry)
3424 000068BE 81E900040000              <1>       sub    ecx, 1024
3425 000068C4 77E9                      <1>       ja     short da_u_pd_2 ; 20/05/2017
3426                                    <1> da_u_pd_4:
3427 000068C6 58                        <1>       pop    eax ; **
3428 000068C7 5B                        <1>       pop    ebx ; *
3429 000068C8 C3                        <1>       retn
3430                                    <1>
3431                                    <1> da_u_pd_err:
3432 000068C9 31C0                      <1>       xor    eax, eax
3433 000068CB F9                        <1>       stc
3434 000068CC C3                        <1>       retn
3435                                    <1>
3436                                    <1> allocate_user_pages:
3437                                    <1>       ; 20/05/2017
3438                                    <1>       ; 01/05/2017, 02/05/2017, 15/05/2017
3439                                    <1>       ; 04/03/2017
3440                                    <1>       ; 20/02/2017 (TRDOS 386 = TRDOS v2.0)
3441                                    <1>       ;
3442                                    <1>       ; Allocate physically contiguous user pages (memory block)
3443                                    <1>       ; (caller: 'sysalloc' system call)
3444                                    <1>       ;
3445                                    <1>       ; Note: This procedure does not alloc a page's itself
3446                                    <1>       ;     (page bit) on Memory Allocation Table.
3447                                    <1>       ;     (allocate_memory_block is needed before this proc)
3448                                    <1>       ;
3449                                    <1>       ; INPUT ->
3450                                    <1>       ;     EAX = PHYSICAL ADDRESS (beginning address)
3451                                    <1>       ;     EBX = VIRTUAL ADDRESS (beginning address)
3452                                    <1>       ;     ECX = byte count (>=4096)
3453                                    <1>       ;     [u.pgdir] = user's page directory
3454                                    <1>       ;
3455                                    <1>       ;     Note: All addresses are (must be) already adjusted
3456                                    <1>       ;     to page      borders, otherwise, lower 12bits of addresses
3457                                    <1>       ;     and byte count would be truncated.
3458                                    <1>       ;
3459                                    <1>       ; OUTPUT ->
3460                                    <1>       ;     none
3461                                    <1>       ;
3462                                    <1>       ;     CF = 1 -> insufficient memory error
3463                                    <1>       ;
3464                                    <1>       ; Note: All pages will be allocated in physical page order
3465                                    <1>       ;     from the beginning page address.
3466                                    <1>       ;     * A new page table will be added to the page dir
3467                                    <1>       ;       when the requested PDE is invalid.
3468                                    <1>       ;     * Those pages will not be added to swap queue
3469                                    <1>       ;       because main purpose of this allocation is to
3470                                    <1>       ;       set a direct memory access (DMA controller) buffer.
3471                                    <1>       ;      (Swapping out a page in a DMA buffer would be wrong!)
3472                                    <1>       ;     * Previous content of page tables (PTEs) would be
3473                                    <1>       ;       (should be) deallocated before entering this
3474                                    <1>       ;       procedure. So, new page table entries (PTEs)
3475                                    <1>       ;       directly will be written without checking
3476                                    <1>       ;       their previous content.
3477                                    <1>       ;     * Only solution to increase free memory by removing
3478                                    <1>       ;       that non-swappable memory block is to terminate
3479                                    <1>       ;       the process or to wait until the process will
3480                                    <1>       ;       deallocate that memory block as itself. ('sysdalloc')
3481                                    <1>       ;       (No problem, if the process does not grab all of
3482                                    <1>       ;       -very big amount of- free memory by using
3483                                    <1>       ;       'sysalloc' system call!?)
3484                                    <1>       ;       (Even if the process has grabbed all of free memory,
3485                                    <1>       ;       no problem if the process is not running in
3486                                    <1>       ;       multitasking mode. No problem in multitasking
3487                                    <1>       ;       mode if there is not another process which is running
3488                                    <1>       ;       or waiting or sleeping for an event as it's pages
3489                                    <1>       ;       are swapped-out. But a new process can not start to
3490                                    <1>       ;       run if all of free memory has beeen allocated
3491                                    <1>       ;       by running processes. Deallocation -'sysdalloc'-
3492                                    <1>       ;       or terminate a running process is needed
3493                                    <1>       ;       in order to run a new process.)
3494                                    <1>       ;
3495                                    <1>       ; Modified Registers -> EAX, EDX, ESI, EDI, EBX, ECX, EBP
3496                                    <1>       ;
3497                                    <1>
3498                                    <1>       ; 01/05/2017
3499 000068CD 662500F0                  <1>       and    ax, ~PAGE_OFF
3500 000068D1 6681E300F0                <1>       and    bx, ~PAGE_OFF
3501                                    <1>       ; 02/05/2017
```

```
3502 000068D6 BD00F0FFFF          <1>       mov    ebp, 0FFFFF000h ; 4 Giga Bytes - 4096 Bytes (for Stack)
3503 000068DB C1E90C              <1>       shr    ecx, PAGE_SHIFT ; page count
3504 000068DE 83F901              <1>       cmp    ecx, 1
3505 000068E1 7251                <1>       jb     short a_u_im_retn
3506 000068E3 89C2                <1>       mov    edx, eax
3507 000068E5 01CA                <1>       add    edx, ecx
3508 000068E7 724B                <1>       jc     short a_u_im_retn
3509 000068E9 39D5                <1>       cmp    ebp, edx
3510 000068EB 7247                <1>       jb     short a_u_im_retn
3511 000068ED 89DA                <1>       mov    edx, ebx
3512 000068EF 81C200004000        <1>       add    edx, CORE
3513 000068F5 723D                <1>       jc     short a_u_im_retn
3514 000068F7 01CA                <1>       add    edx, ecx
3515 000068F9 7239                <1>       jc     short a_u_im_retn
3516 000068FB 39D5                <1>       cmp    ebp, edx
3517 000068FD 7235                <1>       jb     short a_u_im_retn
3518                              <1>       ;
3519 000068FF 89C5                <1>       mov    ebp, eax ; physical address
3520 00006901 89DE                <1>       mov    esi, ebx
3521 00006903 81C600004000        <1>       add    esi, CORE ; start of user's memory (4M)
3522 00006909 C1EE0C              <1>       shr    esi, PAGE_SHIFT ; higher 20 bits of the linear address
3523                              <1>       ;shr   ecx, PAGE_SHIFT ; page count
3524 0000690C 8B1D[B8030300]      <1>       mov    ebx, [u.pgdir] ; physical addr of user's page dir
3525 00006912 89F7                <1>       mov    edi, esi
3526 00006914 81E7FF030000        <1>       and    edi, PTE_MASK ; PTE entry index in the page table
3527 0000691A 57                  <1>       push   edi  ; * ; PTE index (in page directory)
3528 0000691B C1EE0A              <1>       shr    esi, PAGE_D_SHIFT - PAGE_SHIFT ; 22-12=10
3529 0000691E 89F2                <1>       mov    edx, esi
3530                              <1>       ; EDX = PDE index
3531 00006920 C1E602              <1>       shl    esi, 2 ; convert PDE index to dword offset
3532 00006923 01DE                <1>       add    esi, ebx ; add page directory address
3533                              <1> a_u_pd_0:
3534 00006925 AD                  <1>       lodsd
3535                              <1>       ;
3536 00006926 89F3                <1>       mov    ebx, esi ; next PDE address
3537                              <1>       ;
3538 00006928 A801                <1>       test   al, PDE_A_PRESENT ; bit 0, present flag (must be 1)
3539 0000692A 7513                <1>       jnz    short a_u_pd_2
3540                              <1>       ;
3541                              <1>       ; empty PDE (it does not point to valid page table address)
3542 0000692C E8A2F2FFFF          <1>       call   allocate_page  ; (allocate a new page table)
3543 00006931 7302                <1>       jnc    short a_u_pd_1 ; OK... now, we have a new page table.
3544                              <1>       ; cf = 1
3545                              <1>       ; There is not a free memory page to allocate a new page table !!!
3546 00006933 5E                  <1>       pop    esi ; *
3547                              <1> a_u_im_retn:
3548 00006934 C3                  <1>       retn   ; return to 'sysalloc' with 'insufficient memory' error
3549                              <1>       ;
3550                              <1> a_u_pd_1: ; clear the new page table content
3551                              <1>       ; EAX = Physical (base) address of the new page table
3552 00006935 E813F3FFFF          <1>       call   clear_page ; Clear page content
3553                              <1>       ;
3554 0000693A 0C07                <1>       or     al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
3555                              <1>              ; set bit 0, bit 1 and bit 2 to 1
3556                              <1>              ; (present, writable, user)
3557 0000693C 8946FC              <1>       mov    [esi-4], eax
3558                              <1> a_u_pd_2:
3559 0000693F 662500F0            <1>       and    ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3560                              <1>       ; EAX = PHYSICAL (flat) ADDRESS OF THE PAGE TABLE
3561 00006943 8B3C24              <1>       mov    edi, [esp] ; *
3562                              <1>       ; EDI = PTE index (of page directory)
3563                              <1>       ;and   edi, PTE_MASK ; PTE entry index in the page table
3564                              <1>       ; EBX = next PDE address
3565 00006946 89FE                <1>       mov    esi, edi ; PTE index in page table (0-1023)
3566 00006948 C1E702              <1>       shl    edi, 2 ; convert PTE index to dword offset
3567 0000694B 01C7                <1>       add    edi, eax ; now, edi points to requested PTE
3568                              <1> a_u_pt_0:
3569                              <1>       ; 02/05/2017
3570 0000694D 8B07                <1>       mov    eax, [edi]
3571                              <1>       ;
3572 0000694F A801                <1>       test   al, PTE_A_PRESENT ; bit 0, present flag (must be 1)
3573 00006951 7445                <1>       jz     short a_u_pt_1
3574                              <1>       ;
3575 00006953 A802                <1>       test   al, PTE_A_WRITE   ; bit 1, writable (r/w) flag
3576                              <1>                                ; (must be 1)
3577 00006955 7550                <1>       jnz    short a_u_pt_3
3578                              <1>       ; Read only -duplicated- page (belongs to a parent or a child)
3579 00006957 66A90002            <1>         test    ax, PTE_DUPLICATED ; Was this page duplicated
3580                              <1>                                    ; as child's page ?
3581 0000695B 7455                <1>       jz     short a_u_pt_4      ; Clear PTE but don't deallocate the page!
3582                              <1>       ;
3583                              <1>       ; check the parent's PTE value is read only & same page or not..
3584                              <1>       ; EDX = page directory entry index (0-1023)
3585 0000695D 52                  <1>       push   edx ; **
3586 0000695E 53                  <1>       push   ebx ; ***
3587                              <1>       ; ESI = page table entry index (0-1023)
3588                              <1>       ;push  esi ; **** ; 20/05/2017
3589 0000695F 8B1D[BC030300]      <1>       mov    ebx, [u.ppgdir] ; page directory of the parent process
3590 00006965 66C1E202            <1>       shl    dx, 2 ; *4
3591 00006969 01D3                <1>       add    ebx, edx ; PTE address,0 (for the parent)
3592 0000696B 8B13                <1>       mov    edx, [ebx] ; page table address
3593 0000696D F6C201              <1>       test   dl, PDE_A_PRESENT ; present (valid) or not ?
3594 00006970 7433                <1>       jz     short a_u_pt_2      ; parent process does not use this page
3595 00006972 6681E200F0          <1>       and    dx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
3596 00006977 66C1E602            <1>       shl    si, 2 ; *4
3597                              <1>       ; ESI = page table entry offset (0-4092)
3598 0000697B 01D6                <1>       add    esi, edx     ; PTE address (for the parent)
3599 0000697D 8B1E                <1>       mov    ebx, [esi]
3600 0000697F F6C301              <1>       test   bl, PTE_A_PRESENT ; present or not ?
3601 00006982 7421                <1>       jz     short a_u_pt_2      ; parent process does not use this page
3602 00006984 662500F0            <1>       and    ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3603 00006988 6681E300F0          <1>       and    bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3604 0000698D 39D8                <1>       cmp    eax, ebx     ; parent's and child's pages are same ?
3605 0000698F 7514                <1>       jne    short a_u_pt_2      ; not same page
3606                              <1>                                ; deallocate the child's page
```

```
3607 00006991 800E02              <1>         or      byte [esi], PTE_A_WRITE ; convert to writable page (parent)
3608                              <1>       ;pop   esi ; **** ; 20/05/2017
3609 00006994 5B                  <1>       pop    ebx ; ***
3610 00006995 5A                  <1>       pop    edx ; **
3611 00006996 EB1A                <1>       jmp    short a_u_pt_4
3612                              <1> a_u_pt_1:
3613 00006998 09C0                <1>       or     eax, eax     ; swapped page ?
3614 0000699A 7416                <1>       jz     short a_u_pt_4      ; no
3615                              <1>                                  ; yes
3616 0000699C D1E8                <1>       shr    eax, 1
3617 0000699E E8B5F8FFFF          <1>       call   unlink_swap_block ; Deallocate swapped page block
3618                              <1>                                ; on the swap disk (or in file)
3619 000069A3 EB0D                <1>       jmp    short a_u_pt_4
3620                              <1> a_u_pt_2:
3621                              <1>       ;pop   esi ; **** ; 20/05/2017
3622 000069A5 5B                  <1>       pop    ebx ; ***
3623 000069A6 5A                  <1>       pop    edx ; **
3624                              <1> a_u_pt_3:
3625 000069A7 66A90004            <1>       test   ax, PTE_SHARED      ; shared or direct memory access indicator
3626 000069AB 7505                <1>       jnz    short a_u_pt_4      ; AVL bit 1 = 1, do not deallocate this page!
3627                              <1>       ;
3628                              <1>       ;and   ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3629 000069AD E8FFF3FFFF          <1>       call   deallocate_page ; set the mem allocation bit of this page
3630                              <1>       ;
3631                              <1> a_u_pt_4:
3632 000069B2 89E8                <1>       mov    eax, ebp ; physical address
3633 000069B4 0C07                <1>       or     al, PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER ; 04/03/2017
3634 000069B6 AB                  <1>       stosd
3635 000069B7 5E                  <1>       pop    esi ; * ; 20/05/2017
3636 000069B8 49                  <1>       dec    ecx ; remain page count
3637 000069B9 7417                <1>       jz     short a_u_pd_5
3638 000069BB 81C500100000        <1>       add    ebp, PAGE_SIZE
3639 000069C1 46                  <1>       inc    esi ; next PTE (index)
3640                              <1>       ; 20/05/2017
3641                              <1>       ;cmp   esi, PAGE_SIZE/4 ; 1024
3642                              <1>       ;jb    short a_u_pt_0
3643 000069C2 6681E6FF03          <1>       and    si, PTE_MASK ; 3FFh (0 to 1023)
3644 000069C7 56                  <1>       push   esi ; *
3645 000069C8 7583                <1>       jnz    short a_u_pt_0 ; > 0 (<1024)
3646                              <1> a_u_pd_3:
3647 000069CA 42                  <1>       inc    edx
3648                              <1> ;     cmp    edx, 1024
3649                              <1> ;     jnb    short a_u_pd_4 ; 02/05/2017 (error!, ecx > 0)
3650 000069CB 89DE                <1>       mov    esi, ebx ; the next PDE address
3651 000069CD E953FFFFFF          <1>       jmp    a_u_pd_0
3652                              <1> a_u_pd_4:
3653                              <1>       ; 02/05/2017
3654                              <1> ;     stc
3655                              <1> a_u_pd_5:
3656                              <1>       ; 20/05/2017
3657                              <1>       ;pop   edi ; *
3658 000069D2 C3                  <1>       retn
3659                              <1>
3660                              <1> allocate_lfb_pages_for_kernel:
3661                              <1>       ; 15/12/2020
3662                              <1>       ; 14/12/2020 - TRDOS 386 v2.0.3
3663                              <1>       ; Set kernel page tables for linear frame buffer
3664                              <1>       ; (this procedure will be called by kernel only)
3665                              <1>       ;
3666                              <1>       ; Input:
3667                              <1>       ;     [LFB_ADDR] = linear frame buffer base address
3668                              <1>       ;     [LFB_SIZE] = linear frame buffer size in bytes
3669                              <1>       ; Output:
3670                              <1>       ;     none
3671                              <1>       ;     cf = 1 -> error
3672                              <1>       ;
3673                              <1>       ; Modified registers: eax, ecx, edx, edi
3674                              <1>
3675 000069D3 8B3D[2C120300]      <1>       mov    edi, [LFB_ADDR]
3676 000069D9 8B15[30120300]      <1>       mov    edx, [LFB_SIZE]
3677                              <1>
3678 000069DF C1EF16              <1>       shr    edi, 22      ; convert address to page number
3679                              <1>                           ; and then convert it to PDE entry offset
3680                              <1>                           ; (1 PDE is for 4MB, 22 bit shift)
3681                              <1>
3682 000069E2 66C1E702            <1>       shl    di, 2  ; * 4 for offset
3683                              <1>
3684                              <1>       ;add   edx, 4095
3685 000069E6 C1EA0C              <1>       shr    edx, 12      ; convert LFB size to LFB page count
3686                              <1>
3687 000069E9 89D1                <1>       mov    ecx, edx ; * ; LFB page count
3688                              <1>
3689 000069EB 81C1FF030000        <1>       add    ecx, 1023 ; page count + 1023
3690 000069F1 C1E90A              <1>       shr    ecx, 10 ; convert to page directory entry count
3691                              <1>                      ; (page table count)
3692 000069F4 51                  <1>       push   ecx ; **
3693 000069F5 C1E10C              <1>       shl    ecx, 12 ; convert to byte count
3694                              <1>
3695 000069F8 31C0                <1>       xor    eax, eax ; first available pages
3696                              <1>
3697                              <1>       ; allocate contiguous memory block for these kernel pages
3698                              <1>
3699 000069FA E87EFAFFFF          <1>       call   allocate_memory_block
3700                              <1>       ; eax = start address of (contiguous) memory block
3701 000069FF 59                  <1>       pop    ecx ; ** ; PDE count
3702 00006A00 7301                <1>       jnc    short a_lfb_k_1
3703                              <1>       ; error (cf=1)
3704 00006A02 C3                  <1>       retn
3705                              <1> a_lfb_k_1:
3706                              <1>       ; Allocate (new) page tables in kernel's page directory
3707 00006A03 51                  <1>       push   ecx ; PDE (page table) count
3708 00006A04 50                  <1>       push   eax ; start address of contiguous memory pages
3709                              <1>                  ; (at page boundary)
3710                              <1>       ; edi = 1st page directory entry offset
3711 00006A05 033D[C0890100]      <1>       add    edi, [k_page_dir] ; Kernel's Page Dir Address
```

```
3712                            <1> a_lfb_k_2:
3713 00006A0B 660D0304          <1>         or      ax, PDE_A_PRESENT + PDE_A_WRITE + PDE_EXTERNAL
3714                            <1>                                 ; supervisor + read&write + present
3715                            <1>                                 ; + external memory block (LFB)
3716 00006A0F AB                <1>         stosd
3717 00006A10 0500100000        <1>         add     eax, 4096
3718 00006A15 E2F4              <1>         loop    a_lfb_k_2
3719                            <1>
3720 00006A17 5F                <1>         pop     edi ; start addr of contiguous memory pages
3721 00006A18 59                <1>         pop     ecx ; page table (PDE) count
3722                            <1>
3723                            <1>         ; Allocate pages in (new) kernel page tables
3724                            <1>
3725                            <1>         ; (Note: page tables are contiguous in pyhsical memory)
3726 00006A19 C1E10A            <1>         shl     ecx, 10 ; * 1024, convert to (total) PTE count
3727                            <1>
3728 00006A1C A1[2C120300]      <1>         mov     eax, [LFB_ADDR]
3729                            <1>         ; edx = LFB page count
3730                            <1>         ;and    ax, ~4095  ; lw of LFB address is 0
3731                            <1> a_lfb_k_3:
3732 00006A21 660D0304          <1>         or      ax, PTE_A_PRESENT + PTE_A_WRITE + PTE_EXTERNAL
3733                            <1>                                 ; supervisor + read&write + present
3734                            <1>                                 ; + external memory block (LFB)
3735 00006A25 AB                <1>         stosd
3736 00006A26 4A                <1>         dec     edx
3737 00006A27 7408              <1>         jz      short a_lfb_k_4 ; LFB size has been completed (!?)
3738 00006A29 0500100000        <1>         add     eax, 4096
3739 00006A2E E2F1              <1>         loop    a_lfb_k_3
3740                            <1>
3741 00006A30 C3                <1>         retn
3742                            <1>
3743                            <1> a_lfb_k_4:
3744                            <1>         ; clear PTEs for empty/free pages
3745                            <1>         ;       (if there are after LFB !?)
3746 00006A31 31C0              <1>         xor     eax, eax ; clear page table entry (empty)
3747 00006A33 F3AB              <1>         rep     stosd
3748 00006A35 C3                <1>         retn
3749                            <1>
3750                            <1> ;deallocate_lfb_pages_for_kernel:
3751                            <1>         ; 15/12/2020
3752                            <1>         ; 14/12/2020 - TRDOS 386 v2.0.3
3753                            <1>         ; Reset/Release kernel page tables
3754                            <1>         ;       which are used for linear frame buffer
3755                            <1>         ; (this procedure will be called by kernel only)
3756                            <1>         ;
3757                            <1>         ; Input:
3758                            <1>         ;       [LFB_ADDR] = linear frame buffer base address
3759                            <1>         ;       [FFB_SIZE] = linear frame buffer size in bytes
3760                            <1>         ; Output:
3761                            <1>         ;       none
3762                            <1>         ;
3763                            <1>         ; Modified registers: eax, ecx, edi
3764                            <1>
3765                            <1>         ;mov   edi, [LFB_ADDR]
3766                            <1>         ;mov   ecx, [LFB_SIZE]
3767                            <1>         ;
3768                            <1>         ;shr   edi, 22      ; convert address to page number
3769                            <1>         ;           ; and then convert it to PDE entry offset
3770                            <1>         ;           ; (1 PDE is for 4MB, 22 bit shift)
3771                            <1>         ;
3772                            <1>         ;shl   di, 2  ; * 4 for offset
3773                            <1>         ;
3774                            <1>         ;;add  ecx, 4095
3775                            <1>         ;shr   ecx, 12      ; convert LFB size to page count
3776                            <1>         ;
3777                            <1>         ;add   ecx, 1023 ; page count + 1023
3778                            <1>         ;shr   ecx, 10 ; convert to page directory entry count
3779                            <1>         ;           ; (page table count)
3780                            <1>         ;push  ecx ; *
3781                            <1>         ;shl   ecx, 12 ; convert to byte count
3782                            <1>         ;
3783                            <1>         ;xor   eax, eax ; first available pages
3784                            <1>         ;
3785                            <1>         ;; deallocate contiguous memory block for kernel pages
3786                            <1>         ;
3787                            <1>         ;call  deallocate_memory_block
3788                            <1>         ;
3789                            <1>         ;pop   ecx ; * ; PDE count
3790                            <1>         ;
3791                            <1>         ;; Release/Free PDEs (page tables) in kernel's page dir
3792                            <1>         ;; edi = 1st page directory entry offset
3793                            <1>         ;add   edi, [k_page_dir] ; Kernel's Page Dir Address
3794                            <1>         ;sub   eax, eax ; clear (also invalidate)
3795                            <1>         ;rep   stosd
3796                            <1>         ;
3797                            <1>         ;retn
3798                            <1>
3799                            <1> ; /// End Of MEMORY MANAGEMENT FUNCTIONS ///
3800                            <1>
3801                            <1> ;; Data:
3802                            <1>
3803                            <1> ; 09/03/2015
3804                            <1> ;swpq_count: dw 0 ; count of pages on the swap que
3805                            <1> ;swp_drv:    dd 0 ; logical drive description table address of the swap drive/disk
3806                            <1> ;swpd_size:  dd 0 ; size of swap drive/disk (volume) in sectors (512 bytes).

3807                            <1> ;swpd_free:  dd 0 ; free page blocks (4096 bytes) on swap disk/drive (logical)
3808                            <1> ;swpd_next:  dd 0 ; next free page block
3809                            <1> ;swpd_last:  dd 0 ; last swap page block
2888                                %include 'timer.s'   ; 17/01/2015
   1                            <1> ; ****************************************************************************
   2                            <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - timer.s
   3                            <1> ; ----------------------------------------------------------------------------
   4                            <1> ; Last Update: 15/01/2017
   5                            <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)
00006A36 9C             <1>       pushfd
00006A37 0E             <1>       push   cs
00006A38 E801000000     <1>       call   TIME_OF_DAY_1
00006A3D 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
00006A3E 80642408FE             <1>       and    byte [esp+8], 11111110b   ; clear carry bit of eflags register
                                <1>       ;
00006A43 80FC08                 <1>       cmp    ah, (RTC_TBE-RTC_TB)/4    ; CHECK IF COMMAND IN VALID RANGE (0-7)
00006A46 F5                     <1>       cmc                       ; COMPLEMENT CARRY FOR ERROR EXIT
                                <1>       ; (*) jc short TIME_9           ; EXIT WITH CARRY = 1 IF NOT VALID
00006A47 721A                   <1>       jc     short _TIME_9 ; 29/05/2016
                                <1>
00006A49 1E                     <1>       push   ds
00006A4A 56                     <1>       push   esi
00006A4B 66BE1000               <1>       mov    si, KDATA          ; kernel data segment
00006A4F 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>       ;
00006A51 C0E402                 <1>       shl    ah, 2              ; convert function to dword offset
00006A54 0FB6F4                 <1>       movzx  esi, ah                  ; PLACE INTO ADDRESSING REGISTER
                                <1>       ;cli                       ; NO INTERRUPTS DURING TIME FUNCTIONS
00006A57 FF96[696A0000]         <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
00006A5D B400                   <1>       mov    ah, 0               ; CLEAR (AH) TO ZERO
00006A5F 5E                     <1>       pop    esi                 ; RECOVER USERS REGISTER
00006A60 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
00006A61 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
00006A63 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:
00006A68 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:
00006A69 [896A0000]             <1>       dd     RTC_00              ; 0 = READ CURRENT CLOCK COUNT
00006A6D [9C6A0000]             <1>       dd     RTC_10              ; 1 = SET CLOCK COUNT
00006A71 [AA6A0000]             <1>       dd     RTC_20              ; 2 = READ THE REAL TIME CLOCK TIME
00006A75 [D96A0000]             <1>       dd     RTC_30              ; 3 = SET REAL TIME CLOCK TIME
00006A79 [1B6B0000]             <1>       dd     RTC_40              ; 4 = READ THE REAL TIME CLOCK DATE
00006A7D [486B0000]             <1>       dd     RTC_50              ; 5 = SET REAL TIME CLOCK DATE
00006A81 [956B0000]             <1>       dd     RTC_60              ; 6 = SET THE REAL TIME CLOCK ALARM
00006A85 [E86B0000]             <1>       dd     RTC_70              ; 7 = RESET ALARM
                                <1>
                                <1> RTC_TBE      equ    $
                                <1>
                                <1> RTC_00:                          ; READ TIME COUNT
00006A89 A0[448A0100]           <1>       mov    al, [TIMER_OFL]           ; GET THE OVERFLOW FLAG
00006A8E C605[448A0100]00       <1>       mov    byte [TIMER_OFL], 0 ; AND THEN RESET THE OVERFLOW FLAG
00006A95 8B0D[408A0100]         <1>         mov     ecx, [TIMER_LH]          ; GET COUNT OF TIME
00006A9B C3                     <1>       retn
                                <1>
                                <1> RTC_10:                          ; SET TIME COUNT
00006A9C 890D[408A0100]         <1>         mov     [TIMER_LH], ecx         ; SET TIME COUNT
00006AA2 C605[448A0100]00       <1>       mov    byte [TIMER_OFL], 0 ; RESET OVERFLOW FLAG
00006AA9 C3                     <1>       retn                        ; RETURN WITH NO CARRY
                                <1>
                                <1> RTC_20:                          ; GET RTC TIME
00006AAA E8EB010000             <1>       call   UPD_IPR                   ; CHECK FOR UPDATE IN PROCESS
00006AAF 7227                   <1>       jc     short RTC_29         ; EXIT IF ERROR (CY= 1)
                                <1>
00006AB1 B000                   <1>       mov    al, CMOS_SECONDS     ; SET ADDRESS OF SECONDS
00006AB3 E8FD010000             <1>       call   CMOS_READ            ; GET SECONDS
00006AB8 88C6                   <1>       mov    dh, al               ; SAVE
00006ABA B00B                   <1>       mov    al, CMOS_REG_B            ; ADDRESS ALARM REGISTER
00006ABC E8F4010000             <1>       call   CMOS_READ            ; READ CURRENT VALUE OF DSE BIT
00006AC1 2401                   <1>       and    al, 00000001b        ; MASK FOR VALID DSE BIT
00006AC3 88C2                   <1>       mov    dl, al               ; SET [DL] TO ZERO FOR NO DSE BIT
00006AC5 B002                   <1>       mov    al, CMOS_MINUTES     ; SET ADDRESS OF MINUTES
00006AC7 E8E9010000             <1>       call   CMOS_READ            ; GET MINUTES
00006ACC 88C1                   <1>       mov    cl, al               ; SAVE
```

```
                        <1>         mov     al, CMOS_HOURS          ; SET ADDRESS OF HOURS
00006ACE B004           <1>       call   CMOS_READ          ; GET HOURS
00006AD0 E8E0010000     <1>       mov    ch, al             ; SAVE
00006AD5 88C5           <1>       clc                       ; SET CY= 0
00006AD7 F8             <1> RTC_29:
                        <1>       retn                      ; RETURN WITH RESULT IN CARRY FLAG
00006AD8 C3             <1>
                        <1> RTC_30:                         ; SET RTC TIME
00006AD9 E8BC010000     <1>       call   UPD_IPR                  ; CHECK FOR UPDATE IN PROCESS
00006ADE 7305           <1>       jnc    short RTC_35       ; GO AROUND IF CLOCK OPERATING
00006AE0 E817010000     <1>       call   RTC_STA                  ; ELSE TRY INITIALIZING CLOCK
                        <1> RTC_35:
00006AE5 88F4           <1>       mov    ah, dh             ; GET TIME BYTE - SECONDS
00006AE7 B000           <1>       mov    al, CMOS_SECONDS   ; ADDRESS SECONDS
00006AE9 E8E0010000     <1>       call   CMOS_WRITE         ; UPDATE SECONDS
00006AEE 88CC           <1>       mov    ah, cl             ; GET TIME BYTE - MINUTES
00006AF0 B002           <1>       mov    al, CMOS_MINUTES   ; ADDRESS MINUTES
00006AF2 E8D7010000     <1>       call   CMOS_WRITE         ; UPDATE MINUTES
00006AF7 88EC           <1>       mov    ah, ch             ; GET TIME BYTE - HOURS
00006AF9 B004           <1>       mov    al, CMOS_HOURS          ; ADDRESS HOURS
00006AFB E8CE010000     <1>       call   CMOS_WRITE         ; UPDATE ADDRESS
                        <1>       ;mov   al, CMOS_REG_B          ; ADDRESS ALARM REGISTER
                        <1>       ;mov   ah, al
00006B00 66B80B0B       <1>       mov    ax, CMOS_REG_B * 257
00006B04 E8AC010000     <1>       call   CMOS_READ          ; READ CURRENT TIME
00006B09 2462           <1>       and    al, 01100010b      ; MASK FOR VALID BIT POSITIONS
00006B0B 0C02           <1>       or     al, 00000010b      ; TURN ON 24 HOUR MODE
00006B0D 80E201         <1>       and    dl, 00000001b      ; USE ONLY THE DSE BIT
00006B10 08D0           <1>       or     al, dl             ; GET DAY LIGHT SAVINGS TIME BIT (OSE)
00006B12 86E0           <1>       xchg   ah, al             ; PLACE IN WORK REGISTER AND GET ADDRESS
00006B14 E8B5010000     <1>       call   CMOS_WRITE         ; SET NEW ALARM SITS
00006B19 F8             <1>       clc                       ; SET CY= 0
00006B1A C3             <1>       retn                      ; RETURN WITH CY= 0
                        <1>
                        <1> RTC_40:                         ; GET RTC DATE
00006B1B E87A010000     <1>       call   UPD_IPR                  ; CHECK FOR UPDATE IN PROCESS
00006B20 7225           <1>       jc     short RTC_49       ; EXIT IF ERROR (CY= 1)
                        <1>
00006B22 B007           <1>       mov    al, CMOS_DAY_MONTH ; ADDRESS DAY OF MONTH
00006B24 E88C010000     <1>       call   CMOS_READ          ; READ DAY OF MONTH
00006B29 88C2           <1>       mov    dl, al             ; SAVE
00006B2B B008           <1>       mov    al, CMOS_MONTH          ; ADDRESS MONTH
00006B2D E883010000     <1>       call   CMOS_READ          ; READ MONTH
00006B32 88C6           <1>       mov    dh, al             ; SAVE
00006B34 B009           <1>       mov    al, CMOS_YEAR      ; ADDRESS YEAR
00006B36 E87A010000     <1>       call   CMOS_READ          ; READ YEAR
00006B3B 88C1           <1>       mov    cl, al             ; SAVE
00006B3D B032           <1>       mov    al, CMOS_CENTURY   ; ADDRESS CENTURY LOCATION
00006B3F E871010000     <1>       call   CMOS_READ          ; GET CENTURY BYTE
00006B44 88C5           <1>       mov    ch, al             ; SAVE
00006B46 F8             <1>       clc                       ; SET CY=0
                        <1> RTC_49:
00006B47 C3             <1>       retn                      ; RETURN WITH RESULTS IN CARRY FLAG
                        <1>
                        <1> RTC_50:                         ; SET RTC DATE
00006B48 E84D010000     <1>       call   UPD_IPR                  ; CHECK FOR UPDATE IN PROCESS
00006B4D 7305           <1>       jnc    short RTC_55       ; GO AROUND IF NO ERROR
00006B4F E8A8000000     <1>       call   RTC_STA                  ; ELSE INITIALIZE CLOCK
                        <1> RTC_55:
00006B54 66B80600       <1>       mov    ax, CMOS_DAY_WEEK  ; ADDRESS OF DAY OF WEEK BYTE
00006B58 E871010000     <1>       call   CMOS_WRITE         ; LOAD ZEROS TO DAY OF WEEK
00006B5D 88D4           <1>       mov    ah, dl             ; GET DAY OF MONTH BYTE
00006B5F B007           <1>       mov    al, CMOS_DAY_MONTH ; ADDRESS DAY OF MONTH BYTE
00006B61 E868010000     <1>       call   CMOS_WRITE         ; WRITE OF DAY OF MONTH REGISTER
00006B66 88F4           <1>       mov    ah, dh             ; GET MONTH
00006B68 B008           <1>       mov    al, CMOS_MONTH          ; ADDRESS MONTH BYTE
00006B6A E85F010000     <1>       call   CMOS_WRITE         ; WRITE MONTH REGISTER
00006B6F 88CC           <1>       mov    ah, cl             ; GET YEAR BYTE
00006B71 B009           <1>       mov    al, CMOS_YEAR      ; ADDRESS YEAR REGISTER
00006B73 E856010000     <1>       call   CMOS_WRITE         ; WRITE YEAR REGISTER
00006B78 88EC           <1>       mov    ah, ch             ; GET CENTURY BYTE
00006B7A B032           <1>       mov    al, CMOS_CENTURY   ; ADDRESS CENTURY BYTE
00006B7C E84D010000     <1>       call   CMOS_WRITE         ; WRITE CENTURY LOCATION
                        <1>       ;mov   al, CMOS_REG_B          ; ADDRESS ALARM REGISTER
                        <1>       ;mov   ah, al
00006B81 66B80B0B       <1>       mov    ax, CMOS_REG_B * 257
00006B85 E82B010000     <1>       call   CMOS_READ          ; READ WIRRENT SETTINGS
00006B8A 247F           <1>       and    al, 07Fh           ; CLEAR 'SET BIT'
00006B8C 86E0           <1>       xchg   ah, al             ; MOVE TO WORK REGISTER
00006B8E E83B010000     <1>       call   CMOS_WRITE         ; AND START CLOCK UPDATING
00006B93 F8             <1>       clc                       ; SET CY= 0
00006B94 C3             <1>       retn                      ; RETURN CY=0
                        <1>
                        <1> RTC_60:                         ; SET RTC ALARM
00006B95 B00B           <1>       mov    al, CMOS_REG_B          ; ADDRESS ALARM
00006B97 E819010000     <1>       call   CMOS_READ          ; READ ALARM REGISTER
00006B9C A820           <1>       test   al, 20h                  ; CHECK FOR ALARM ALREADY ENABLED
00006B9E F9             <1>       stc                       ; SET CARRY IN CASE OF ERROR
00006B9F 7542           <1>       jnz    short RTC_69       ; ERROR EXIT IF ALARM SET
00006BA1 E8F4000000     <1>       call   UPD_IPR                  ; CHECK FOR UPDATE IN PROCESS
00006BA6 7305           <1>       jnc    short RTC_65       ; SKIP INITIALIZATION IF NO ERROR
00006BA8 E84F000000     <1>       call   RTC_STA                  ; ELSE INITIALIZE CLOCK
                        <1> RTC_65:
00006BAD 88F4           <1>       mov    ah, dh             ; GET SECONDS BYTE
00006BAF B001           <1>       mov    al, CMOS_SEC_ALARM ; ADDRESS THE SECONDS ALARM REGISTER
00006BB1 E818010000     <1>       call   CMOS_WRITE         ; INSERT SECONDS
00006BB6 88CC           <1>       mov    ah, cl             ; GET MINUTES PARAMETER
00006BB8 B003           <1>       mov    al, CMOS_MIN_ALARM ; ADDRESS MINUTES ALARM REGISTER
00006BBA E80F010000     <1>       call   CMOS_WRITE         ; INSERT MINUTES
00006BBF 88EC           <1>       mov    ah, ch             ; GET HOURS PARAMETER
00006BC1 B005           <1>       mov    al, CMOS_HR_ALARM  ; ADDRESS HOUR ALARM REGISTER
00006BC3 E806010000     <1>       call   CMOS_WRITE         ; INSERT HOURS
00006BC8 E4A1           <1>       in     al, INTB01         ; READ SECOND INTERRUPT MASK REGISTER
00006BCA 24FE           <1>       and    al, 0FEh           ; ENABLE ALARM TIMER BIT (CY= 0)
00006BCC E6A1           <1>       out    INTB01, al         ; WRITE UPDATED MASK
```

```
                        <1>       ;mov   al, CMOS_REG_B           ; ADDRESS ALARM REGISTER
                        <1>       ;mov   ah, al
00006BCE 66B80B0B       <1>       mov    ax, CMOS_REG_B * 257
00006BD2 E8DE000000     <1>       call   CMOS_READ          ; READ CURRENT ALARM REGISTER
00006BD7 247F           <1>       and    al, 07Fh           ; ENSURE SET BIT TURNED OFF
00006BD9 0C20           <1>       or     al, 20h                 ; TURN ON ALARM ENABLE
00006BDB 86E0           <1>       xchg   ah, al             ; MOVE MASK TO OUTPUT REGISTER
00006BDD E8EC000000     <1>       call   CMOS_WRITE         ; WRITE NEW ALARM MASK
00006BE2 F8             <1>       clc                       ; SET CY= 0
                        <1> RTC_69:
00006BE3 66B80000       <1>       mov    ax, 0              ; CLEAR AX REGISTER
00006BE7 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
00006BE8 66B80B0B       <1>       mov    ax, CMOS_REG_B * 257     ; ADDRESS ALARM REGISTER (TO BOTH AH,AL)
00006BEC E8C4000000     <1>       call   CMOS_READ          ; READ ALARM REGISTER
00006BF1 2457           <1>       and    al, 57h                 ; TURN OFF ALARM ENABLE
00006BF3 86E0           <1>       xchg   ah, al             ; SAVE DATA AND RECOVER ADDRESS
00006BF5 E8D4000000     <1>       call   CMOS_WRITE         ; RESTORE NEW VALUE
00006BFA F8             <1>       clc                       ; SET CY= 0
00006BFB 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
00006BFC 66B80A26       <1>       mov    ax, (26h*100h)+CMOS_REG_A
00006C00 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
00006C05 66B80B82       <1>       mov    ax, (82h*100h)+CMOS_REG_B
00006C09 E8C0000000     <1>       call   CMOS_WRITE         ; AND 24 HOUR MODE TO REGISTER B
00006C0E B00C           <1>       mov    al, CMOS_REG_C           ; ADDRESS REGISTER C
00006C10 E8A0000000     <1>       call   CMOS_READ          ; READ REGISTER C TO INITIALIZE
00006C15 B00D           <1>       mov    al, CMOS_REG_D           ; ADDRESS REGISTER D
00006C17 E899000000     <1>       call   CMOS_READ          ; READ REGISTER D TO INITIALIZE
00006C1C 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
00006C1D 1E             <1>       push   ds                 ; LEAVE INTERRUPTS DISABLED
00006C1E 50             <1>       push   eax                ; SAVE REGISTERS
00006C1F 57             <1>       push   edi
                        <1> RTC_I_1:                        ; CHECK FOR SECOND INTERRUPT
00006C20 66B88C8B       <1>       mov    ax, 256*(CMOS_REG_B+NMI)+CMOS_REG_C+NMI ; ALARM AND STATUS
00006C24 E670           <1>       out    CMOS_PORT, al      ; WRITE ALARM FLAG MASK ADDRESS
00006C26 90             <1>       nop                       ; I/O DELAY
00006C27 EB00           <1>       jmp    short $+2
00006C29 E471           <1>       in     al, CMOS_DATA      ; READ AND RESET INTERRUPT REQUEST FLAGS
00006C2B A860           <1>       test   al, 01100000b      ; CHECK FOR EITHER INTERRUPT PENDING
00006C2D 745D           <1>       jz     short  RTC_I_9           ; EXIT IF NOT A VALID RTC INTERRUPT
                        <1>
00006C2F 86E0           <1>       xchg   ah, al             ; SAVE FLAGS AND GET ENABLE ADDRESS
00006C31 E670           <1>       out    CMOS_PORT, al      ; WRITE ALARM ENABLE MASK ADDRESS
00006C33 90             <1>       nop                       ; I/O DELAY
00006C34 EB00           <1>       jmp    short $+2
00006C36 E471           <1>       in     al, CMOS_DATA      ; READ CURRENT ALARM ENABLE MASK
00006C38 20E0           <1>       and    al, ah             ; ALLOW ONLY SOURCES THAT ARE ENABLED
00006C3A A840           <1>       test   al, 01000000b      ; CHECK FOR PERIODIC INTERRUPT
00006C3C 743B           <1>       jz     short RTC_I_5      ; SKIP IF NOT A PERIODIC INTERRUPT
                        <1>
                        <1> ;-----      DECREMENT WAIT COUNT BY INTERRUPT INTERVAL
                        <1>
00006C3E 66BF1000       <1>       mov    di, KDATA          ; kernel data segment
00006C42 8EDF           <1>       mov    ds, di
                        <1>
00006C44 812D[388A0100]D003- <1>  sub    dword [RTC_LH], 976 ; DECREMENT COUNT BY 1/1024
00006C4C 0000           <1>
00006C4E 7329           <1>       jnc    short RTC_I_5      ; SKIP TILL 32 BIT WORD LESS THAN ZERO
                        <1>
                        <1> ;-----      TURN OFF PERIODIC INTERRUPT ENABLE
                        <1>
00006C50 6650           <1>       push   ax                 ; SAVE INTERRUPT FLAG MASK
00006C52 66B88B8B       <1>       mov    ax, 257*(CMOS_REG_B+NMI) ; INTERRUPT ENABLE REGISTER
00006C56 E670           <1>       out    CMOS_PORT, al      ; WRITE ADDRESS TO CMOS CLOCK
00006C58 90             <1>       nop                       ; I/O DELAY
00006C59 EB00           <1>       jmp    short $+2
00006C5B E471           <1>       in     al, CMOS_DATA      ; READ CURRENT ENABLES
00006C5D 24BF           <1>       and    al, 0BFh           ; TURN OFF PIE
00006C5F 86C4           <1>       xchg   al, ah             ; GET CMOS ADDRESS AND SAVE VALUE
00006C61 E670           <1>       out    CMOS_PORT, al      ; ADDRESS REGISTER B
00006C63 86C4           <1>       xchg   al, ah             ; GET NEW INTERRUPT ENABLE MASK
00006C65 E671           <1>       out    CMOS_DATA, al      ; SET MASK IN INTERRUPT ENABLE REGISTER
00006C67 C605[3C8A0100]00    <1>  mov    byte [RTC_WAIT_FLAG], 0   ; SET FUNCTION ACTIVE FLAG OFF
00006C6E 8B3D[3D8A0100]      <1>  mov    edi, [USER_FLAG]    ; SET UP (DS:DI) TO POINT TO USER FLAG
00006C74 C60780              <1>  mov    byte [edi], 80h           ; TURN ON USERS FLAG
00006C77 6658                <1>  pop    ax                 ; GET INTERRUPT SOURCE BACK
```

```
                        <1> RTC_I_5:
00006C79 A820           <1>       test   al, 00100000b       ; TEST FOR ALARM INTERRUPT
00006C7B 740D           <1>       jz     short RTC_I_7       ; SKIP USER INTERRUPT CALL IF NOT ALARM
                        <1>
00006C7D B00D           <1>       mov    al, CMOS_REG_D            ; POINT TO DEFAULT READ ONLY REGISTER
00006C7F E670           <1>       out    CMOS_PORT, al       ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
00006C81 FB             <1>       sti                        ; INTERRUPTS BACK ON NOW
00006C82 52             <1>       push   edx
00006C83 E8C0C00000     <1>       call   INT4Ah              ; TRANSFER TO USER ROUTINE
00006C88 5A             <1>       pop    edx
00006C89 FA             <1>       cli                        ; BLOCK INTERRUPT FOR RETRY
                        <1> RTC_I_7:                       ; RESTART ROUTINE TO HANDLE DELAYED
00006C8A EB94           <1>       jmp    short RTC_I_1       ;  ENTRY AND SECOND EVENT BEFORE DONE
                        <1>
                        <1> RTC_I_9:                       ; EXIT - NO PENDING INTERRUPTS
00006C8C B00D           <1>       mov    al, CMOS_REG_D            ; POINT TO DEFAULT READ ONLY REGISTER
00006C8E E670           <1>       out    CMOS_PORT, al       ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
00006C90 B020           <1>       mov    al, EOI                   ; END OF INTERRUPT MASK TO 8259 - 2
00006C92 E6A0           <1>       out    INTB00, al          ; TO 8259 - 2
00006C94 E620           <1>       out    INTA00,      al           ; TO 8259 - 1
00006C96 5F             <1>       pop    edi                 ; RESTORE REGISTERS
00006C97 58             <1>       pop    eax
00006C98 1F             <1>       pop    ds
00006C99 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
00006C9A 51             <1>       push   ecx
                        <1>
                        <1>       ; 29/05/2016
00006C9B 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:
00006CA0 B00A           <1>       mov    al, CMOS_REG_A            ; ADDRESS STATUS REGISTER A
00006CA2 FA             <1>       cli                        ; NO TIMER INTERRUPTS DURING UPDATES
00006CA3 E80D000000     <1>       call   CMOS_READ           ; READ UPDATE IN PROCESS FLAG
00006CA8 A880           <1>       test   al, 80h                   ; IF UIP BIT IS ON ( CANNOT READ TIME )
00006CAA 7406           <1>       jz     short UPD_90        ; EXIT WITH CY= 0 IF CAN READ CLOCK NOW
00006CAC FB             <1>       sti                        ; ALLOW INTERRUPTS WHILE WAITING
00006CAD E2F1           <1>       loop   UPD_10              ; LOOP TILL READY OR TIMEOUT
00006CAF 31C0           <1>       xor    eax, eax            ; CLEAR RESULTS IF ERROR
                        <1>             ; xor ax, ax
00006CB1 F9             <1>       stc                        ; SET CARRY FOR ERROR
                        <1> UPD_90:
00006CB2 59             <1>       pop    ecx                 ; RESTORE CALLERS REGISTER
00006CB3 FA             <1>       cli                        ; INTERRUPTS OFF DURING SET
00006CB4 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:
00006CB5 9C             <1>       pushf                      ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
00006CB6 D0C0           <1>       rol    al, 1               ; MOVE NMI BIT TO LOW POSITION
00006CB8 F9             <1>       stc                        ; FORCE NMI BIT ON IN CARRY FLAG
00006CB9 D0D8           <1>       rcr    al, 1               ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
00006CBB FA             <1>       cli                        ; DISABLE INTERRUPTS
00006CBC E670           <1>       out    CMOS_PORT, al       ; ADDRESS LOCATION AND DISABLE NMI
                        <1>       ; 29/05/2016
                        <1>       ;nop                       ; I/O DELAY
00006CBE E6EB           <1>       out    0ebh,al      ; NEWIODELAY ; AWARD BIOS 1999, ATIME.ASM
                        <1>       ;
00006CC0 E471           <1>       in     al, CMOS_DATA       ; READ THE REQUESTED CMOS LOCATION
00006CC2 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)
00006CC4 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
00006CC6 D0D8           <1>       rcr    al, 1               ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
00006CC8 E670           <1>       out    CMOS_PORT, al       ; SET DEFAULT TO READ ONLY REGISTER
00006CCA 6658           <1>       pop    ax                  ; RESTORE (AH) AND (AL), CMOS BYTE
00006CCC 9D             <1>       popf
00006CCD 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)
00006CCE 9C             <1>       pushf                     ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
00006CCF 6650           <1>       push   ax                 ; SAVE WORK REGISTER VALUES
00006CD1 D0C0           <1>       rol    al, 1              ; MOVE NMI BIT TO LOW POSITION
00006CD3 F9             <1>       stc                       ; FORCE NMI BIT ON IN CARRY FLAG
00006CD4 D0D8           <1>       rcr    al, 1              ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
00006CD6 FA             <1>       cli                       ; DISABLE INTERRUPTS
00006CD7 E670           <1>       out    CMOS_PORT, al      ; ADDRESS LOCATION AND DISABLE NMI
00006CD9 88E0           <1>       mov    al, ah             ; GET THE DATA BYTE TO WRITE
00006CDB E671           <1>       out    CMOS_DATA, al      ; PLACE IN REQUESTED CMOS LOCATION
00006CDD 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
00006CDF D0D8           <1>       rcr    al, 1              ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
00006CE1 E670           <1>       out    CMOS_PORT, al      ; SET DEFAULT TO READ ONLY REGISTER
00006CE3 90             <1>       nop                       ; I/O DELAY
00006CE4 E471           <1>       in     al, CMOS_DATA      ; OPEN STANDBY LATCH
00006CE6 6658           <1>       pop    ax                 ; RESTORE WORK REGISTERS
00006CE8 9D             <1>       popf
00006CE9 C3             <1>       retn
                        <1>
                        <1> ; /// End Of TIMER FUNCTIONS ///
```

```
                        2889
00006CEA 90<rept>       2890 Align 16
                        2891
                        2892 gdt:  ; Global Descriptor Table
                        2893       ; 02/12/2020
                        2894       ; (30/07/2015, conforming cs)
                        2895       ; (26/03/2015)
                        2896       ; (24/03/2015, tss)
                        2897       ; (19/03/2015)
                        2898       ; (29/12/2013)
                        2899       ;
00006CF0 0000000000000000 2900       dw 0, 0, 0, 0       ; NULL descriptor
                        2901 gdt_kcode:
                        2902       ; 18/08/2014
                        2903                           ; 8h kernel code segment, base = 00000000h
                        2904       ;dw 0FFFFh, 0, 9E00h, 00CFh      ; KCODE  ; 30/12/2016
00006CF8 FFFF0000009ACF00 2905       dw 0FFFFh, 0, 9A00h, 00CFh; KCODE
                        2906 gdt_kdata:
                        2907                           ; 10h kernel data segment, base = 00000000h
00006D00 FFFF00000092CF00 2908       dw 0FFFFh, 0, 9200h, 00CFh; KDATA
                        2909 gdt_ucode:
                        2910                           ; 1Bh user code segment, base address = 400000h ; CORE
                        2911       ;dw 0FBFFh, 0, 0FE40h, 00CFh     ; UCODE  ; 30/12/2016
00006D08 FFFB000040FACF00 2912       dw 0FBFFh, 0, 0FA40h, 00CFh      ; UCODE
                        2913 gdt_udata:
                        2914                           ; 23h user data segment, base address = 400000h ; CORE
00006D10 FFFB000040F2CF00 2915       dw 0FBFFh, 0, 0F240h, 00CFh      ; UDATA
                        2916 gdt_tss:
                        2917                           ; Task State Segment
00006D18 6700           2918       dw 0067h ; Limit = 103 ; (104-1, tss size = 104 byte,
                        2919                                ;  no IO permission in ring 3)
                        2920 gdt_tss0:
00006D1A 0000           2921       dw 0  ; TSS base address, bits 0-15
                        2922 gdt_tss1:
00006D1C 00             2923       db 0  ; TSS base address, bits 16-23
                        2924                           ; 49h
00006D1D E9             2925       db 11101001b ; 0E9h => P=1/DPL=11/0/1/0/B/1 --> B = Task is busy (1)
00006D1E 00             2926       db 0 ; G/0/0/AVL/LIMIT=0000 ; (Limit bits 16-19 = 0000) (G=0, 1 byte)
                        2927 gdt_tss2:
00006D1F 00             2928       db 0  ; TSS base address, bits 24-31
                        2929
                        2930       ; 30/11/2020
                        2931       ; 29/11/2020 - TRDOS v2.0.3
                        2932       ; VESA VBE3 VIDE BIOS 32 BIT PMI SEGMENTS (16 bit segments)
                        2933                           ; 30h ; VBE3CS
                        2934 _vbe3_CS:  ; vesa vbe3 bios uses this as code seg (same addr with _vbe3_DS)
                        2935       ; limit = 65536, base addr = 0, P/DPL/1/Type/C/R/A = 9Ah, 16 bit
00006D20 FFFF0000009A0000 2936       dw 0FFFFh, 0, 9A00h, 0 ; Note: base addr will be initialized
                        2937                           ; 38h ; VBE3BDS
                        2938 _vbe3_BDS: ; vesa vbe3 bios uses this as equivalent of rombios data segment
                        2939       ; limit = 1536, base addr = 0, P/DPL/1/Type/E/W/A = 92h, 16 bit
00006D28 FF05000000920000 2940       dw 05FFh, 0, 9200h, 0 ; Note: base addr will be initialized
                        2941                           ; 40h ; VBE3A000
                        2942 _A0000Sel: ; VGA default video memory address
                        2943       ; limit = 65536, base addr = 0A0000h, 16 bit
00006D30 FFFF00000A920000 2944       dw 0FFFFh, 0, 920Ah, 0
                        2945                           ; 48h ; VBE3B000
                        2946 _B0000Sel: ; MDA (monochrome) video memory address
                        2947       ; limit = 65536, base addr = 0B0000h, 16 bit
00006D38 FFFF00000B920000 2948       dw 0FFFFh, 0, 920Bh, 0
                        2949                           ; 50h ; VBE3B800
                        2950 _B8000Sel: ; CGA video memory address
                        2951       ; limit = 32768, base addr = 0B8000h, 16 bit
00006D40 FF7F00800B920000 2952       dw 07FFFh, 8000h, 920Bh, 0
                        2953                           ; 58h ; VBE3DS
                        2954 _vbe3_DS: ; vesa vbe3 bios uses this as data seg (CodeSegSel in PMInfoBlock)
                        2955       ; limit = 65536, base addr = 0, P/DPL/1/Type/E/W/A = 92h, 16 bit
00006D48 FFFF000000920000 2956       dw 0FFFFh, 0, 9200h, 0 ; Note: base addr will be initialized
                        2957                           ; 60h ; VBE3SS
                        2958 _vbe3_SS: ; kernel's stack segment but 16 bit version (same stack addr)
                        2959       ; limit = 1024, base addr = 0, P/DPL/1/Type/E/W/A = 92h, 16 bit
00006D50 FF03000000920000 2960       dw 03FFh, 0, 9200h, 0 ; Note: base addr will be initialized
```

```
                                        ; 68h ; VBE3ES
                                _vbe3_ES: ; extra 16 bit segment points to buffers in kernel's mem space
                                        ; limit = 2048, base addr = 0, P/DPL/1/Type/E/W/A = 92h, 16 bit
00006D58 FF07000000920000               dw 07FFh, 0, 9200h,0 ; Note: base addr will be initialized
                                        ; 70h ; KODE16
                                _16bit_CS: ; 16 bit code segment points to kernel's far return addr
                                        ; limit = 16M, base addr = 0, P/DPL/1/Type/E/W/A = 92h, 16 bit
00006D60 FFFF0000009AFF00               dw 0FFFFh, 0, 9A00h, 00FFh ; Note: base addr will be initialized

                                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:
00006D68 7700                           dw gdt_end - gdt - 1    ; Limit (size)
00006D6A [F06C0000]                     dd gdt                  ; Address of the GDT

                                        ; 20/08/2014
                                idtd:
00006D6E 7F02                           dw idt_end - idt - 1    ; Limit (size)
00006D70 [D8860100]                     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>
00006D74 [D76D0000]           <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:
00006D78 01             <1>             DB      01              ;DRIVE TYPE, MEDIA TABLE
                        <1>                 ;DW      MD_TBL1
00006D79 [966D0000]     <1>             dd      MD_TBL1
00006D7D 82             <1>             DB      02+BIT7ON
                        <1>             ;DW      MD_TBL2
00006D7E [A36D0000]     <1>                 dd      MD_TBL2
00006D82 02             <1> DR_DEFAULT:  DB      02
                        <1>                 ;DW      MD_TBL3
00006D83 [B06D0000]     <1>             dd      MD_TBL3
00006D87 03             <1>             DB      03
                        <1>                 ;DW      MD_TBL4
00006D88 [BD6D0000]     <1>             dd      MD_TBL4
00006D8C 84             <1>             DB      04+BIT7ON
                        <1>                 ;DW      MD_TBL5
00006D8D [CA6D0000]     <1>             dd      MD_TBL5
00006D91 04             <1>             DB      04
                        <1>                 ;DW      MD_TBL6
00006D92 [D76D0000]     <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:
00006D96 DF             <1>         DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00006D97 02             <1>         DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00006D98 25             <1>         DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00006D99 02             <1>         DB      2               ; 512 BYTES/SECTOR
00006D9A 09             <1>         DB      09              ; EOT (LAST SECTOR ON TRACK)
00006D9B 2A             <1>         DB      02AH            ; GAP LENGTH
00006D9C FF             <1>         DB      0FFH            ; DTL
00006D9D 50             <1>         DB      050H            ; GAP LENGTH FOR FORMAT
00006D9E F6             <1>         DB      0F6H            ; FILL BYTE FOR FORMAT
```

```
00006D9F 0F                     <1>       DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
00006DA0 08                     <1>       DB     8            ; MOTOR START TIME (1/8 SECONDS)
00006DA1 27                     <1>       DB     39           ; MAX. TRACK NUMBER
00006DA2 80                     <1>       DB     RATE_250     ; DATA TRANSFER RATE
                                <1> ;--------------------------------------------------------
                                <1> ;     360 KB MEDIA IN 1.2 MB DRIVE                  :
                                <1> ;--------------------------------------------------------
                                <1> MD_TBL2:
00006DA3 DF                     <1>       DB     11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00006DA4 02                     <1>       DB     2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00006DA5 25                     <1>       DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00006DA6 02                     <1>       DB     2            ; 512 BYTES/SECTOR
00006DA7 09                     <1>       DB     09           ; EOT (LAST SECTOR ON TRACK)
00006DA8 2A                     <1>       DB     02AH         ; GAP LENGTH
00006DA9 FF                     <1>       DB     0FFH         ; DTL
00006DAA 50                     <1>       DB     050H         ; GAP LENGTH FOR FORMAT
00006DAB F6                     <1>       DB     0F6H         ; FILL BYTE FOR FORMAT
00006DAC 0F                     <1>       DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
00006DAD 08                     <1>       DB     8            ; MOTOR START TIME (1/8 SECONDS)
00006DAE 27                     <1>       DB     39           ; MAX. TRACK NUMBER
00006DAF 40                     <1>       DB     RATE_300     ; DATA TRANSFER RATE
                                <1> ;--------------------------------------------------------
                                <1> ;     1.2 MB MEDIA IN 1.2 MB DRIVE                  :
                                <1> ;--------------------------------------------------------
                                <1> MD_TBL3:
00006DB0 DF                     <1>       DB     11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00006DB1 02                     <1>       DB     2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00006DB2 25                     <1>       DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00006DB3 02                     <1>       DB     2            ; 512 BYTES/SECTOR
00006DB4 0F                     <1>       DB     15           ; EOT (LAST SECTOR ON TRACK)
00006DB5 1B                     <1>       DB     01BH         ; GAP LENGTH
00006DB6 FF                     <1>       DB     0FFH         ; DTL
00006DB7 54                     <1>       DB     054H         ; GAP LENGTH FOR FORMAT
00006DB8 F6                     <1>       DB     0F6H         ; FILL BYTE FOR FORMAT
00006DB9 0F                     <1>       DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
00006DBA 08                     <1>       DB     8            ; MOTOR START TIME (1/8 SECONDS)
00006DBB 4F                     <1>       DB     79           ; MAX. TRACK NUMBER
00006DBC 00                     <1>       DB     RATE_500     ; DATA TRANSFER RATE
                                <1> ;--------------------------------------------------------
                                <1> ;     720 KB MEDIA IN 720 KB DRIVE                  :
                                <1> ;--------------------------------------------------------
                                <1> MD_TBL4:
00006DBD DF                     <1>       DB     11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00006DBE 02                     <1>       DB     2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00006DBF 25                     <1>       DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00006DC0 02                     <1>       DB     2            ; 512 BYTES/SECTOR
00006DC1 09                     <1>       DB     09           ; EOT (LAST SECTOR ON TRACK)
00006DC2 2A                     <1>       DB     02AH         ; GAP LENGTH
00006DC3 FF                     <1>       DB     0FFH         ; DTL
00006DC4 50                     <1>       DB     050H         ; GAP LENGTH FOR FORMAT
00006DC5 F6                     <1>       DB     0F6H         ; FILL BYTE FOR FORMAT
00006DC6 0F                     <1>       DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
00006DC7 08                     <1>       DB     8            ; MOTOR START TIME (1/8 SECONDS)
00006DC8 4F                     <1>       DB     79           ; MAX. TRACK NUMBER
00006DC9 80                     <1>       DB     RATE_250     ; DATA TRANSFER RATE
                                <1> ;--------------------------------------------------------
                                <1> ;     720 KB MEDIA IN 1.44 MB DRIVE                 :
                                <1> ;--------------------------------------------------------
                                <1> MD_TBL5:
00006DCA DF                     <1>       DB     11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00006DCB 02                     <1>       DB     2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00006DCC 25                     <1>       DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00006DCD 02                     <1>       DB     2            ; 512 BYTES/SECTOR
00006DCE 09                     <1>       DB     09           ; EOT (LAST SECTOR ON TRACK)
00006DCF 2A                     <1>       DB     02AH         ; GAP LENGTH
00006DD0 FF                     <1>       DB     0FFH         ; DTL
00006DD1 50                     <1>       DB     050H         ; GAP LENGTH FOR FORMAT
00006DD2 F6                     <1>       DB     0F6H         ; FILL BYTE FOR FORMAT
00006DD3 0F                     <1>       DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
00006DD4 08                     <1>       DB     8            ; MOTOR START TIME (1/8 SECONDS)
00006DD5 4F                     <1>       DB     79           ; MAX. TRACK NUMBER
00006DD6 80                     <1>       DB     RATE_250     ; DATA TRANSFER RATE
                                <1> ;--------------------------------------------------------
                                <1> ;     1.44 MB MEDIA IN 1.44 MB DRIVE                :
                                <1> ;--------------------------------------------------------
                                <1> MD_TBL6:
00006DD7 AF                     <1>       DB     10101111B    ; SRT=A, HD UNLOAD=0F - 1ST SPECIFY BYTE
00006DD8 02                     <1>       DB     2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00006DD9 25                     <1>       DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00006DDA 02                     <1>       DB     2            ; 512 BYTES/SECTOR
00006DDB 12                     <1>       DB     18           ; EOT (LAST SECTOR ON TRACK)
00006DDC 1B                     <1>       DB     01BH         ; GAP LENGTH
00006DDD FF                     <1>       DB     0FFH         ; DTL
00006DDE 6C                     <1>       DB     06CH         ; GAP LENGTH FOR FORMAT
00006DDF F6                     <1>       DB     0F6H         ; FILL BYTE FOR FORMAT
00006DE0 0F                     <1>       DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
00006DE1 08                     <1>       DB     8            ; MOTOR START TIME (1/8 SECONDS)
00006DE2 4F                     <1>       DB     79           ; MAX. TRACK NUMBER
00006DE3 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:
00006DE4 E0             <1>       db     NO_ERR
00006DE5 024001BB       <1>       db     BAD_ADDR_MARK,BAD_SEEK,BAD_CMD,UNDEF_ERR
00006DE9 04BB100A       <1>       db     RECORD_NOT_FND,UNDEF_ERR,BAD_ECC,BAD_SECTOR
                        <1>
                        <1> ; 17/12/2014 (mov ax, [cfd])
                        <1> ; 11/12/2014
00006DED 00             <1> cfd:         db 0               ; current floppy drive (for GET_PARM)
                        <1> ; 17/12/2014                    ; instead of 'DISK_POINTER'
00006DEE 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)
00006DEF 90             <1> align 2
                        <1>
00006DF0 F001           <1> HF_PORT:     dw     1F0h  ; Default = 1F0h
                        <1>                            ; (170h)
00006DF2 F603           <1> HF_REG_PORT: dw     3F6h  ; HF_PORT + 206h
                        <1>
                        <1> ; 05/01/2015
00006DF4 00             <1> hf_m_s:         db      0     ; (0 = Master, 1 = Slave)
                        <1>
                        <1> ; ****************************************************************************
                        
                            Align 2
                        
                            ; 04/11/2014 (Retro UNIX 386 v1)
00006DF6 0000               mem_1m_1k:   dw 0  ; Number of contiguous KB between
                                               ; 1 and 16 MB, max. 3C00h = 15 MB.
00006DF8 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)
00006DFA 00                 boot_drv:    db 0 ; boot drive number (physical)
                            ; 24/11/2014
00006DFB 00                 drv:       db 0
00006DFC 00                 last_drv:    db 0 ; last hdd
00006DFD 00                 hdc:         db 0  ; number of hard disk drives
                                             ; (present/detected)
                        
                            ; 24/11/2014 (Retro UNIX 386 v1)
                            ; Physical drive type & flags
00006DFE 00                 fd0_type:    db 0  ; floppy drive type
00006DFF 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)
00006E00 00                 hd0_type:    db 0  ; EDD status for hd0 (bit 7 = present flag)
00006E01 00                 hd1_type:    db 0  ; EDD status for hd1 (bit 7 = present flag)
00006E02 00                 hd2_type:    db 0  ; EDD status for hd2 (bit 7 = present flag)
00006E03 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
00006E04 00000000000000000-     drv.cylinders: dw 0,0,0,0,0,0,0
00006E0D 0000000000
00006E12 00000000000000000-     drv.heads:     dw 0,0,0,0,0,0,0
00006E1B 0000000000
00006E20 00000000000000000-     drv.spt:       dw 0,0,0,0,0,0,0
00006E29 0000000000
00006E2E 00000000000000000-     drv.size:      dd 0,0,0,0,0,0,0
00006E37 00000000000000000-
00006E40 00000000000000000-
00006E49 00
00006E4A 00000000000000         drv.status:    db 0,0,0,0,0,0,0
00006E51 00000000000000         drv.error:     db 0,0,0,0,0,0,0
                        
                                Align 2
                        
                                ;;; 11/03/2015
                                %include 'kybdata.s'      ; KEYBOARD (BIOS) DATA
                              1 <1> ; ****************************************************************************
                              2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - kybdata.s
                              3 <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
00006E58 524F50514B             <1> K30:  db    82,79,80,81,75
00006E5D 4C4D474849             <1>       db    76,77,71,72,73          ; 10 NUMBER ON KEYPAD
                                <1> ;-----      SUPER-SHIFT-TABLE
00006E62 101112131415           <1>       db    16,17,18,19,20,21   ; A-Z TYPEWRITER CHARS
00006E68 161718191E1F           <1>       db    22,23,24,25,30,31
00006E6E 202122232425           <1>       db    32,33,34,35,36,37
00006E74 262C2D2E2F30           <1>       db    38,44,45,46,47,48
00006E7A 3132                   <1>       db    49,50
                                <1>
                                <1> ;-----      TABLE OF SHIFT KEYS AND MASK VALUES
                                <1> ;-----      KEY_TABLE
00006E7C 52                     <1> _K6:    db      INS_KEY                 ; INSERT KEY
00006E7D 3A4546381D             <1>       db    CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
00006E82 2A36                   <1>         db      LEFT_KEY,RIGHT_KEY
                                <1> _K6L    equ     $-_K6
                                <1>
                                <1> ;-----      MASK_TABLE
00006E84 80                     <1> _K7:    db      INS_SHIFT               ; INSERT MODE SHIFT
00006E85 4020100804             <1>       db    CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
00006E8A 0201                   <1>       db    LEFT_SHIFT,RIGHT_SHIFT
                                <1>
                                <1> ;-----      TABLES FOR CTRL CASE            ;---- CHARACTERS ------
00006E8C 1BFF00FFFFFF           <1> _K8:  db    27,-1,0,-1,-1,-1    ; Esc, 1, 2, 3, 4, 5
00006E92 1EFFFFFFFF1F           <1>       db    30,-1,-1,-1,-1,31   ; 6, 7, 8, 9, 0, -
00006E98 FF7FFF111705           <1>       db    -1,127,-1,17,23,5   ; =, Bksp, Tab, Q, W, E
00006E9E 12141915090F           <1>       db    18,20,25,21,9,15    ; R, T, Y, U, I, O
00006EA4 101B1D0AFF01           <1>       db    16,27,29,10,-1,1    ; P, [, ], Enter, Ctrl, A
00006EAA 13040607080A           <1>       db    19,4,6,7,8,10       ; S, D, F, G, H, J
00006EB0 0B0CFFFFFFFF           <1>       db    11,12,-1,-1,-1,-1   ; K, L, :, ', `, LShift
00006EB6 1C1A18031602           <1>       db    28,26,24,3,22,2           ; Bkslash, Z, X, C, V, B
00006EBC 0E0DFFFFFFFF           <1>       db    14,13,-1,-1,-1,-1   ; N, M, ,, ., /, RShift
00006EC2 96FF20FF               <1>       db    150,-1,' ',-1       ; *, ALT, Spc, CL
                                <1>       ;                         ;----- FUNCTIONS ------
00006EC6 5E5F60616263           <1>       db    94,95,96,97,98,99   ; F1 - F6
00006ECC 64656667FFFF           <1>       db    100,101,102,103,-1,-1     ; F7 - F10, NL, SL
00006ED2 778D848E738F           <1>       db    119,141,132,142,115,143   ; Home, Up, PgUp, -, Left, Pad5
00006ED8 749075917692           <1>       db    116,144,117,145,118,146 ; Right, +, End, Down, PgDn, Ins
00006EDE 93FFFFFF898A           <1>       db    147,-1,-1,-1,137,138      ; Del, SysReq, Undef, WT, F11, F12
                                <1>
                                <1> ;-----      TABLES FOR LOWER CASE ----------
00006EE4 1B3132333435363738-    <1> K10:  db    27,'1234567890-=',8,9
00006EED 39302D3D0809           <1>
00006EF3 71776572747975696F-    <1>       db    'qwertyuiop[]',13,-1,'asdfghjkl;',39
00006EFC 705B5D0DFF61736466-    <1>
00006F05 67686A6B6C3B27         <1>
00006F0C 60FF5C7A786376626E-    <1>       db    96,-1,92,'zxcvbnm,./',-1,'*',-1,' ',-1
00006F15 6D2C2E2FFF2AFF20FF     <1>
                                <1> ;-----      LC TABLE SCAN
00006F1E 3B3C3D3E3F             <1>       db    59,60,61,62,63            ; BASE STATE OF F1 - F10
00006F23 4041424344             <1>       db    64,65,66,67,68
00006F28 FFFF                   <1>       db    -1,-1               ; NL, SL
                                <1>
                                <1> ;-----      KEYPAD TABLE
00006F2A 474849FF4BFF           <1> K15:  db    71,72,73,-1,75,-1   ; BASE STATE OF KEYPAD KEYS
00006F30 4DFF4F50515253         <1>       db    77,-1,79,80,81,82,83
00006F37 FFFF5C8586             <1>       db    -1,-1,92,133,134    ; SysRq, Undef, WT, F11, F12
                                <1>
                                <1> ;-----      TABLES FOR UPPER CASE ----------
00006F3C 1B21402324255E262A-    <1> K11:  db    27,'!@#$%',94,'&*()_+',8,0
00006F45 28295F2B0800           <1>
00006F4B 51574552545955494F-    <1>       db    'QWERTYUIOP{}',13,-1,'ASDFGHJKL:"'
00006F54 507B7D0DFF41534446-    <1>
00006F5D 47484A4B4C3A22         <1>
00006F64 7EFF7C5A584356424E-    <1>       db    126,-1,'|ZXCVBNM<>?',-1,'*',-1,' ',-1
00006F6D 4D3C3E3FFF2AFF20FF     <1>
                                <1> ;-----      UC TABLE SCAN
00006F76 5455565758             <1> K12:  db    84,85,86,87,88            ; SHIFTED STATE OF F1 - F10
00006F7B 595A5B5C5D             <1>       db    89,90,91,92,93
00006F80 FFFF                   <1>       db    -1,-1               ; NL, SL
                                <1>
                                <1> ;-----      NUM STATE TABLE
00006F82 3738392D3435362B31-    <1> K14:  db    '789-456+1230.'           ; NUMLOCK STATE OF KEYPAD KEYS
00006F8B 3233302E               <1>
                                <1>       ;
```

```
 100 00006F8F FFFF7C8788          <1>       db    -1,-1,124,135,136   ; SysRq, Undef, WT, F11, F12
 101                              <1>
 102                              <1> ; 26/08/2014
 103                              <1> ; Retro UNIX 8086 v1 - UNIX.ASM (03/03/2014)
 104                              <1> ; Derived from IBM "pc-at"
 105                              <1> ; rombios source code (06/10/1985)
 106                              <1> ; 'dseg.inc'
 107                              <1>
 108                              <1> ;----------------------------------------;
 109                              <1> ;     SYSTEM DATA AREA          ;
 110                              <1> ;----------------------------------------
 111 00006F94 00                  <1> BIOS_BREAK   db     0            ; BIT 7=1 IF BREAK KEY HAS BEEN PRESSED
 112                              <1>
 113                              <1> ;----------------------------------------
 114                              <1> ;     KEYBOARD DATA AREAS       ;
 115                              <1> ;----------------------------------------
 116                              <1>
 117 00006F95 00                  <1> KB_FLAG            db     0            ; KEYBOARD SHIFT STATE AND STATUS FLAGS
 118 00006F96 00                  <1> KB_FLAG_1    db     0            ; SECOND BYTE OF KEYBOARD STATUS
 119 00006F97 00                  <1> KB_FLAG_2    db     0            ; KEYBOARD LED FLAGS
 120 00006F98 00                  <1> KB_FLAG_3    db     0            ; KEYBOARD MODE STATE AND TYPE FLAGS
 121 00006F99 00                  <1> ALT_INPUT    db     0            ; STORAGE FOR ALTERNATE KEY PAD ENTRY
 122 00006F9A [AA6F0000]          <1> BUFFER_START dd     KB_BUFFER    ; OFFSET OF KEYBOARD BUFFER START
 123 00006F9E [CA6F0000]          <1> BUFFER_END   dd     KB_BUFFER + 32      ; OFFSET OF END OF BUFFER
 124 00006FA2 [AA6F0000]          <1> BUFFER_HEAD  dd     KB_BUFFER    ; POINTER TO HEAD OF KEYBOARD BUFFER
 125 00006FA6 [AA6F0000]          <1> BUFFER_TAIL  dd     KB_BUFFER    ; POINTER TO TAIL OF KEYBOARD BUFFER
 126                              <1> ; ------     HEAD = TAIL  INDICATES THAT THE BUFFER IS EMPTY
 127 00006FAA 0000<rept>          <1> KB_BUFFER    times  16 dw 0             ; ROOM FOR 16 SCAN CODE ENTRIES
 128                              <1>
 129                              <1> ; /// End Of KEYBOARD DATA ///
3071                                  %include 'vidata.s'       ; VIDEO (BIOS) DATA
   1                              <1> ; ****************************************************************************
   2                              <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.3 - vidata.s
   3                              <1> ; ----------------------------------------------------------------------------
   4                              <1> ; Last Update: 24/11/2020
   5                              <1> ; ----------------------------------------------------------------------------
   6                              <1> ; Beginning: 16/01/2016
   7                              <1> ; ----------------------------------------------------------------------------
   8                              <1> ; Assembler: NASM version 2.15 (trdos386.s)
   9                              <1> ; ----------------------------------------------------------------------------
  10                              <1> ; Turkish Rational DOS
  11                              <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
  12                              <1> ;
  13                              <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
  14                              <1> ; vidata.inc (11/03/2015)
  15                              <1> ;
  16                              <1> ; Derived from 'IBM PC-AT' BIOS source code (1985)
  17                              <1> ; ****************************************************************************
  18                              <1>
  19                              <1> ; Retro UNIX 386 v1 Kernel - VIDATA.S
  20                              <1> ; Last Modification: 11/03/2015
  21                              <1> ;                (Data section for 'VIDEO.INC')
  22                              <1> ;
  23                              <1> ; ///////// VIDEO DATA ///////////////
  24                              <1>
  25                              <1> ;----------------------------------------
  26                              <1> ;     VIDEO DISPLAY DATA AREA          ;
  27                              <1> ;----------------------------------------
  28 00006FCA 03                  <1> CRT_MODE:    db     3      ; CURRENT DISPLAY MODE (TYPE)
  29 00006FCB 29                  <1> CRT_MODE_SET:      db     29h    ; CURRENT SETTING OF THE 3X8 REGISTER
  30                              <1>                            ; (29h default setting for video mode 3)
  31                              <1>                            ; Mode Select register Bits
  32                              <1>                            ;   BIT 0 - 80x25 (1), 40x25 (0)
  33                              <1>                            ;   BIT 1 - ALPHA (0), 320x200 GRAPHICS (1)
  34                              <1>                            ;   BIT 2 - COLOR (0), BW (1)
  35                              <1>                            ;   BIT 3 - Video Sig. ENABLE (1), DISABLE (0)
  36                              <1>                            ;   BIT 4 - 640x200 B&W Graphics Mode (1)
  37                              <1>                            ;   BIT 5 - ALPHA mode BLINKING (1)
  38                              <1>                            ;   BIT 6, 7 - Not Used
  39                              <1>
  40                              <1> ; Mode 0 - 2Ch = 101100b  ; 40x25 text, 16 gray colors
  41                              <1> ; Mode 1 - 28h = 101000b  ; 40x25 text, 16 fore colors, 8 back colors
  42                              <1> ; Mode 2 - 2Dh = 101101b  ; 80x25 text, 16 gray colors
  43                              <1> ; Mode 3 - 29h = 101001b  ; 80x25 text, 16 fore color, 8 back color
  44                              <1> ; Mode 4 - 2Ah = 101010b  ; 320x200 graphics, 4 colors
  45                              <1> ; Mode 5 - 2Eh = 101110b  ; 320x200 graphics, 4 gray colors
  46                              <1> ; Mode 6 - 1Eh = 011110b  ; 640x200 graphics, 2 colors
  47                              <1> ; Mode 7 - 29h = 101001b  ; 80x25 text, black & white colors
  48                              <1> ; Mode & 37h = Video signal OFF
  49                              <1>
  50                              <1> ; 24/06/2016
  51 00006FCC 50                  <1> CRT_COLS:    db     80     ; Number of columns
  52                              <1>
  53                              <1> ; 01/07/2016
  54 00006FCD 00                  <1> CRT_PALETTE: db     0      ; Current palette setting
  55                              <1>
  56                              <1> ; 03/07/2016
  57 00006FCE 10                  <1> CHAR_HEIGHT: db     16     ; Default character height
  58 00006FCF 60                  <1> VGA_VIDEO_CTL:     db     60h    ; ROM BIOS DATA AREA Offset 87h
  59 00006FD0 F9                  <1> VGA_SWITCHES:      db     0F9h   ; Feature Bit Switches (the basic screen)
  60 00006FD1 51                  <1> VGA_MODESET_CTL: db       051h   ; Basic mode set options (VGA video flags)
  61                              <1>                            ; ROM BIOS DATA AREA Offset 89h
  62                              <1>                            ; Bit 7, 4 : Mode
  63                              <1>                            ;       01 : 400-line mode
  64                              <1>                            ; Bit 6      : Display switch enabled = 1
  65                              <1>                            ; Bit 5      : Reserved = 0
  66                              <1>                            ; Bit 3      : Default palette loading
  67                              <1>                            ;       disabled = 0
  68                              <1>                            ; Bit 2 : Color monitor = 0
  69                              <1>                            ; Bit 1 = Gray scale summing
  70                              <1>                            ;       disabled = 0
  71                              <1>                            ; Bit 0 = VGA active = 1
  72 00006FD2 19                  <1> VGA_ROWS:    db     25
  73                              <1>
  74                              <1> ; 16/01/2016
```

```
                                <1> chr_attrib:  ; Character color/attributes for video pages (0 to 7)
00006FD3 0707070707070707       <1>       db     07h, 07h, 07h, 07h, 07h, 07h, 07h, 07h
                                <1> ; 30/01/2016
                                <1> vmode:
00006FDB 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
00006FE3 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>
00006FE5 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)
00006FE6 0302010007040506       <1>       db     03h, 02h, 01h, 00h, 07h, 04h, 05h, 06h
                                <1> vga_g_modes: ; 31/07/2016
00006FEE 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
00006FF6 [56700000]             <1>       dd     vga_mode_03h
00006FFA [56700000]             <1>       dd     vga_mode_03h ; mode 02h -> mode 03h
00006FFE [96700000]             <1>       dd     vga_mode_01h
00007002 [96700000]             <1>       dd     vga_mode_01h ; mode 00h -> mode 01h
                                <1>       ;dd    vga_mode_07h
00007006 [56700000]             <1>       dd     vga_mode_03h ; mode 07h -> mode 03h
0000700A [D6700000]             <1>       dd     vga_mode_04h
0000700E [D6700000]             <1>       dd     vga_mode_04h ; mode 05h -> mode 04h
00007012 [16710000]             <1>       dd     vga_mode_06h
00007016 [56710000]             <1>       dd     vga_mode_13h
0000701A [96710000]             <1>       dd     vga_mode_F0h
0000701E [D6710000]             <1>       dd     vga_mode_12h
00007022 [16720000]             <1>       dd     vga_mode_6Ah
00007026 [56720000]             <1>       dd     vga_mode_0Dh
0000702A [96720000]             <1>       dd     vga_mode_0Eh
0000702E [D6720000]             <1>       dd     vga_mode_10h
00007032 [16730000]             <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
00007036 0000000000020202       <1>       db     CTEXT, CTEXT, CTEXT, CTEXT, MTEXT, CGA, CGA, CGA
                                <1> vga_g_memmodel: ; 31/07/2016
0000703E 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:
00007046 020202020001010103-    <1>       db     2, 2, 2, 2, 0, 1, 1, 1, 3, 3, 2, 2, 1, 1, 2, 2
0000704F 03020201010202         <1>
                                <1>                                       ; (vgatables.h, VGAMODES, dac)
                                <1>                                       ; 17/11/2020
                                <1> vga_params:
                                <1>       ; 23/11/2020
                                <1>       ; 16/11/2020
                                <1>       ; 09/11/2020, 10/11/2020, 11/11/2020 (TRDOS 386 v2.0.3)
                                <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>       ; 09/11/2020
                                <1>       ; Block Structure of Video Parameter Table
                                <1>       ;
                                <1>       ; Offset     # Bytes            Contents
                                <1>       ;
                                <1>       ;  0         1            # columns
                                <1>       ;  1         1            # rows - 1
                                <1>       ;  2         1            Pixels/character
                                <1>       ;  3-4       2            Page length
                                <1>       ;  5-8       4            Sequencer Registers
                                <1>       ;  9         1            Miscellaneous Register
                                <1>       ;  10-34     25           CRTC Registers
                                <1>       ;  35-54     20           Attribute Registers
                                <1>       ;  55-63     9            Graphics Controller Registers
                                <1>       ;
                                <1>       ; Ref: Programmer's Guide to EGA, VGA, and Super VGA cards
                                <1>       ; (Richard F. Ferraro, 1994)
                                <1>
                                <1>       ;
                                <1> vga_mode_03h:  ; mode 03h, 80*25 text, CGA colors
                                <1>       ; 11/11/2020
00007056 5018100010             <1>       db     80, 24, 16, 00h, 10h ; tw, th-1, ch, slength (5)
0000705B 00030002               <1>       db     00h, 03h, 00h, 02h ; sequ regs (4)
```

```
0000705F 67                  <1>       db     67h    ; misc reg (1)
00007060 5F4F50825581BF1F    <1>       db     5Fh, 4Fh, 50h, 82h, 55h, 81h, 0BFh, 1Fh
                             <1>       ; 09/11/2020
                             <1>       ;db    5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
00007068 004F                <1>       db     00h, 4Fh
                             <1> vga_p_cm_pos equ $ - vga_mode_03h
0000706A 0D0E00000000        <1>       db     0Dh, 0Eh, 00h, 00h, 00h, 00h
00007070 9C8E8F281F96B9A3    <1>       db     9Ch, 8Eh, 8Fh, 28h, 1Fh, 96h, 0B9h, 0A3h
00007078 FF                  <1>       db     0FFh   ; crtc_regs (25)
00007079 0001020304051407    <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
00007081 38393A3B3C3D3E3F    <1>       db     38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
                             <1>       ;db    0Ch, 00h, 0Fh, 08h  ; actl regs (20)
00007089 0C000F00            <1>       db     0Ch, 00h, 0Fh, 00h  ; 19/11/2020
0000708D 0000000000100E0FFF  <1>       db     00h, 00h, 00h, 00h, 00h, 10h, 0Eh, 0Fh, 0FFh ; grdc regs (9)
                             <1>       ; 09/11/2020
                             <1>       ;db    00h, 00h, 00h, 00h, 00h, 10h, 0Eh, 00h, 0FFh ; grdc regs (9)
                             <1> vga_mode_01h:      ; mode 01h, 40*25 text, CGA colors
00007096 2818100008          <1>       db     40, 24, 16, 00h, 08h ; tw, th-1, ch, slength
0000709B 08030002            <1>       db     08h, 03h, 00h, 02h  ; sequ regs
0000709F 67                  <1>       db     67h    ; misc reg
000070A0 2D2728902BA0BF1F    <1>       db     2Dh, 27h, 28h, 90h, 2Bh, 0A0h, 0BFh, 1Fh
000070A8 004F0D0E00000000    <1>       db     00h, 4Fh, 0Dh, 0Eh, 00h, 00h, 00h, 00h
000070B0 9C8E8F141F96B9A3    <1>       db     9Ch, 8Eh, 8Fh, 14h, 1Fh, 96h, 0B9h, 0A3h
000070B8 FF                  <1>       db     0FFh   ; crtc_regs
000070B9 0001020304051407    <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
000070C1 38393A3B3C3D3E3F    <1>       db     38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
                             <1>       ;db    0Ch, 00h, 0Fh, 08h  ; actl regs (20)
000070C9 0C000F00            <1>       db     0Ch, 00h, 0Fh, 00h  ; 19/11/2020
000070CD 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
                             <1>       ; 11/11/2020
000070D6 2818080040          <1>       db     40, 24, 8, 00h, 40h ; tw, th-1, ch, slength
000070DB 09030002            <1>       db     09h, 03h, 00h, 02h ; sequ regs
000070DF 63                  <1>       db     63h    ; misc reg
000070E0 2D2728902B80BF1F    <1>       db     2Dh, 27h, 28h, 90h, 2Bh, 80h, 0BFh, 1Fh
000070E8 00C1000000000000    <1>       db     00h, 0C1h, 00h, 00h, 00h, 00h, 00h, 00h
000070F0 9C8E8F140096B9A2    <1>       db     9Ch, 8Eh, 8Fh, 14h, 00h, 96h, 0B9h, 0A2h
000070F8 FF                  <1>       db     0FFh   ; crtc_regs
000070F9 0013151702040607    <1>       db     00h, 13h, 15h, 17h, 02h, 04h, 06h, 07h
00007101 1011121314151617    <1>       db     10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h
00007109 01000300            <1>       db     01h, 00h, 03h, 00h ; actl regs
0000710D 0000000000300F0FFF  <1>       db     00h, 00h, 00h, 00h, 00h, 30h, 0Fh, 0Fh, 0FFh ; grdc regs
                             <1> vga_mode_06h:      ; 640*200 graphics, 2 colors, CGA
                             <1>       ; 11/11/2020
00007116 5018080040          <1>       db     80, 24, 8, 00h, 40h   ;   tw, th-1, ch, slength
0000711B 01010006            <1>       db     01h, 01h, 00h, 06h ; sequ regs
0000711F 63                  <1>       db     63h    ; misc reg
00007120 5F4F50825480BF1F    <1>       db     5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
00007128 00C1000000000000    <1>       db     00h, 0C1h, 00h, 00h, 00h, 00h, 00h, 00h
00007130 9C8E8F280096B9C2    <1>       db     9Ch, 8Eh, 8Fh, 28h, 00h, 96h, 0B9h, 0C2h
00007138 FF                  <1>       db     0FFh   ; crtc regs
00007139 0017171717171717    <1>       db     00h, 17h, 17h, 17h, 17h, 17h, 17h, 17h
00007141 1717171717171717    <1>       db     17h, 17h, 17h, 17h, 17h, 17h, 17h, 17h
00007149 01000100            <1>       db     01h, 00h, 01, 00h  ; actl regs
0000714D 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
                             <1>       ; 11/11/2020
                             <1>       ;db    40, 24, 8, 00h, 20h ; tw, th-1, ch, slength (5)
                             <1>       ; 23/11/2020 - 10/11/2020
00007156 28180800FA          <1>       db     40, 24, 8, 00h, 0FAh ; tw, th-1, ch, slength (5)
0000715B 010F000E            <1>       db     01h, 0Fh, 00h, 0Eh ; sequ regs (4)
0000715F 63                  <1>       db     63h    ; misc reg (1)
00007160 5F4F50825480BF1F    <1>       db     5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
00007168 0041000000000000    <1>       db     00h, 041h, 00h, 00h, 00h, 00h, 00h, 00h
00007170 9C8E8F284096B9A3    <1>       db     9Ch, 8Eh, 8Fh, 28h, 40h, 96h, 0B9h, 0A3h
00007178 FF                  <1>       db     0FFh   ; crtc regs (25)
00007179 0001020304050607    <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
00007181 08090A0B0C0D0E0F    <1>       db     08h, 09h, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh, 0Fh
00007189 41000F00            <1>       db     41h, 00h, 0Fh, 00h  ; actl regs (20)
                             <1>       ; 10/11/2020
                             <1>       ;db    41h, 01h, 0Fh, 13h  ; actl regs (20)
0000718D 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
00007196 2818080000          <1>       db     40, 24, 8, 00h, 00h ; tw, th-1, ch, slength
0000719B 010F0006            <1>       db     01h, 0Fh, 00h, 06h ; sequ regs
0000719F E3                  <1>       db     0E3h   ; misc reg
000071A0 5F4F508254800D3E    <1>       db     5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Dh, 3Eh
000071A8 0041000000000000    <1>       db     00h, 41h, 00h, 00h, 00h, 00h, 00h, 00h
000071B0 EAACDF2800E706E3    <1>       db     0EAh, 0ACh, 0DFh, 28h, 00h, 0E7h, 06h, 0E3h
000071B8 FF                  <1>       db     0FFh   ; crtc regs (25)
000071B9 0001020304050607    <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
000071C1 08090A0B0C0D0E0F    <1>       db     08h, 09h, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh, 0Fh
000071C9 41000F00            <1>       db     41h, 00h, 0Fh, 00h  ; actl regs
000071CD 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
                             <1>       ; 11/11/2020
                             <1>       ;db    80, 29, 16, 0, 0 ; tw, th-1, ch, slength
                             <1>       ; 09/11/2020
000071D6 501D1000A0          <1>       db     80, 29, 16, 00h, 0A0h ; tw, th-1, ch, slength
000071DB 010F0006            <1>       db     01h, 0Fh, 00h, 06h ; sequ regs
000071DF E3                  <1>       db     0E3h   ; misc reg
000071E0 5F4F508254800B3E    <1>       db     5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Bh, 3Eh
```

```
                                <1>       ; 09/11/2020
                                <1>       ;db    5Fh, 4Fh, 50h, 82h, 53h, 9Fh, 0Bh, 3Eh
000071E8 0040000000000000       <1>       db     00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h
000071F0 EA8CDF2800E704E3       <1>       db     0EAh, 8Ch, 0DFh, 28h, 00h, 0E7h, 04h, 0E3h
                                <1>       ; 09/11/2020
                                <1>       ;db    0E9h, 8Bh, 0DFh, 28h, 00h, 0E7h, 04h, 0E3h
000071F8 FF                     <1>       db     0FFh   ; crtc regs
000071F9 0001020304051407       <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
00007201 38393A3B3C3D3E3F       <1>       db     38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
00007209 01000F00               <1>       db     01h, 00h, 0Fh, 00h  ; actl regs
0000720D 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
                                <1>       ; 11/11/2020
00007216 6424100000             <1>       db     100, 36, 16, 00h, 00h ; tw, th-1, ch, slength
0000721B 010F0006               <1>       db     01h, 0Fh, 00h, 06h ; sequ regs
0000721F E3                     <1>       db     0E3h   ; misc reg
00007220 7F6363836B1B72F0       <1>       db     7Fh, 63h, 63h, 83h, 6Bh, 1Bh, 72h, 0F0h
00007228 0060000000000000       <1>       db     00h, 60h, 00h, 00h, 00h, 00h, 00h, 00h
00007230 598D5732005773E3       <1>       db     59h, 8Dh, 57h, 32h, 00h, 57h, 73h, 0E3h
00007238 FF                     <1>       db     0FFh   ; crtc regs
00007239 0001020304051407       <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
00007241 38393A3B3C3D3E3F       <1>       db     38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
00007249 01000F00               <1>       db     01h, 00h, 0Fh, 00h  ; actl regs
0000724D 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
00007256 2818080020             <1>       db     40, 24, 8, 00h, 20h ; tw, th-1, ch, slength
0000725B 090F0006               <1>       db     09h, 0Fh, 00h, 06h ; sequ regs
0000725F 63                     <1>       db     63h    ; misc reg
00007260 2D2728902B80BF1F       <1>       db     2Dh, 27h, 28h, 90h, 2Bh, 80h, 0BFh, 1Fh
00007268 00C0000000000000       <1>       db     00h, 0C0h, 00h, 00h, 00h, 00h, 00h, 00h
00007270 9C8E8F140096B9E3       <1>       db     9Ch, 8Eh, 8Fh, 14h, 00h, 96h, 0B9h, 0E3h
00007278 FF                     <1>       db     0FFh   ; crtc regs
00007279 0001020304050607       <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
00007281 1011121314151617       <1>       db     10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h
00007289 01000F00               <1>       db     01h, 00h, 0Fh, 00h  ; actl regs
0000728D 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
00007296 5018080040             <1>       db     80, 24, 8, 00h, 40h ; tw, th-1, ch, slength
0000729B 010F0006               <1>       db     01h, 0Fh, 00h, 06h ; sequ regs
0000729F 63                     <1>       db     63h    ; misc reg
000072A0 5F4F50825480BF1F       <1>       db     5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
000072A8 00C0000000000000       <1>       db     00h, 0C0h, 00h, 00h, 00h, 00h, 00h, 00h
000072B0 9C8E8F280096B9E3       <1>       db     9Ch, 8Eh, 8Fh, 28h, 00h, 96h, 0B9h, 0E3h
000072B8 FF                     <1>       db     0FFh   ; crtc regs
000072B9 0001020304050607       <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
000072C1 1011121314151617       <1>       db     10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h
000072C9 01000F00               <1>       db     01h, 00h, 0Fh, 00h  ; actl regs
000072CD 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
000072D6 50180E0080             <1>       db     80, 24, 14, 00h, 80h ; tw, th-1, ch, slength
000072DB 010F0006               <1>       db     01h, 0Fh, 00h, 06h ; sequ regs
000072DF A3                     <1>       db     0A3h   ; misc reg
000072E0 5F4F50825480BF1F       <1>       db     5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
000072E8 0040000000000000       <1>       db     00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h
000072F0 83855D280F63BAE3       <1>       db     83h, 85h, 5Dh, 28h, 0Fh, 63h, 0BAh, 0E3h
000072F8 FF                     <1>       db     0FFh   ; crtc regs
000072F9 0001020304051407       <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
00007301 38393A3B3C3D3E3F       <1>       db     38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
00007309 01000F00               <1>       db     01h, 00h, 0Fh, 00h  ; actl regs
0000730D 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
                                <1>       ; 11/11/2020
00007316 501D1000A0             <1>       db     80, 29, 16, 00h, 0A0h ; tw, th-1, ch, slength
0000731B 010F0006               <1>       db     01h, 0Fh, 00h, 06h ; sequ regs
0000731F E3                     <1>       db     0E3h   ; misc reg
00007320 5F4F508254800B3E       <1>       db     5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Bh, 3Eh
00007328 0040000000000000       <1>       db     00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h
00007330 EA8CDF2800E704C3       <1>       db     0EAh, 8Ch, 0DFh, 28h, 00h, 0E7h, 04h, 0C3h ; 11/11/2020
00007338 FF                     <1>       db     0FFh   ; crtc regs
00007339 003F003F003F003F       <1>       db     00h, 3Fh, 00h, 3Fh, 00h, 3Fh, 00h, 3Fh
00007341 003F003F003F003F       <1>       db     00h, 3Fh, 00h, 3Fh, 00h, 3Fh, 00h, 3Fh
00007349 01000F00               <1>       db     01h, 00h, 0Fh, 00h  ; actl regs
0000734D 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 ///
                                <1>
                                <1> ; 23/11/2020
                                <1> ; VBE 2 BOCHS/QEMU emulator extensions
                                <1> ;     for TRDOS 386 v2 kernel (video bios)
                                <1>
                                <1> ; vbetables.h by Volker Rupper (02/01/2020)
                                <1>
                                <1> b_vbe_modes:
                                <1>       ;/* standard VESA modes */
00007356 0001800290010800       <1>       dw 100h,  640, 400,  8
0000735E 01018002E0010800       <1>       dw 101h,  640, 480,  8
00007366 0301200358020800       <1>       dw 103h,  800, 600,  8
0000736E 0501000400030800       <1>       dw 105h, 1024, 768,  8
00007376 0E014001C8001000       <1>       dw 10Eh,  320, 200, 16
0000737E 0F014001C8001800       <1>       dw 10Fh,  320, 200, 24
00007386 11018002E0011000       <1>       dw 111h,  640, 480, 16
0000738E 12018002E0011800       <1>       dw 112h,  640, 480, 24
00007396 1401200358021000       <1>       dw 114h,  800, 600, 16
0000739E 1501200358021800       <1>       dw 115h,  800, 600, 24
000073A6 1701000400031000       <1>       dw 117h, 1024, 768, 16
000073AE 1801000400031800       <1>       dw 118h, 1024, 768, 24
                                <1>
                                <1> ;/* BOCHS/PLEX86 'own' mode numbers */
000073B6 40014001C8002000       <1>       dw 140h,  320,  200, 32
000073BE 4101800290012000       <1>       dw 141h,  640,  400, 32
000073C6 42018002E0012000       <1>       dw 142h,  640,  480, 32
000073CE 4301200358022000       <1>       dw 143h,  800,  600, 32
000073D6 4401000400032000       <1>       dw 144h, 1024,  768, 32
000073DE 46014001C8000800       <1>       dw 146h,  320,  200,  8
```

```
000073E6 8D010005D0021000    <1>       dw 18Dh, 1280,  720, 16
000073EE 8E010005D0021800    <1>       dw 18Eh, 1280,  720, 24
000073F6 8F010005D0022000    <1>       dw 18Fh, 1280,  720, 32
000073FE 9001800738041000    <1>       dw 190h, 1920, 1080, 16
00007406 9101800738041800    <1>       dw 191h, 1920, 1080, 24
0000740E 9201800738042000    <1>       dw 192h, 1920, 1080, 32
                             <1>
                             <1> end_of_b_vbe_modes:
                             <1>
                             <1> MA1 equ VBE_MODE_ATTRIBUTE_SUPPORTED
                             <1> MA2 equ VBE_MODE_ATTRIBUTE_EXTENDED_INFO_AVAILABLE
                             <1> MA3 equ VBE_MODE_ATTRIBUTE_COLOR_MODE
                             <1> MA4 equ VBE_MODE_ATTRIBUTE_LINEAR_FRAME_BUFFER_MODE
                             <1> MA5 equ     VBE_MODE_ATTRIBUTE_GRAPHICS_MODE
                             <1>
                             <1> MODE_ATTRIBUTES equ MA1|MA2|MA3|MA4|MA5
                             <1>
                             <1> WA1 equ VBE_WINDOW_ATTRIBUTE_RELOCATABLE
                             <1> WA2 equ     VBE_WINDOW_ATTRIBUTE_READABLE
                             <1> WA3 equ VBE_WINDOW_ATTRIBUTE_WRITEABLE
                             <1>
                             <1> WINA_ATTRIBUTES equ WA1|WA2|WA3
                             <1>
                             <1> ; 24/11/2020
                             <1>
                             <1> %if 0
                             <1>
                             <1> MODE_INFO_LIST:
                             <1>
                             <1> ; 24/11/2020
                             <1> ; '%if 0' disables 24 mode info tables here, until %endif
                             <1> ; ('set_mode_info_list' will set only 1 list for selected mode)
                             <1> ; (Purpose: To save about 1 KB kernel size by removing fixed data)
                             <1>
                             <1>                   dw    0100h ; 640x400x8
                             <1> ModeAttributes1:  dw    MODE_ATTRIBUTES
                             <1> WinAAttributes1:  db    WINA_ATTRIBUTES
                             <1> WinBAttributes1:  db    0
                             <1> WinGranularity1:  dw    VBE_DISPI_BANK_SIZE_KB
                             <1> WinSize1:         dw    VBE_DISPI_BANK_SIZE_KB
                             <1> WinASegment1:           dw    VGAMEM_GRAPH
                             <1> WinBSegment1:           dw    0000h
                             <1> WinFuncPtr1:      dd    0
                             <1> BytesPerScanLine1:      dw    640
                             <1> XResolution1:           dw    640
                             <1> YResolution1:           dw    400
                             <1> XCharSize1:       db    8
                             <1> YCharSize1:       db    16
                             <1> NumberOfPlanes1:  db    1
                             <1> BitsPerPixel1:          db    8
                             <1> NumberOfBanks1:         db    4
                             <1> MemoryModel1:           db    VBE_MEMORYMODEL_PACKED_PIXEL
                             <1> BankSize1:        db    0
                             <1> NumberOfImagePages1:    db    64
                             <1> Reserved_page1:         db    0
                             <1> RedMaskSize1:           db    0
                             <1> RedFieldPosition1:      db    0
                             <1> GreenMaskSize1:         db    0
                             <1> GreenFieldPosition1:    db    0
                             <1> BlueMaskSize1:          db    0
                             <1> BlueFieldPosition1:     db    0
                             <1> RsvdMaskSize1:          db    0
                             <1> RsvdFieldPosition1:     db    0
                             <1> DirectColorModeInfo1:   db    0
                             <1> PhysBasePtr1:           dd    VBE_DISPI_LFB_PHYSICAL_ADDRESS
                             <1> OffScreenMemOffset1:    dd    0
                             <1> OffScreenMemSize1:      dw    0
                             <1> LinBytesPerScanLine1:   dw    640
                             <1> BnkNumberOfPages1:      db    0
                             <1> LinNumberOfPages1:      db    0
                             <1> LinRedMaskSize1:  db    0
                             <1> LinRedFieldPosition1:   db    0
                             <1> LinGreenMaskSize1:      db    0
                             <1> LinGreenFieldPosition1: db      0
                             <1> LinBlueMaskSize1: db    0
                             <1> LinBlueFieldPosition1:  db    0
                             <1> LinRsvdMaskSize1: db    0
                             <1> LinRsvdFieldPosition1:  db    0
                             <1> MaxPixelClock1:         dd    0
                             <1>
                             <1>                   dw    0101h ; 640x480x8
                             <1> ModeAttributes2:  dw    MODE_ATTRIBUTES
                             <1> WinAAttributes2:  db    WINA_ATTRIBUTES
                             <1> WinBAttributes2:  db    0
                             <1> WinGranularity2:  dw    VBE_DISPI_BANK_SIZE_KB
                             <1> WinSize2:         dw    VBE_DISPI_BANK_SIZE_KB
                             <1> WinASegment2:           dw    VGAMEM_GRAPH
                             <1> WinBSegment2:           dw    0000h
                             <1> WinFuncPtr2:      dd    0
                             <1> BytesPerScanLine2: dw   640
                             <1> XResolution2:           dw    640
                             <1> YResolution2:           dw    480
                             <1> XCharSize2:       db    8
                             <1> YCharSize2:       db    16
                             <1> NumberOfPlanes2:  db    1
                             <1> BitsPerPixel2:          db    8
                             <1> NumberOfBanks2:         db    5
                             <1> MemoryModel2:           db    VBE_MEMORYMODEL_PACKED_PIXEL
                             <1> BankSize2:        db    0
                             <1> NumberOfImagePages2:    db    53
                             <1> Reserved_page2:         db    0
                             <1> RedMaskSize2:           db    0
                             <1> RedFieldPosition2: db   0
                             <1> GreenMaskSize2:         db    0
                             <1> GreenFieldPosition2:    db    0
```

```
494                             <1> BlueMaskSize2:            db    0
495                             <1> BlueFieldPosition2:       db    0
496                             <1> RsvdMaskSize2:            db    0
497                             <1> RsvdFieldPosition2:       db    0
498                             <1> DirectColorModeInfo2:     db    0
499                             <1> PhysBasePtr2:             dd    VBE_DISPI_LFB_PHYSICAL_ADDRESS
500                             <1> OffScreenMemOffset2:      dd    0
501                             <1> OffScreenMemSize2: dw     0
502                             <1> LinBytesPerScanLine2:     dw    640
503                             <1> BnkNumberOfPages2:        db    0
504                             <1> LinNumberOfPages2:        db    0
505                             <1> LinRedMaskSize2:   db     0
506                             <1> LinRedFieldPosition2:     db    0
507                             <1> LinGreenMaskSize2:        db    0
508                             <1> LinGreenFieldPosition2: db      0
509                             <1> LinBlueMaskSize2:  db     0
510                             <1> LinBlueFieldPosition2:    db    0
511                             <1> LinRsvdMaskSize2:  db     0
512                             <1> LinRsvdFieldPosition2:    db    0
513                             <1> MaxPixelClock2:           dd    0
514                             <1>
515                             <1>                   dw      0103h ; 800x600x8
516                             <1> ModeAttributes3:  dw      MODE_ATTRIBUTES
517                             <1> WinAAttributes3:  db      WINA_ATTRIBUTES
518                             <1> WinBAttributes3:  db      0
519                             <1> WinGranularity3:  dw      VBE_DISPI_BANK_SIZE_KB
520                             <1> WinSize3:         dw      VBE_DISPI_BANK_SIZE_KB
521                             <1> WinASegment3:             dw    VGAMEM_GRAPH
522                             <1> WinBSegment3:             dw    0000h
523                             <1> WinFuncPtr3:      dd      0
524                             <1> BytesPerScanLine3: dw     800
525                             <1> XResolution3:             dw    800
526                             <1> YResolution3:             dw    600
527                             <1> XCharSize3:       db      8
528                             <1> YCharSize3:       db      16
529                             <1> NumberOfPlanes3:  db      1
530                             <1> BitsPerPixel3:            db    8
531                             <1> NumberOfBanks3:           db    8
532                             <1> MemoryModel3:             db    VBE_MEMORYMODEL_PACKED_PIXEL
533                             <1> BankSize3:        db      0
534                             <1> NumberOfImagePages3:      db    33
535                             <1> Reserved_page3:           db    0
536                             <1> RedMaskSize3:             db    0
537                             <1> RedFieldPosition3: db     0
538                             <1> GreenMaskSize3:           db    0
539                             <1> GreenFieldPosition3:      db    0
540                             <1> BlueMaskSize3:            db    0
541                             <1> BlueFieldPosition3:       db    0
542                             <1> RsvdMaskSize3:            db    0
543                             <1> RsvdFieldPosition3:       db    0
544                             <1> DirectColorModeInfo3:     db    0
545                             <1> PhysBasePtr3:             dd    VBE_DISPI_LFB_PHYSICAL_ADDRESS
546                             <1> OffScreenMemOffset3:      dd    0
547                             <1> OffScreenMemSize3: dw     0
548                             <1> LinBytesPerScanLine3:     dw    800
549                             <1> BnkNumberOfPages3:        db    0
550                             <1> LinNumberOfPages3:        db    0
551                             <1> LinRedMaskSize3:   db     0
552                             <1> LinRedFieldPosition3:     db    0
553                             <1> LinGreenMaskSize3:        db    0
554                             <1> LinGreenFieldPosition:    db    0
555                             <1> LinBlueMaskSize:   db     0
556                             <1> LinBlueFieldPosition:     db    0
557                             <1> LinRsvdMaskSize:   db     0
558                             <1> LinRsvdFieldPosition:     db    0
559                             <1> MaxPixelClock:            dd    0
560                             <1>
561                             <1>                   dw      0105h ; 1024x768x8
562                             <1> ModeAttributes4:  dw      MODE_ATTRIBUTES
563                             <1> WinAAttributes4:  db      WINA_ATTRIBUTES
564                             <1> WinBAttributes4:  db      0
565                             <1> WinGranularity4:  dw      VBE_DISPI_BANK_SIZE_KB
566                             <1> WinSize4:         dw      VBE_DISPI_BANK_SIZE_KB
567                             <1> WinASegment4:             dw    VGAMEM_GRAPH
568                             <1> WinBSegment4:             dw    0000h
569                             <1> WinFuncPtr4:      dd      0
570                             <1> BytesPerScanLine4: dw     1024
571                             <1> XResolution4:             dw    1024
572                             <1> YResolution4:             dw    768
573                             <1> XCharSize4:       db      8
574                             <1> YCharSize4:       db      16
575                             <1> NumberOfPlanes4:  db      1
576                             <1> BitsPerPixel4:            db    8
577                             <1> NumberOfBanks4:           db    12
578                             <1> MemoryModel4:             db    VBE_MEMORYMODEL_PACKED_PIXEL
579                             <1> BankSize4:        db      0
580                             <1> NumberOfImagePages4:      db    20
581                             <1> Reserved_page4:           db    0
582                             <1> RedMaskSize4:             db    0
583                             <1> RedFieldPosition4: db     0
584                             <1> GreenMaskSize4:           db    0
585                             <1> GreenFieldPosition4:      db    0
586                             <1> BlueMaskSize4:            db    0
587                             <1> BlueFieldPosition4:       db    0
588                             <1> RsvdMaskSize4:            db    0
589                             <1> RsvdFieldPosition4:       db    0
590                             <1> DirectColorModeInfo4:     db    0
591                             <1> PhysBasePtr4:             dd    VBE_DISPI_LFB_PHYSICAL_ADDRESS
592                             <1> OffScreenMemOffset4:      dd    0
593                             <1> OffScreenMemSize4: dw     0
594                             <1> LinBytesPerScanLine4:     dw    1024
595                             <1> BnkNumberOfPages4:        db    0
596                             <1> LinNumberOfPages4:        db    0
597                             <1> LinRedMaskSize4:   db     0
598                             <1> LinRedFieldPosition4:     db    0
```

```
599                             <1> LinGreenMaskSize4:        db     0
600                             <1> LinGreenFieldPosition4: db       0
601                             <1> LinBlueMaskSize4:  db     0
602                             <1> LinBlueFieldPosition4:    db     0
603                             <1> LinRsvdMaskSize4:  db     0
604                             <1> LinRsvdFieldPosition4:    db     0
605                             <1> MaxPixelClock4:           dd     0
606                             <1>
607                             <1>                     dw     010Eh ; 320x200x16
608                             <1> ModeAttributes5:    dw     MODE_ATTRIBUTES
609                             <1> WinAAttributes5:    db     WINA_ATTRIBUTES
610                             <1> WinBAttributes5:    db     0
611                             <1> WinGranularity5:    dw     VBE_DISPI_BANK_SIZE_KB
612                             <1> WinSize5:           dw     VBE_DISPI_BANK_SIZE_KB
613                             <1> WinASegment5:             dw     VGAMEM_GRAPH
614                             <1> WinBSegment5:             dw     0000h
615                             <1> WinFuncPtr5:        dd     0
616                             <1> BytesPerScanLine5: dw      640
617                             <1> XResolution5:             dw     320
618                             <1> YResolution5:             dw     200
619                             <1> XCharSize5:         db     8
620                             <1> YCharSize5:         db     16
621                             <1> NumberOfPlanes5:    db     1
622                             <1> BitsPerPixel5:            db     16
623                             <1> NumberOfBanks5:           db     2
624                             <1> MemoryModel5:             db     VBE_MEMORYMODEL_DIRECT_COLOR
625                             <1> BankSize5:          db     0
626                             <1> NumberOfImagePages5:      db     130
627                             <1> Reserved_page5:           db     0
628                             <1> RedMaskSize5:             db     5
629                             <1> RedFieldPosition5: db      11
630                             <1> GreenMaskSize5:           db     6
631                             <1> GreenFieldPosition5:      db     5
632                             <1> BlueMaskSize5:            db     5
633                             <1> BlueFieldPosition5:       db     0
634                             <1> RsvdMaskSize5:            db     0
635                             <1> RsvdFieldPosition5:       db     0
636                             <1> DirectColorModeInfo5:     db     0
637                             <1> PhysBasePtr5:             dd     VBE_DISPI_LFB_PHYSICAL_ADDRESS
638                             <1> OffScreenMemOffset5:      dd     0
639                             <1> OffScreenMemSize5: dw      0
640                             <1> LinBytesPerScanLine5:     dw     640
641                             <1> BnkNumberOfPages5:        db     0
642                             <1> LinNumberOfPages5:        db     0
643                             <1> LinRedMaskSize5:   db     5
644                             <1> LinRedFieldPosition5:     db     11
645                             <1> LinGreenMaskSize5: db     6
646                             <1> LinGreenFieldPosition5:   db     5
647                             <1> LinBlueMaskSize5:  db     5
648                             <1> LinBlueFieldPosition5:    db     0
649                             <1> LinRsvdMaskSize5:  db     0
650                             <1> LinRsvdFieldPosition5:    db     0
651                             <1> MaxPixelClock5:           dd     0
652                             <1>
653                             <1>                     dw     010Fh ; 320x200x24
654                             <1> ModeAttributes6:    dw     MODE_ATTRIBUTES
655                             <1> WinAAttributes6:    db     WINA_ATTRIBUTES
656                             <1> WinBAttributes6:    db     0
657                             <1> WinGranularity6:    dw     VBE_DISPI_BANK_SIZE_KB
658                             <1> WinSize6:           dw     VBE_DISPI_BANK_SIZE_KB
659                             <1> WinASegment6:             dw     VGAMEM_GRAPH
660                             <1> WinBSegment6:             dw     0000h
661                             <1> WinFuncPtr6:        dd     0
662                             <1> BytesPerScanLine6: dw      960
663                             <1> XResolution6:             dw     320
664                             <1> YResolution6:             dw     200
665                             <1> XCharSize6:         db     8
666                             <1> YCharSize6:         db     16
667                             <1> NumberOfPlanes6:    db     1
668                             <1> BitsPerPixel6:            db     24
669                             <1> NumberOfBanks6:           db     3
670                             <1> MemoryModel6:             db     VBE_MEMORYMODEL_DIRECT_COLOR
671                             <1> BankSize6:          db     0
672                             <1> NumberOfImagePages6:      db     86
673                             <1> Reserved_page6:           db     0
674                             <1> RedMaskSize6:             db     8
675                             <1> RedFieldPosition6: db      16
676                             <1> GreenMaskSize6:           db     8
677                             <1> GreenFieldPosition6:      db     8
678                             <1> BlueMaskSize6:            db     8
679                             <1> BlueFieldPosition6:       db     0
680                             <1> RsvdMaskSize6:            db     0
681                             <1> RsvdFieldPosition6:       db     0
682                             <1> DirectColorModeInfo6:     db     0
683                             <1> PhysBasePtr6:             dd     VBE_DISPI_LFB_PHYSICAL_ADDRESS
684                             <1> OffScreenMemOffset6:      dd     0
685                             <1> OffScreenMemSize6: dw      0
686                             <1> LinBytesPerScanLine6:     dw     960
687                             <1> BnkNumberOfPages6:        db     0
688                             <1> LinNumberOfPages6:        db     0
689                             <1> LinRedMaskSize6:   db     8
690                             <1> LinRedFieldPosition6:     db     16
691                             <1> LinGreenMaskSize6: db     8
692                             <1> LinGreenFieldPosition6:   db     8
693                             <1> LinBlueMaskSize6:  db     8
694                             <1> LinBlueFieldPosition6:    db     0
695                             <1> LinRsvdMaskSize6:  db     0
696                             <1> LinRsvdFieldPosition6:    db     0
697                             <1> MaxPixelClock6:           dd     0
698                             <1>
699                             <1>                     dw     0111h ; 640x480x16
700                             <1> ModeAttributes7:    dw     MODE_ATTRIBUTES
701                             <1> WinAAttributes7:    db     WINA_ATTRIBUTES
702                             <1> WinBAttributes7:    db     0
703                             <1> WinGranularity7:    dw     VBE_DISPI_BANK_SIZE_KB
```

```
                        <1> WinSize7:          dw    VBE_DISPI_BANK_SIZE_KB
                        <1> WinASegment7:            dw    VGAMEM_GRAPH
                        <1> WinBSegment7:            dw    0000h
                        <1> WinFuncPtr7:       dd    0
                        <1> BytesPerScanLine7: dw    1280
                        <1> XResolution7:            dw    640
                        <1> YResolution7:            dw    480
                        <1> XCharSize7:        db    8
                        <1> YCharSize7:        db    16
                        <1> NumberOfPlanes7:   db    1
                        <1> BitsPerPixel7:           db    16
                        <1> NumberOfBanks7:          db    10
                        <1> MemoryModel7:            db    VBE_MEMORYMODEL_DIRECT_COLOR
                        <1> BankSize7:         db    0
                        <1> NumberOfImagePages7:     db    26
                        <1> Reserved_page7:          db    0
                        <1> RedMaskSize7:            db    5
                        <1> RedFieldPosition7: db    11
                        <1> GreenMaskSize7:          db    6
                        <1> GreenFieldPosition7:     db    5
                        <1> BlueMaskSize7:           db    5
                        <1> BlueFieldPosition7:      db    0
                        <1> RsvdMaskSize7:           db    0
                        <1> RsvdFieldPosition7:      db    0
                        <1> DirectColorModeInfo7:    db    0
                        <1> PhysBasePtr7:            dd    VBE_DISPI_LFB_PHYSICAL_ADDRESS,
                        <1> OffScreenMemOffset7:     dd    0
                        <1> OffScreenMemSize7: dw    0
                        <1> LinBytesPerScanLine7:    dw    1280
                        <1> BnkNumberOfPages7:       db    0
                        <1> LinNumberOfPages7:       db    0
                        <1> LinRedMaskSize7:   db    5
                        <1> LinRedFieldPosition7:    db    11
                        <1> LinGreenMaskSize7: db    6
                        <1> LinGreenFieldPosition7:  db    5
                        <1> LinBlueMaskSize7:  db    5
                        <1> LinBlueFieldPosition7:   db    0
                        <1> LinRsvdMaskSize7:  db    0
                        <1> LinRsvdFieldPosition7:   db    0
                        <1> MaxPixelClock7:          dd    0
                        <1>
                        <1>                    dw    0112h ; 640x480x24
                        <1> ModeAttributes8:   dw    MODE_ATTRIBUTES
                        <1> WinAAttributes8:   db    WINA_ATTRIBUTES
                        <1> WinBAttributes8:   db    0
                        <1> WinGranularity8:   dw    VBE_DISPI_BANK_SIZE_KB
                        <1> WinSize8:          dw    VBE_DISPI_BANK_SIZE_KB
                        <1> WinASegment8:            dw    VGAMEM_GRAPH
                        <1> WinBSegment8:            dw    0000h
                        <1> WinFuncPtr8:       dd    0
                        <1> BytesPerScanLine8: dw    1920
                        <1> XResolution8:            dw    640
                        <1> YResolution8:            dw    480
                        <1> XCharSize8:        db    8
                        <1> YCharSize8:        db    16
                        <1> NumberOfPlanes8:   db    1
                        <1> BitsPerPixel8:           db    24
                        <1> NumberOfBanks8:          db    15
                        <1> MemoryModel8:            db    VBE_MEMORYMODEL_DIRECT_COLOR
                        <1> BankSize8:         db    0
                        <1> NumberOfImagePages8:     db    17
                        <1> Reserved_page8:          db    0
                        <1> RedMaskSize8:            db    8
                        <1> RedFieldPosition8: db    16
                        <1> GreenMaskSize8:          db    8
                        <1> GreenFieldPosition8:     db    8
                        <1> BlueMaskSize8:           db    8
                        <1> BlueFieldPosition8:      db    0
                        <1> RsvdMaskSize8:           db    0
                        <1> RsvdFieldPosition8:      db    0
                        <1> DirectColorModeInfo8:    db    0
                        <1> PhysBasePtr8:            dd    VBE_DISPI_LFB_PHYSICAL_ADDRESS
                        <1> OffScreenMemOffset8:     dd    0
                        <1> OffScreenMemSize8: dw    0
                        <1> LinBytesPerScanLine8:    dw    1920
                        <1> BnkNumberOfPages8:       db    0
                        <1> LinNumberOfPages8:       db    0
                        <1> LinRedMaskSize8:   db    8
                        <1> LinRedFieldPosition8:    db    16
                        <1> LinGreenMaskSize8: db    8
                        <1> LinGreenFieldPosition8:  db    8
                        <1> LinBlueMaskSize8:  db    8
                        <1> LinBlueFieldPosition8:   db    0
                        <1> LinRsvdMaskSize8:  db    0
                        <1> LinRsvdFieldPosition8:   db    0
                        <1> MaxPixelClock8:          dd    0
                        <1>
                        <1>                    dw    0114h ; 800x600x16
                        <1> ModeAttributes9:   dw    MODE_ATTRIBUTES
                        <1> WinAAttributes9:   db    WINA_ATTRIBUTES
                        <1> WinBAttributes9:   db    0
                        <1> WinGranularity9:   dw    VBE_DISPI_BANK_SIZE_KB
                        <1> WinSize9:          dw    VBE_DISPI_BANK_SIZE_KB
                        <1> WinASegment9:            dw    VGAMEM_GRAPH
                        <1> WinBSegment9:            dw    0000h
                        <1> WinFuncPtr9:       dd    0
                        <1> BytesPerScanLine9: dw    1600
                        <1> XResolution9:            dw    800
                        <1> YResolution9:            dw    600
                        <1> XCharSize9:        db    8
                        <1> YCharSize9:        db    16
                        <1> NumberOfPlanes9:   db    1
                        <1> BitsPerPixel9:           db    16
                        <1> NumberOfBanks9:          db    15
                        <1> MemoryModel9:            db    VBE_MEMORYMODEL_DIRECT_COLOR
```

```
                        <1> BankSize9:          db      0
                        <1> NumberOfImagePages9:        db      16
                        <1> Reserved_page9:             db      0
                        <1> RedMaskSize9:               db      5
                        <1> RedFieldPosition9: db       11
                        <1> GreenMaskSize9:             db      6
                        <1> GreenFieldPosition9:        db      5
                        <1> BlueMaskSize9:              db      5
                        <1> BlueFieldPosition9:         db      0
                        <1> RsvdMaskSize9:              db      0
                        <1> RsvdFieldPosition9:         db      0
                        <1> DirectColorModeInfo9:       db      0
                        <1> PhysBasePtr9:               dd      VBE_DISPI_LFB_PHYSICAL_ADDRESS
                        <1> OffScreenMemOffset9:        dd      0
                        <1> OffScreenMemSize9: dw       0
                        <1> LinBytesPerScanLine9:       dw      1600
                        <1> BnkNumberOfPages9:          db      0
                        <1> LinNumberOfPages9:          db      0
                        <1> LinRedMaskSize9:    db      5
                        <1> LinRedFieldPosition9:       db      11
                        <1> LinGreenMaskSize9: db       6
                        <1> LinGreenFieldPosition9:     db      5
                        <1> LinBlueMaskSize9:   db      5
                        <1> LinBlueFieldPosition9:      db      0
                        <1> LinRsvdMaskSize9:   db      0
                        <1> LinRsvdFieldPosition9:      db      0
                        <1> MaxPixelClock9:             dd      0
                        <1>
                        <1>                     dw      0115h ; 800x600x24
                        <1> ModeAttributes10:   dw      MODE_ATTRIBUTES
                        <1> WinAAttributes10:   db      WINA_ATTRIBUTES
                        <1> WinBAttributes10:   db      0
                        <1> WinGranularity10:   dw      VBE_DISPI_BANK_SIZE_KB
                        <1> WinSize10:          dw      VBE_DISPI_BANK_SIZE_KB
                        <1> WinASegment10:              dw      VGAMEM_GRAPH
                        <1> WinBSegment10:              dw      0000h
                        <1> WinFuncPtr10:               dd      0
                        <1> BytesPerScanLine10:         dw      2400
                        <1> XResolution10:              dw      800
                        <1> YResolution10:              dw      600
                        <1> XCharSize10:        db      8
                        <1> YCharSize10:        db      16
                        <1> NumberOfPlanes10:   db      1
                        <1> BitsPerPixel10:             db      24
                        <1> NumberOfBanks10:    db      22
                        <1> MemoryModel10:              db      VBE_MEMORYMODEL_DIRECT_COLOR
                        <1> BankSize10:         db      0
                        <1> NumberOfImagePages10:       db      10
                        <1> Reserved_page10:    db      0
                        <1> RedMaskSize10:              db      8
                        <1> RedFieldPosition10:         db      16
                        <1> GreenMaskSize10:    db      8
                        <1> GreenFieldPosition10:       db      8
                        <1> BlueMaskSize10:             db      8
                        <1> BlueFieldPosition10:        db      0
                        <1> RsvdMaskSize10:             db      0
                        <1> RsvdFieldPosition10:        db      0
                        <1> DirectColorModeInfo10:      db      0
                        <1> PhysBasePtr10:              dd      VBE_DISPI_LFB_PHYSICAL_ADDRESS
                        <1> OffScreenMemOffset10:       dd      0
                        <1> OffScreenMemSize10:         dw      0
                        <1> LinBytesPerScanLine10:      dw      2400
                        <1> BnkNumberOfPages10:         db      0
                        <1> LinNumberOfPages10:         db      0
                        <1> LinRedMaskSize10:   db      8
                        <1> LinRedFieldPosition10:      db      16
                        <1> LinGreenMaskSize10:         db      8
                        <1> LinGreenFieldPosition10:db          8
                        <1> LinBlueMaskSize10: db       8
                        <1> LinBlueFieldPosition10:     db      0
                        <1> LinRsvdMaskSize10: db       0
                        <1> LinRsvdFieldPosition10:     db      0
                        <1> MaxPixelClock10:    dd      0
                        <1>
                        <1>                     dw      0117h ; 1024x768x16
                        <1> ModeAttributes11:   dw      MODE_ATTRIBUTES
                        <1> WinAAttributes11:   db      WINA_ATTRIBUTES
                        <1> WinBAttributes11:   db      0
                        <1> WinGranularity11:   dw      VBE_DISPI_BANK_SIZE_KB
                        <1> WinSize11:          dw      VBE_DISPI_BANK_SIZE_KB
                        <1> WinASegment11:              dw      VGAMEM_GRAPH
                        <1> WinBSegment11:              dw      0000h
                        <1> WinFuncPtr11:               dd      0
                        <1> BytesPerScanLine11:         dw      2048
                        <1> XResolution11:              dw      1024
                        <1> YResolution11:              dw      768
                        <1> XCharSize11:        db      8
                        <1> YCharSize11:        db      16
                        <1> NumberOfPlanes11:   db      1
                        <1> BitsPerPixel11:             db      16
                        <1> NumberOfBanks11:    db      24
                        <1> MemoryModel11:              db      VBE_MEMORYMODEL_DIRECT_COLOR
                        <1> BankSize11:         db      0
                        <1> NumberOfImagePages11:       db      9
                        <1> Reserved_page11:    db      0
                        <1> RedMaskSize11:              db      5
                        <1> RedFieldPosition11:         db      11
                        <1> GreenMaskSize11:    db      6
                        <1> GreenFieldPosition11:       db      5
                        <1> BlueMaskSize11:             db      5
                        <1> BlueFieldPosition11:        db      0
                        <1> RsvdMaskSize11:             db      0
                        <1> RsvdFieldPosition11:        db      0
                        <1> DirectColorModeInfo11:      db      0
                        <1> PhysBasePtr11:              dd      VBE_DISPI_LFB_PHYSICAL_ADDRESS,
```

```
                        <1> OffScreenMemOffset11:     dd     0
                        <1> OffScreenMemSize11:       dw     0
                        <1> LinBytesPerScanLine11:    dw     2048
                        <1> BnkNumberOfPages11:       db     0
                        <1> LinNumberOfPages11:       db     0
                        <1> LinRedMaskSize11:  db     5
                        <1> LinRedFieldPosition11:    db     11
                        <1> LinGreenMaskSize11:       db     6
                        <1> LinGreenFieldPosition11:db       5
                        <1> LinBlueMaskSize11: db     5
                        <1> LinBlueFieldPosition11:   db     0
                        <1> LinRsvdMaskSize11: db     0
                        <1> LinRsvdFieldPosition11:   db     0
                        <1> MaxPixelClock11:   dd     0
                        <1>
                        <1>                    dw     0118h ; 1024x768x24
                        <1> ModeAttributes12:  dw     MODE_ATTRIBUTES
                        <1> WinAAttributes12:  db     WINA_ATTRIBUTES
                        <1> WinBAttributes12:  db     0
                        <1> WinGranularity12:  dw     VBE_DISPI_BANK_SIZE_KB
                        <1> WinSize12:         dw     VBE_DISPI_BANK_SIZE_KB
                        <1> WinASegment12:             dw     VGAMEM_GRAPH
                        <1> WinBSegment12:             dw     0000h
                        <1> WinFuncPtr12:              dd     0
                        <1> BytesPerScanLine12:        dw     3072
                        <1> XResolution12:             dw     1024
                        <1> YResolution12:             dw     768
                        <1> XCharSize12:       db     8
                        <1> YCharSize12:       db     16
                        <1> NumberOfPlanes12:  db     1
                        <1> BitsPerPixel12:            db     24
                        <1> NumberOfBanks12:   db     36
                        <1> MemoryModel12:             db     VBE_MEMORYMODEL_DIRECT_COLOR
                        <1> BankSize12:        db     0
                        <1> NumberOfImagePages12:      db     6
                        <1> Reserved_page12:   db     0
                        <1> RedMaskSize12:             db     8
                        <1> RedFieldPosition12:        db     16
                        <1> GreenMaskSize12:   db     8
                        <1> GreenFieldPosition12:      db     8
                        <1> BlueMaskSize12:            db     8
                        <1> BlueFieldPosition12:       db     0
                        <1> RsvdMaskSize12:            db     0
                        <1> RsvdFieldPosition12:       db     0
                        <1> DirectColorModeInfo12:     db     0
                        <1> PhysBasePtr12:             dd     VBE_DISPI_LFB_PHYSICAL_ADDRESS
                        <1> OffScreenMemOffset12:      dd     0
                        <1> OffScreenMemSize12:        dw     0
                        <1> LinBytesPerScanLine12:     dw     3072
                        <1> BnkNumberOfPages12:        db     0
                        <1> LinNumberOfPages12:        db     0
                        <1> LinRedMaskSize12:  db     8
                        <1> LinRedFieldPosition12:     db     16
                        <1> LinGreenMaskSize12:        db     8
                        <1> LinGreenFieldPosition12:db       8
                        <1> LinBlueMaskSize12: db     8
                        <1> LinBlueFieldPosition12:    db     0
                        <1> LinRsvdMaskSize12: db     0
                        <1> LinRsvdFieldPosition12:    db     0
                        <1> MaxPixelClock12:   dd     0
                        <1>
                        <1>                    dw     0140h ; 320x200x32
                        <1> ModeAttributes13:  dw     MODE_ATTRIBUTES
                        <1> WinAAttributes13:  db     WINA_ATTRIBUTES
                        <1> WinBAttributes13:  db     0
                        <1> WinGranularity13:  dw     VBE_DISPI_BANK_SIZE_KB
                        <1> WinSize13:         dw     VBE_DISPI_BANK_SIZE_KB
                        <1> WinASegment13:             dw     VGAMEM_GRAPH
                        <1> WinBSegment13:             dw     0000h
                        <1> WinFuncPtr13:              dd     0
                        <1> BytesPerScanLine13:        dw     1280
                        <1> XResolution13:             dw     320
                        <1> YResolution13:             dw     200
                        <1> XCharSize13:       db     8
                        <1> YCharSize13:       db     16
                        <1> NumberOfPlanes13:  db     1
                        <1> BitsPerPixel13:            db     32
                        <1> NumberOfBanks13:   db     4
                        <1> MemoryModel13:             db     VBE_MEMORYMODEL_DIRECT_COLOR
                        <1> BankSize13:        db     0
                        <1> NumberOfImagePages13:      db     64
                        <1> Reserved_page13:   db     0
                        <1> RedMaskSize13:             db     8
                        <1> RedFieldPosition13:        db     16
                        <1> GreenMaskSize13:   db     8
                        <1> GreenFieldPosition13:      db     8
                        <1> BlueMaskSize13:            db     8
                        <1> BlueFieldPosition13:       db     0
                        <1> RsvdMaskSize13:            db     8
                        <1> RsvdFieldPosition13:       db     24
                        <1> DirectColorModeInfo13:     db     VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE
                        <1> PhysBasePtr13:             dd     VBE_DISPI_LFB_PHYSICAL_ADDRESS
                        <1> OffScreenMemOffset13:      dd     0
                        <1> OffScreenMemSize13:        dw     0
                        <1> LinBytesPerScanLine13:     dw     1280
                        <1> BnkNumberOfPages13:        db     0
                        <1> LinNumberOfPages13:        db     0
                        <1> LinRedMaskSize13:  db     8
                        <1> LinRedFieldPosition13:     db     16
                        <1> LinGreenMaskSize13:        db     8
                        <1> LinGreenFieldPosition13:db       8
                        <1> LinBlueMaskSize13: db     8
                        <1> LinBlueFieldPosition13:    db     0
                        <1> LinRsvdMaskSize13: db     8
                        <1> LinRsvdFieldPosition13:    db     24
```

```
1019                            <1> MaxPixelClock13:   dd     0
1020                            <1>
1021                            <1>                    dw     0141h ; 640x400x32
1022                            <1> ModeAttributes14:  dw     MODE_ATTRIBUTES
1023                            <1> WinAAttributes14:  db     WINA_ATTRIBUTES
1024                            <1> WinBAttributes14:  db     0
1025                            <1> WinGranularity14:  dw     VBE_DISPI_BANK_SIZE_KB
1026                            <1> WinSize14:         dw     VBE_DISPI_BANK_SIZE_KB
1027                            <1> WinASegment14:             dw     VGAMEM_GRAPH
1028                            <1> WinBSegment14:             dw     0000h
1029                            <1> WinFuncPtr14:              dd     0
1030                            <1> BytesPerScanLine14:        dw     2560
1031                            <1> XResolution14:             dw     640
1032                            <1> YResolution14:             dw     400
1033                            <1> XCharSize14:       db     8
1034                            <1> YCharSize14:       db     16
1035                            <1> NumberOfPlanes14:  db     1
1036                            <1> BitsPerPixel14:            db     32
1037                            <1> NumberOfBanks14:   db     16
1038                            <1> MemoryModel14:             db     VBE_MEMORYMODEL_DIRECT_COLOR
1039                            <1> BankSize14:        db     0
1040                            <1> NumberOfImagePages14:      db     15
1041                            <1> Reserved_page14:   db     0
1042                            <1> RedMaskSize14:             db     8
1043                            <1> RedFieldPosition14:        db     16
1044                            <1> GreenMaskSize14:   db     8
1045                            <1> GreenFieldPosition14:      db     8
1046                            <1> BlueMaskSize14:            db     8
1047                            <1> BlueFieldPosition14:       db     0
1048                            <1> RsvdMaskSize14:            db     8
1049                            <1> RsvdFieldPosition14:       db     24
1050                            <1> DirectColorModeInfo14:     db     VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE
1051                            <1> PhysBasePtr14:             dd     VBE_DISPI_LFB_PHYSICAL_ADDRESS
1052                            <1> OffScreenMemOffset14:      dd     0
1053                            <1> OffScreenMemSize14:        dw     0
1054                            <1> LinBytesPerScanLine14:     dw     2560
1055                            <1> BnkNumberOfPages14:        db     0
1056                            <1> LinNumberOfPages14:        db     0
1057                            <1> LinRedMaskSize14:  db     8
1058                            <1> LinRedFieldPosition14:     db     16
1059                            <1> LinGreenMaskSize14:        db     8
1060                            <1> LinGreenFieldPosition14:db       8
1061                            <1> LinBlueMaskSize14: db     8
1062                            <1> LinBlueFieldPosition14:    db     0
1063                            <1> LinRsvdMaskSize14: db     8
1064                            <1> LinRsvdFieldPosition14:    db     24
1065                            <1> MaxPixelClock14:   dd     0
1066                            <1>
1067                            <1>                    dw     0142 ; 640x480x32
1068                            <1> ModeAttributes15:  dw     MODE_ATTRIBUTES
1069                            <1> WinAAttributes15:  db     WINA_ATTRIBUTES
1070                            <1> WinBAttributes15:  db     0
1071                            <1> WinGranularity15:  dw     VBE_DISPI_BANK_SIZE_KB
1072                            <1> WinSize15:         dw     VBE_DISPI_BANK_SIZE_KB
1073                            <1> WinASegment15:             dw     VGAMEM_GRAPH
1074                            <1> WinBSegment15:             dw     0000h
1075                            <1> WinFuncPtr15:              dd     0
1076                            <1> BytesPerScanLine15:        dw     2560
1077                            <1> XResolution15:             dw     640
1078                            <1> YResolution15:             dw     480
1079                            <1> XCharSize15:       db     8
1080                            <1> YCharSize15:       db     16
1081                            <1> NumberOfPlanes15:  db     1
1082                            <1> BitsPerPixel15:            db     32
1083                            <1> NumberOfBanks15:   db     19
1084                            <1> MemoryModel15:             db     VBE_MEMORYMODEL_DIRECT_COLOR
1085                            <1> BankSize15:        db     0
1086                            <1> NumberOfImagePages15:      db     12
1087                            <1> Reserved_page15:   db     0
1088                            <1> RedMaskSize15:             db     8
1089                            <1> RedFieldPosition15:        db     16
1090                            <1> GreenMaskSize15:   db     8
1091                            <1> GreenFieldPosition15:      db     8
1092                            <1> BlueMaskSize15:            db     8
1093                            <1> BlueFieldPosition15:       db     0
1094                            <1> RsvdMaskSize15:            db     8
1095                            <1> RsvdFieldPosition15:       db     24
1096                            <1> DirectColorModeInfo15:     db     VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE,
1097                            <1> PhysBasePtr15:             dd     VBE_DISPI_LFB_PHYSICAL_ADDRESS,
1098                            <1> OffScreenMemOffset15:      dd     0
1099                            <1> OffScreenMemSize15:        dw     0
1100                            <1> LinBytesPerScanLine15:     dw     2560
1101                            <1> BnkNumberOfPages15:        db     0
1102                            <1> LinNumberOfPages15:        db     0
1103                            <1> LinRedMaskSize15:  db     8
1104                            <1> LinRedFieldPosition15:     db     16
1105                            <1> LinGreenMaskSize15:        db     8
1106                            <1> LinGreenFieldPosition15:db       8
1107                            <1> LinBlueMaskSize15: db     8
1108                            <1> LinBlueFieldPosition15:    db     0
1109                            <1> LinRsvdMaskSize15: db     8
1110                            <1> LinRsvdFieldPosition15:    db     24
1111                            <1> MaxPixelClock15:   dd     0
1112                            <1>
1113                            <1>                    dw     0143h ; 800x600x32
1114                            <1> ModeAttributes16:  dw     MODE_ATTRIBUTES
1115                            <1> WinAAttributes16:  db     WINA_ATTRIBUTES
1116                            <1> WinBAttributes16:  db     0
1117                            <1> WinGranularity16:  dw     VBE_DISPI_BANK_SIZE_KB
1118                            <1> WinSize16:         dw     VBE_DISPI_BANK_SIZE_KB
1119                            <1> WinASegment16:             dw     VGAMEM_GRAPH
1120                            <1> WinBSegment16:             dw     0000h
1121                            <1> WinFuncPtr16:              dd     0
1122                            <1> BytesPerScanLine16:        dw     3200
1123                            <1> XResolution16:             dw     800
```

```
1124                            <1> YResolution16:           dw     600
1125                            <1> XCharSize16:       db     8
1126                            <1> YCharSize16:       db     16
1127                            <1> NumberOfPlanes16:  db     1
1128                            <1> BitsPerPixel16:          db     32
1129                            <1> NumberOfBanks16:   db     30
1130                            <1> MemoryModel16:           db     VBE_MEMORYMODEL_DIRECT_COLOR
1131                            <1> BankSize16:        db     0
1132                            <1> NumberOfImagePages16:    db     7
1133                            <1> Reserved_page16:   db     0
1134                            <1> RedMaskSize16:           db     8
1135                            <1> RedFieldPosition16:      db     16
1136                            <1> GreenMaskSize16:   db     8
1137                            <1> GreenFieldPosition16:    db     8
1138                            <1> BlueMaskSize16:          db     8
1139                            <1> BlueFieldPosition16:     db     0
1140                            <1> RsvdMaskSize16:          db     8
1141                            <1> RsvdFieldPosition16:     db     24
1142                            <1> DirectColorModeInfo16:   db     VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE,
1143                            <1> PhysBasePtr16:           dd     VBE_DISPI_LFB_PHYSICAL_ADDRESS,
1144                            <1> OffScreenMemOffset16:    dd     0
1145                            <1> OffScreenMemSize16:      dw     0
1146                            <1> LinBytesPerScanLine16:   dw     3200
1147                            <1> BnkNumberOfPages16:      db     0
1148                            <1> LinNumberOfPages16:      db     0
1149                            <1> LinRedMaskSize16:  db     8
1150                            <1> LinRedFieldPosition16:   db     16
1151                            <1> LinGreenMaskSize16:      db     8
1152                            <1> LinGreenFieldPosition16:db      8
1153                            <1> LinBlueMaskSize16: db     8
1154                            <1> LinBlueFieldPosition16:  db     0
1155                            <1> LinRsvdMaskSize16: db     8
1156                            <1> LinRsvdFieldPosition16:  db     24
1157                            <1> MaxPixelClock16:   dd     0
1158                            <1>
1159                            <1>                    dw     0144h ; 1024x768x32
1160                            <1> ModeAttributes17:  dw     MODE_ATTRIBUTES
1161                            <1> WinAAttributes17:  db     WINA_ATTRIBUTES
1162                            <1> WinBAttributes17:  db     0
1163                            <1> WinGranularity17:  dw     VBE_DISPI_BANK_SIZE_KB
1164                            <1> WinSize17:         dw     VBE_DISPI_BANK_SIZE_KB
1165                            <1> WinASegment17:           dw     VGAMEM_GRAPH
1166                            <1> WinBSegment17:           dw     0000h
1167                            <1> WinFuncPtr17:            dd     0
1168                            <1> BytesPerScanLine17:      dw     4096
1169                            <1> XResolution17:           dw     1024
1170                            <1> YResolution17:           dw     768
1171                            <1> XCharSize17:       db     8
1172                            <1> YCharSize17:       db     16
1173                            <1> NumberOfPlanes17:  db     1
1174                            <1> BitsPerPixel17:          db     32
1175                            <1> NumberOfBanks17:   db     48
1176                            <1> MemoryModel17:           db     VBE_MEMORYMODEL_DIRECT_COLOR
1177                            <1> BankSize17:        db     0
1178                            <1> NumberOfImagePages17:    db     4
1179                            <1> Reserved_page17:   db     0
1180                            <1> RedMaskSize17:           db     8
1181                            <1> RedFieldPosition17:      db     16
1182                            <1> GreenMaskSize17:   db     8
1183                            <1> GreenFieldPosition17:    db     8
1184                            <1> BlueMaskSize17:          db     8
1185                            <1> BlueFieldPosition17:     db     0
1186                            <1> RsvdMaskSize17:          db     8
1187                            <1> RsvdFieldPosition17:     db     24
1188                            <1> DirectColorModeInfo17:   db     VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE
1189                            <1> PhysBasePtr17:           dd     VBE_DISPI_LFB_PHYSICAL_ADDRESS
1190                            <1> OffScreenMemOffset17:    dd     0
1191                            <1> OffScreenMemSize17:      dw     0
1192                            <1> LinBytesPerScanLine17:   dw     4096
1193                            <1> BnkNumberOfPages17:      db     0
1194                            <1> LinNumberOfPages17:      db     0
1195                            <1> LinRedMaskSize17:  db     8
1196                            <1> LinRedFieldPosition17:   db     16
1197                            <1> LinGreenMaskSize17:      db     8
1198                            <1> LinGreenFieldPosition17:db      8
1199                            <1> LinBlueMaskSize17: db     8
1200                            <1> LinBlueFieldPosition17:  db     0
1201                            <1> LinRsvdMaskSize17: db     8
1202                            <1> LinRsvdFieldPosition17:  db     24
1203                            <1> MaxPixelClock17:   dd     0
1204                            <1>
1205                            <1>                    dw     0146h ; 320x200x8
1206                            <1> ModeAttributes18:  dw     MODE_ATTRIBUTES
1207                            <1> WinAAttributes18:  db     WINA_ATTRIBUTES
1208                            <1> WinBAttributes18:  db     0
1209                            <1> WinGranularity18:  dw     VBE_DISPI_BANK_SIZE_KB
1210                            <1> WinSize18:         dw     VBE_DISPI_BANK_SIZE_KB
1211                            <1> WinASegment18:           dw     VGAMEM_GRAPH
1212                            <1> WinBSegment18:           dw     0000h
1213                            <1> WinFuncPtr18:            dd     0
1214                            <1> BytesPerScanLine18:      dw     320
1215                            <1> XResolution18:           dw     320
1216                            <1> YResolution18:           dw     200
1217                            <1> XCharSize18:       db     8
1218                            <1> YCharSize18:       db     16
1219                            <1> NumberOfPlanes18:  db     1
1220                            <1> BitsPerPixel18:          db     8
1221                            <1> NumberOfBanks18:   db     1
1222                            <1> MemoryModel18:           db     VBE_MEMORYMODEL_PACKED_PIXEL
1223                            <1> BankSize18:        db     0
1224                            <1> NumberOfImagePages18:    db     255 ; 261 in vbetables.h (03/01/2020) !
1225                            <1> Reserved_page18:   db     0
1226                            <1> RedMaskSize18:           db     0
1227                            <1> RedFieldPosition18:      db     0
1228                            <1> GreenMaskSize18:   db     0
```

```
1229                            <1> GreenFieldPosition18:     db     0
1230                            <1> BlueMaskSize18:           db     0
1231                            <1> BlueFieldPosition18:      db     0
1232                            <1> RsvdMaskSize18:           db     0
1233                            <1> RsvdFieldPosition18:      db     0
1234                            <1> DirectColorModeInfo18:    db     0
1235                            <1> PhysBasePtr18:            dd     VBE_DISPI_LFB_PHYSICAL_ADDRESS
1236                            <1> OffScreenMemOffset18:     dd     0
1237                            <1> OffScreenMemSize18:       dw     0
1238                            <1> LinBytesPerScanLine18:    dw     320
1239                            <1> BnkNumberOfPages18:       db     0
1240                            <1> LinNumberOfPages18:       db     0
1241                            <1> LinRedMaskSize18:  db     0
1242                            <1> LinRedFieldPosition18:    db     0
1243                            <1> LinGreenMaskSize18:       db     0
1244                            <1> LinGreenFieldPosition18:db       0
1245                            <1> LinBlueMaskSize18:        db     0
1246                            <1> LinBlueFieldPosition18: db       0
1247                            <1> LinRsvdMaskSize18:        db     0
1248                            <1> LinRsvdFieldPosition18: db       0
1249                            <1> MaxPixelClock18:   dd     0
1250                            <1>
1251                            <1>                    dw     018Dh ; 1280x720x16
1252                            <1> ModeAttributes19:  dw     MODE_ATTRIBUTES
1253                            <1> WinAAttributes19:  db     WINA_ATTRIBUTES
1254                            <1> WinBAttributes19:  db     0
1255                            <1> WinGranularity19:  dw     VBE_DISPI_BANK_SIZE_KB
1256                            <1> WinSize19:         dw     VBE_DISPI_BANK_SIZE_KB
1257                            <1> WinASegment19:             dw     VGAMEM_GRAPH
1258                            <1> WinBSegment19:             dw     0000h
1259                            <1> WinFuncPtr19:              dd     0
1260                            <1> BytesPerScanLine19:        dw     2560
1261                            <1> XResolution19:             dw     1280
1262                            <1> YResolution19:             dw     720
1263                            <1> XCharSize19:       db     8
1264                            <1> YCharSize19:       db     16
1265                            <1> NumberOfPlanes19:  db     1
1266                            <1> BitsPerPixel19:            db     16
1267                            <1> NumberOfBanks19:   db     29
1268                            <1> MemoryModel19:             db     VBE_MEMORYMODEL_DIRECT_COLOR
1269                            <1> BankSize19:        db     0
1270                            <1> NumberOfImagePages19:      db     8
1271                            <1> Reserved_page19:   db     0
1272                            <1> RedMaskSize19:             db     5
1273                            <1> RedFieldPosition19:        db     11
1274                            <1> GreenMaskSize19:   db     6
1275                            <1> GreenFieldPosition19:      db     5
1276                            <1> BlueMaskSize19:            db     5
1277                            <1> BlueFieldPosition19:       db     0
1278                            <1> RsvdMaskSize19:            db     0
1279                            <1> RsvdFieldPosition19:       db     0
1280                            <1> DirectColorModeInfo19:     db     0
1281                            <1> PhysBasePtr19:             dd     VBE_DISPI_LFB_PHYSICAL_ADDRESS
1282                            <1> OffScreenMemOffset19:      dd     0
1283                            <1> OffScreenMemSize19:        dw     0
1284                            <1> LinBytesPerScanLine19:     dw     2560
1285                            <1> BnkNumberOfPages19:        db     0
1286                            <1> LinNumberOfPages19:        db     0
1287                            <1> LinRedMaskSize19:  db     5
1288                            <1> LinRedFieldPosition19:     db     11
1289                            <1> LinGreenMaskSize19:        db     6
1290                            <1> LinGreenFieldPosition19:db       5
1291                            <1> LinBlueMaskSize19: db     5
1292                            <1> LinBlueFieldPosition19:    db     0
1293                            <1> LinRsvdMaskSize19: db     0
1294                            <1> LinRsvdFieldPosition19:    db     0
1295                            <1> MaxPixelClock19:   dd     0
1296                            <1>
1297                            <1>                    dw     018Eh ; 1280x720x24
1298                            <1> ModeAttributes20:  dw     MODE_ATTRIBUTES
1299                            <1> WinAAttributes20:  db     WINA_ATTRIBUTES
1300                            <1> WinBAttributes20:  db     0
1301                            <1> WinGranularity20:  dw     VBE_DISPI_BANK_SIZE_KB
1302                            <1> WinSize20:         dw     VBE_DISPI_BANK_SIZE_KB
1303                            <1> WinASegment20:             dw     VGAMEM_GRAPH
1304                            <1> WinBSegment20:             dw     0000h
1305                            <1> WinFuncPtr20:              dd     0
1306                            <1> BytesPerScanLine20:        dw     3840
1307                            <1> XResolution20:             dw     1280
1308                            <1> YResolution20:             dw     720
1309                            <1> XCharSize20:       db     8
1310                            <1> YCharSize20:       db     16
1311                            <1> NumberOfPlanes20:  db     1
1312                            <1> BitsPerPixel20:            db     24
1313                            <1> NumberOfBanks20:   db     43
1314                            <1> MemoryModel20:             db     VBE_MEMORYMODEL_DIRECT_COLOR
1315                            <1> BankSize20:        db     0
1316                            <1> NumberOfImagePages20:      db     5
1317                            <1> Reserved_page20:   db     0
1318                            <1> RedMaskSize20:             db     8
1319                            <1> RedFieldPosition20:        db     16
1320                            <1> GreenMaskSize20:   db     8
1321                            <1> GreenFieldPosition20:      db     8
1322                            <1> BlueMaskSize20:            db     8
1323                            <1> BlueFieldPosition20:       db     0
1324                            <1> RsvdMaskSize20:            db     0
1325                            <1> RsvdFieldPosition20:       db     0
1326                            <1> DirectColorModeInfo20:     db     0
1327                            <1> PhysBasePtr20:             dd     VBE_DISPI_LFB_PHYSICAL_ADDRESS
1328                            <1> OffScreenMemOffset20:      dd     0
1329                            <1> OffScreenMemSize20:        dw     0
1330                            <1> LinBytesPerScanLine20:     dw     3840
1331                            <1> BnkNumberOfPages20:        db     0
1332                            <1> LinNumberOfPages20:        db     0
1333                            <1> LinRedMaskSize20:  db     8
```

```
1334                            <1> LinRedFieldPosition20:    db     16
1335                            <1> LinGreenMaskSize20:       db     8
1336                            <1> LinGreenFieldPosition20:db       8
1337                            <1> LinBlueMaskSize20: db     8
1338                            <1> LinBlueFieldPosition20:   db     0
1339                            <1> LinRsvdMaskSize20: db     0
1340                            <1> LinRsvdFieldPosition20:   db     0
1341                            <1> MaxPixelClock20:   dd     0
1342                            <1>
1343                            <1>                    dw     018Fh ; 1280x720x32
1344                            <1> ModeAttributes21:  dw     MODE_ATTRIBUTES
1345                            <1> WinAAttributes21:  db     WINA_ATTRIBUTES
1346                            <1> WinBAttributes21:  db     0
1347                            <1> WinGranularity21:  dw     VBE_DISPI_BANK_SIZE_KB
1348                            <1> WinSize21:         dw     VBE_DISPI_BANK_SIZE_KB
1349                            <1> WinASegment21:             dw     VGAMEM_GRAPH
1350                            <1> WinBSegment21:             dw     0000h
1351                            <1> WinFuncPtr21:              dd     0
1352                            <1> BytesPerScanLine21:        dw     5120
1353                            <1> XResolution21:             dw     1280
1354                            <1> YResolution21:             dw     720
1355                            <1> XCharSize21:       db     8
1356                            <1> YCharSize21:       db     16
1357                            <1> NumberOfPlanes21:  db     1
1358                            <1> BitsPerPixel21:            db     32
1359                            <1> NumberOfBanks21:   db     57
1360                            <1> MemoryModel21:             db     VBE_MEMORYMODEL_DIRECT_COLOR
1361                            <1> BankSize21:        db     0
1362                            <1> NumberOfImagePages21:      db     3
1363                            <1> Reserved_page21:   db     0
1364                            <1> RedMaskSize21:             db     8
1365                            <1> RedFieldPosition21:        db     16
1366                            <1> GreenMaskSize21:   db     8
1367                            <1> GreenFieldPosition21:      db     8
1368                            <1> BlueMaskSize21:            db     8
1369                            <1> BlueFieldPosition21:       db     0
1370                            <1> RsvdMaskSize21:            db     8
1371                            <1> RsvdFieldPosition21:       db     24
1372                            <1> DirectColorModeInfo21:     db     VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE
1373                            <1> PhysBasePtr21:             dd     VBE_DISPI_LFB_PHYSICAL_ADDRESS
1374                            <1> OffScreenMemOffset21:      dd     0
1375                            <1> OffScreenMemSize21:        dw     0
1376                            <1> LinBytesPerScanLine21:     dw     5120
1377                            <1> BnkNumberOfPages21:        db     0
1378                            <1> LinNumberOfPages21:        db     0
1379                            <1> LinRedMaskSize21:  db     8
1380                            <1> LinRedFieldPosition21:     db     16
1381                            <1> LinGreenMaskSize21:        db     8
1382                            <1> LinGreenFieldPosition21:db       8
1383                            <1> LinBlueMaskSize21: db     8
1384                            <1> LinBlueFieldPosition21:    db     0
1385                            <1> LinRsvdMaskSize21: db     8
1386                            <1> LinRsvdFieldPosition21:    db     24
1387                            <1> MaxPixelClock21:   dd     0
1388                            <1>
1389                            <1>                    dw     0190h ; 1920x1080x16
1390                            <1> ModeAttributes22:  dw     MODE_ATTRIBUTES
1391                            <1> WinAAttributes22:  db     WINA_ATTRIBUTES
1392                            <1> WinBAttributes22:  db     0
1393                            <1> WinGranularity22:  dw     VBE_DISPI_BANK_SIZE_KB
1394                            <1> WinSize22:         dw     VBE_DISPI_BANK_SIZE_KB
1395                            <1> WinASegment22:             dw     VGAMEM_GRAPH
1396                            <1> WinBSegment22:             dw     0000h
1397                            <1> WinFuncPtr22:              dd     0
1398                            <1> BytesPerScanLine22:        dw     3840
1399                            <1> XResolution22:             dw     1920
1400                            <1> YResolution22:             dw     1080
1401                            <1> XCharSize22:       db     8
1402                            <1> YCharSize22:       db     16
1403                            <1> NumberOfPlanes22:  db     1
1404                            <1> BitsPerPixel22:            db     16
1405                            <1> NumberOfBanks22:   db     64
1406                            <1> MemoryModel22:             db     VBE_MEMORYMODEL_DIRECT_COLOR,
1407                            <1> BankSize22:        db     0
1408                            <1> NumberOfImagePages22:      db     3
1409                            <1> Reserved_page22:   db     0
1410                            <1> RedMaskSize22:             db     5
1411                            <1> RedFieldPosition22:        db     11
1412                            <1> GreenMaskSize22:   db     6
1413                            <1> GreenFieldPosition22:      db     5
1414                            <1> BlueMaskSize22:            db     5
1415                            <1> BlueFieldPosition22:       db     0
1416                            <1> RsvdMaskSize22:            db     0
1417                            <1> RsvdFieldPosition22:       db     0
1418                            <1> DirectColorModeInfo22:     db     0
1419                            <1> PhysBasePtr22:             dd     VBE_DISPI_LFB_PHYSICAL_ADDRESS
1420                            <1> OffScreenMemOffset22:      dd     0
1421                            <1> OffScreenMemSize22:        dw     0
1422                            <1> LinBytesPerScanLine22:     dw     3840
1423                            <1> BnkNumberOfPages22:        db     0
1424                            <1> LinNumberOfPages22:        db     0
1425                            <1> LinRedMaskSize22:  db     5
1426                            <1> LinRedFieldPosition22:     db     11
1427                            <1> LinGreenMaskSize22:        db     6
1428                            <1> LinGreenFieldPosition22:db       5
1429                            <1> LinBlueMaskSize22: db     5
1430                            <1> LinBlueFieldPosition22:    db     0
1431                            <1> LinRsvdMaskSize22: db     0
1432                            <1> LinRsvdFieldPosition22:    db     0
1433                            <1> MaxPixelClock22:   dd     0
1434                            <1>
1435                            <1>                    dw     0191h ; 1920x1080x24
1436                            <1> ModeAttributes23:  dw     MODE_ATTRIBUTES
1437                            <1> WinAAttributes23:  db     WINA_ATTRIBUTES
1438                            <1> WinBAttributes23:  db     0
```

```
1439                            <1> WinGranularity23:  dw     VBE_DISPI_BANK_SIZE_KB
1440                            <1> WinSize23:         dw     VBE_DISPI_BANK_SIZE_KB
1441                            <1> WinASegment23:             dw     VGAMEM_GRAPH
1442                            <1> WinBSegment23:             dw     0000h
1443                            <1> WinFuncPtr23:              dd     0
1444                            <1> BytesPerScanLine23:        dw     5760
1445                            <1> XResolution23:             dw     1920
1446                            <1> YResolution23:             dw     1080
1447                            <1> XCharSize23:       db     8
1448                            <1> YCharSize23:       db     16
1449                            <1> NumberOfPlanes23:  db     1
1450                            <1> BitsPerPixel23:            db     24
1451                            <1> NumberOfBanks23:   db     95
1452                            <1> MemoryModel23:             db     VBE_MEMORYMODEL_DIRECT_COLOR
1453                            <1> BankSize23:        db     0
1454                            <1> NumberOfImagePages23:      db     1
1455                            <1> Reserved_page23:   db     0
1456                            <1> RedMaskSize23:             db     8
1457                            <1> RedFieldPosition23:        db     16
1458                            <1> GreenMaskSize23:   db     8
1459                            <1> GreenFieldPosition23:      db     8
1460                            <1> BlueMaskSize23:            db     8
1461                            <1> BlueFieldPosition23:       db     0
1462                            <1> RsvdMaskSize23:            db     0
1463                            <1> RsvdFieldPosition23:       db     0
1464                            <1> DirectColorModeInfo23:     db     0
1465                            <1> PhysBasePtr23:             dd     VBE_DISPI_LFB_PHYSICAL_ADDRESS
1466                            <1> OffScreenMemOffset23:      dd     0
1467                            <1> OffScreenMemSize23:        dw     0
1468                            <1> LinBytesPerScanLine23:     dw     5760
1469                            <1> BnkNumberOfPages23:        db     0
1470                            <1> LinNumberOfPages23:        db     0
1471                            <1> LinRedMaskSize23:  db     8
1472                            <1> LinRedFieldPosition23:     db     16
1473                            <1> LinGreenMaskSize23:        db     8
1474                            <1> LinGreenFieldPosition23:db       8
1475                            <1> LinBlueMaskSize23: db     8
1476                            <1> LinBlueFieldPosition23:    db     0
1477                            <1> LinRsvdMaskSize23: db     0
1478                            <1> LinRsvdFieldPosition23:    db     0
1479                            <1> MaxPixelClock23:   dd     0
1480                            <1>
1481                            <1>                    dw     0192h ; 1920x1080x32
1482                            <1> ModeAttributes24:  dw     MODE_ATTRIBUTES
1483                            <1> WinAAttributes24:  db     WINA_ATTRIBUTES
1484                            <1> WinBAttributes24:  db     0
1485                            <1> WinGranularity24:  dw     VBE_DISPI_BANK_SIZE_KB
1486                            <1> WinSize24:         dw     VBE_DISPI_BANK_SIZE_KB
1487                            <1> WinASegment24:             dw     VGAMEM_GRAPH
1488                            <1> WinBSegment24:             dw     0000h
1489                            <1> WinFuncPtr24:              dd     0
1490                            <1> BytesPerScanLine24:        dw     7680
1491                            <1> XResolution24:             dw     1920
1492                            <1> YResolution24:             dw     1080
1493                            <1> XCharSize24:       db     8
1494                            <1> YCharSize24:       db     16
1495                            <1> NumberOfPlanes24:  db     1
1496                            <1> BitsPerPixel24:            db     32
1497                            <1> NumberOfBanks24:   db     127
1498                            <1> MemoryModel24:             db     VBE_MEMORYMODEL_DIRECT_COLOR
1499                            <1> BankSize24:        db     0
1500                            <1> NumberOfImagePages24:      db     1
1501                            <1> Reserved_page24:   db     0
1502                            <1> RedMaskSize24:             db     8
1503                            <1> RedFieldPosition24:        db     16
1504                            <1> GreenMaskSize24:   db     8
1505                            <1> GreenFieldPosition24:      db     8
1506                            <1> BlueMaskSize24:            db     8
1507                            <1> BlueFieldPosition24:       db     0
1508                            <1> RsvdMaskSize24:            db     8
1509                            <1> RsvdFieldPosition24:       db     24
1510                            <1> DirectColorModeInfo24:     db     VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE
1511                            <1> PhysBasePtr24:             dd     VBE_DISPI_LFB_PHYSICAL_ADDRESS
1512                            <1> OffScreenMemOffset24:      dd     0
1513                            <1> OffScreenMemSize24:        dw     0
1514                            <1> LinBytesPerScanLine24:     dw     7680
1515                            <1> BnkNumberOfPages24:        db     0
1516                            <1> LinNumberOfPages24:        db     0
1517                            <1> LinRedMaskSize24:  db     8
1518                            <1> LinRedFieldPosition24:     db     16
1519                            <1> LinGreenMaskSize24:        db     8
1520                            <1> LinGreenFieldPosition24:db       8
1521                            <1> LinBlueMaskSize24: db     8
1522                            <1> LinBlueFieldPosition24:    db     0
1523                            <1> LinRsvdMaskSize24: db     8
1524                            <1> LinRsvdFieldPosition24:    db     24
1525                            <1> MaxPixelClock24:   dd     0
1526                            <1>
1527                            <1> VBE_VESA_MODE_END_OF_LIST: dw   0
1528                            <1>
1529                            <1> %endif
1530                            <1>
1531                            <1> ; 24/11/2020
1532                            <1>
1533                            <1> direct_color_fields:
1534                            <1>       ; 24/11/2020
1535                            <1>
1536                            <1> ; (vbetables-gen.c)
1537                            <1> ; // Direct Color fields
1538                            <1> ; (required for direct/6 and YUV/7 memory models)
1539                            <1> ; switch(pm->depth) {
1540                            <1>
1541                            <1> ;case 8:
1542 00007416 00                <1> r_size_8:  db 0
1543 00007417 00                <1> r_pos_8:   db 0
```

```
1544 00007418 00                <1> g_size_8:  db 0
1545 00007419 00                <1> g_pos_8:   db 0
1546 0000741A 00                <1> b_size_8:  db 0
1547 0000741B 00                <1> b_pos_8:   db 0
1548 0000741C 00                <1> a_size_8:  db 0
1549 0000741D 00                <1> a_pos_8:   db 0
1550                            <1>
1551                            <1> ;case 16:
1552 0000741E 05                <1> r_size_16:  db 5
1553 0000741F 0B                <1> r_pos_16:   db 11
1554 00007420 06                <1> g_size_16:  db 6
1555 00007421 05                <1> g_pos_16:   db 5
1556 00007422 05                <1> b_size_16:  db 5
1557 00007423 00                <1> b_pos_16:   db 0
1558 00007424 00                <1> a_size_16:  db 0
1559 00007425 00                <1> a_pos_16:   db 0
1560                            <1>
1561                            <1> ;case 24:
1562 00007426 08                <1> r_size_24:  db 8
1563 00007427 10                <1> r_pos_24:   db 16
1564 00007428 08                <1> g_size_24:  db 8
1565 00007429 08                <1> g_pos_24:   db 8
1566 0000742A 08                <1> b_size_24:  db 8
1567 0000742B 00                <1> b_pos_24:   db 0
1568 0000742C 00                <1> a_size_24:  db 0
1569 0000742D 00                <1> a_pos_24:   db 0
1570                            <1>
1571                            <1> ;case 32:
1572 0000742E 08                <1> r_size_32:  db 8
1573 0000742F 10                <1> r_pos_32:   db 16
1574 00007430 08                <1> g_size_32:  db 8
1575 00007431 08                <1> g_pos_32:   db 8
1576 00007432 08                <1> b_size_32:  db 8
1577 00007433 00                <1> b_pos_32:   db 0
1578 00007434 08                <1> a_size_32:  db 8
1579 00007435 18                <1> a_pos_32:   db 24
3072                                ;%include 'diskdata.s'    ; DISK (BIOS) DATA (initialized)
3073                                ;;;
3074
3075                                Align 2
3076
3077                                %include 'sysdefs.s' ; 24/01/2015
   1                            <1> ; ****************************************************************************
   2                            <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - SYSTEM DEFINITIONS : sysdefs.s
   3                            <1> ; ----------------------------------------------------------------------------
   4                            <1> ; Last Update: 31/12/2017
   5                            <1> ; ----------------------------------------------------------------------------
   6                            <1> ; Beginning: 24/01/2016
   7                            <1> ; ----------------------------------------------------------------------------
   8                            <1> ; Assembler: NASM version 2.11 (trdos386.s)
   9                            <1> ; ----------------------------------------------------------------------------
  10                            <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
  11                            <1> ; sysdefs.inc (14/11/2015)
  12                            <1> ; ****************************************************************************
  13                            <1>
  14                            <1> ; Retro UNIX 386 v1 Kernel - SYSDEFS.INC
  15                            <1> ; Last Modification: 14/11/2015
  16                            <1> ;
  17                            <1> ; ///////// RETRO UNIX 386 V1 SYSTEM DEFINITIONS ///////////////
  18                            <1> ; (Modified from
  19                            <1> ;     Retro UNIX 8086 v1 system definitions in 'UNIX.ASM', 01/09/2014)
  20                            <1> ; ((UNIX.ASM (RETRO UNIX 8086 V1 Kernel), 11/03/2013 - 01/09/2014))
  21                            <1> ;     UNIX.ASM (MASM 6.11) --> SYSDEFS.INC (NASM 2.11)
  22                            <1> ; ----------------------------------------------------------------------------
  23                            <1> ;
  24                            <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
  25                            <1> ; (Original) Source Code by Ken Thompson (1971-1972)
  26                            <1> ; <Bell Laboratories (17/3/1972)>
  27                            <1> ; <Preliminary Release of UNIX Implementation Document>
  28                            <1> ;
  29                            <1> ; ****************************************************************************
  30                            <1>
  31                            <1> nproc       equ    16  ; number of processes
  32                            <1> nfiles      equ    50
  33                            <1> ntty  equ     8   ; 8+1 -> 8 (10/05/2013)
  34                            <1> nbuf  equ    4   ; 6 ;; 21/08/2015 - 'namei' buffer problem when nbuf > 4
  35                            <1>              ; NOTE: If fd0 super block buffer addres is beyond of the 1st
  36                            <1>              ; 32K, DMA r/w routine or someting else causes a jump to
  37                            <1>              ; kernel panic routine (in 'alloc' routine, in u5.s)
  38                            <1>              ; because of invalid buffer content (r/w error).
  39                            <1>              ; When all buffers are set before the end of the 1st 32k,
  40                            <1>              ; there is no problem!? (14/11/2015)
  41                            <1>
  42                            <1> ;csgmnt     equ    2000h  ; 26/05/2013 (segment of process 1)
  43                            <1> ;core equ    0          ; 19/04/2013
  44                            <1> ;ecore      equ    32768 - 64  ; 04/06/2013 (24/05/2013)
  45                            <1>       ; (if total size of argument list and arguments is 128 bytes)
  46                            <1>       ; maximum executable file size = 32768-(64+40+128-6) = 32530 bytes
  47                            <1>       ; maximum stack size = 40 bytes (+6 bytes for 'IRET' at 32570)
  48                            <1>       ; initial value of user's stack pointer = 32768-64-128-2 = 32574
  49                            <1>       ;     (sp=32768-args_space-2 at the beginning of execution)
  50                            <1>       ; argument list offset = 32768-64-128 = 32576 (if it is 128 bytes)
  51                            <1>       ; 'u' structure offset (for the '/core' dump file) = 32704
  52                            <1>       ; '/core' dump file size = 32768 bytes
  53                            <1>
  54                            <1> ; 08/03/2014
  55                            <1> ;sdsegmnt equ      6C0h  ; 256*16 bytes (swap data segment size for 16 processes)
  56                            <1> ; 19/04/2013 Retro UNIX 8086 v1 feaure only !
  57                            <1> ;;sdsegmnt equ     740h  ; swap data segment (for user structures and registers)
  58                            <1>
  59                            <1> ; 30/08/2013
  60                            <1> time_count equ 4 ; 10 --> 4 01/02/2014
  61                            <1>
  62                            <1> ; 05/02/2014
  63                            <1> ; process status
```

```
64                                  <1> ;SFREE      equ 0
65                                  <1> ;SRUN equ 1
66                                  <1> ;SWAIT      equ 2
67                                  <1> ;SZOMB      equ 3
68                                  <1> ;SSLEEP     equ 4 ; Retro UNIX 8086 V1 extension (for sleep and wakeup)
69                                  <1>
70                                  <1> ; 09/03/2015
71                                  <1> userdata equ 80000h ; user structure data address for current user ; temporary
72                                  <1> swap_queue equ 90000h - 2000h ; swap queue address ; temporary
73                                  <1> swap_alloc_table equ 0D0000h  ;  swap allocation table address ; temporary
74                                  <1>
75                                  <1> ; 17/09/2015
76                                  <1> ESPACE equ 48 ; [u.usp] (at 'sysent') - [u.sp] value for error return
77                                  <1>
78                                  <1> ; 31/12/2017
79                                  <1> ; 19/02/2017
80                                  <1> ; 15/10/2016
81                                  <1> ; 20/05/2016
82                                  <1> ; 19/05/2016
83                                  <1> ; 18/05/2016
84                                  <1> ; 29/04/2016
85                                  <1> ; TRDOS 386 (TRDOS v2.0) system calls - temporary List
86                                  <1> ; 14/07/2013 - 21/09/2015 (Retro UNIX 8086 & 386 system calls)
87                                  <1> _ver  equ 0 ; Get TRDOS version (v2.0)
88                                  <1> _exit       equ 1
89                                  <1> _fork       equ 2
90                                  <1> _read       equ 3
91                                  <1> _write      equ 4
92                                  <1> _open equ 5
93                                  <1> _close      equ 6
94                                  <1> _wait       equ 7
95                                  <1> _creat      equ 8
96                                  <1> _rename     equ 9  ; TRDOS 386, Rename File (31/12/2017)
97                                  <1> _delete     equ 10 ; TRDOS 386, Delete File (29/12/2017)
98                                  <1> _exec equ 11
99                                  <1> _chdir      equ 12
100                                 <1> _time       equ 13 ; TRDOS 386, Get Sys Date&Time (30/12/2017)
101                                 <1> _mkdir      equ 14
102                                 <1> _chmod      equ 15 ; TRDOS 386, Change Attributes (30/12/2017)
103                                 <1> _rmdir      equ 16 ; TRDOS 386, Remove Directory (29/12/2017)
104                                 <1> _break      equ 17
105                                 <1> _drive      equ 18 ; TRDOS 386, Get/Set Current Drv (30/12/2017)
106                                 <1> _seek equ 19
107                                 <1> _tell       equ 20
108                                 <1> _mem  equ 21 ; TRDOS 386, Get Total&Free Mem (31/12/2017)
109                                 <1> _prompt     equ 22 ; TRDOS 386, Change Cmd Prompt (31/12/2017)
110                                 <1> _path equ 23 ; TRDOS 386, Get/Set Run Path (31/12/2017)
111                                 <1> _env  equ 24 ; TRDOS 386, Get/Set Env Vars (31/12/2017)
112                                 <1> _stime      equ 25 ; TRDOS 386, Set Sys Date&Time (30/12/2017)
113                                 <1> _quit equ 26
114                                 <1> _intr equ 27
115                                 <1> _dir  equ 28 ; TRDOS 386, Get Curr Drive&Dir (30/12/2017)
116                                 <1> _emt  equ 29
117                                 <1> _ldrvt      equ 30 ; TRDOS 386, Get Logical DOS DDT (30/12/2017)
118                                 <1> _video  equ 31 ; TRDOS 386 Video Functions (16/05/2016)
119                                 <1> _audio      equ 32 ; TRDOS 386 Video Functions (16/05/2016)
120                                 <1> _timer      equ 33 ; TRDOS 386 Timer Functions (18/05/2016)
121                                 <1> _sleep      equ 34 ; Retro UNIX 8086 v1 feature only !
122                                 <1> _msg  equ 35 ; Retro UNIX 386 v1 feature only !
123                                 <1> _geterr     equ 36 ; Retro UNIX 386 v1 feature only !
124                                 <1> _fpsave equ 37 ; TRDOS 386 FPU state option (28/02/2017)
125                                 <1> _pri  equ 38 ; change priority - TRDOS 386 (20/05/2016)
126                                 <1> _rele equ 39 ; TRDOS 386 (19/05/2016)
127                                 <1> _fff  equ 40 ; Find First File - TRDOS 386 (15/10/2016)
128                                 <1> _fnf  equ 41 ; Find Next File - TRDOS 386 (15/10/2016)
129                                 <1> _alloc      equ 42 ; Allocate memory - TRDOS 386 (19/02/2017)
130                                 <1>              ; TRDOS 386 (19/02/2017) DMA buff fuctions
131                                 <1> _dalloc equ 43 ; Deallocate mem - TRDOS 386 (19/02/2017)
132                                 <1> _calbac equ 44 ; Set IRQ callback - TRDOS 386 (20/02/2017)
133                                 <1> _dma  equ 45 ; DMA service - TRDOS 386 (20/08/2017)
134                                 <1>
135                                 <1> %macro sys 1-4
136                                 <1>     ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
137                                 <1>     ; 03/09/2015
138                                 <1>     ; 13/04/2015
139                                 <1>     ; Retro UNIX 386 v1 system call.
140                                 <1>     %if %0 >= 2
141                                 <1>         mov ebx, %2
142                                 <1>         %if %0 >= 3
143                                 <1>             mov ecx, %3
144                                 <1>             %if %0 = 4
145                                 <1>                mov edx, %4
146                                 <1>             %endif
147                                 <1>         %endif
148                                 <1>     %endif
149                                 <1>     mov eax, %1
150                                 <1>     ;int 30h
151                                 <1>     int 40h ; TRDOS 386 (TRDOS v2.0)
152                                 <1> %endmacro
153                                 <1>
154                                 <1> ; TRDOS 386 system calls, interrupt number
155                                 <1> ; 25/12/2016
156                                 <1> SYSCALL_INT_NUM   equ '40' ; '40h'
157                                 <1>
158                                 <1> ; 13/05/2015 - ERROR CODES
159                                 <1> ERR_FILE_NOT_OPEN  equ 10 ; 'file not open !' error
160                                 <1> ERR_FILE_ACCESS    equ 11 ; 'permission denied !' error
161                                 <1> ; 14/05/2015
162                                 <1> ERR_DIR_ACCESS     equ 11 ; 'permission denied !' error
163                                 <1> ERR_FILE_NOT_FOUND equ 12 ; 'file not found !' error
164                                 <1> ERR_TOO_MANY_FILES equ 13 ; 'too many open files !' error
165                                 <1> ERR_DIR_EXISTS     equ 14 ; 'directory already exists !' error
166                                 <1> ; 16/05/2015
167                                 <1> ERR_DRV_NOT_RDY    equ 15 ; 'drive not ready !' error
168                                 <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
```

```
122                             <1> LD_StartSector equ 108
123                             <1> LD_TotalSectors equ 112
124                             <1> LD_FreeSectors equ 116
125                             <1> LD_Clusters equ 120
126                             <1> LD_PartitionEntry equ 124
127                             <1> LD_DParamEntry equ 125
128                             <1> LD_MediaChanged equ 126
129                             <1> LD_CDirLevel equ 127
130                             <1> LD_CurrentDirectory equ 128
131                             <1>
132                             <1> ; Singlix FS Extensions to DOS Logical Disks
133                             <1> ; 03/01/2010 (LD_BPB compatibility for CHS r/w)
134                             <1>
135                             <1> LD_FS_Name equ 0
136                             <1> LD_FS_DiskType equ 1
137                             <1> LD_FS_PhyDrvNo equ 2
138                             <1> LD_FS_FATType equ 3
139                             <1> LD_FS_FSType equ 4
140                             <1> LD_FS_LBAYes equ 5
141                             <1> LD_FS_BPB equ 6
142                             <1> LD_FS_MediaAttrib equ 6
143                             <1> LD_FS_VersionMajor equ 7
144                             <1> LD_FS_RootDirD equ 8
145                             <1> LD_FS_MATLocation equ 12
146                             <1> LD_FS_Reserved1 equ 16 ;1 reserved byte
147                             <1> LD_FS_BytesPerSec equ 17 ; LD_BPB + 0Bh
148                             <1> LD_FS_Reserved2 equ 19 ;2 reserved byte
149                             <1> LD_FS_DATLocation equ 20
150                             <1> LD_FS_DATSectors equ 24
151                             <1> LD_FS_Reserved3 equ 28 ;3 reserved word
152                             <1> LD_FS_SecPerTrack equ 30 ; LD_BPB + 18h
153                             <1> LD_FS_NumHeads equ 32    ; LD_BPB + 1Ah
154                             <1> LD_FS_UnDelDirD equ 34
155                             <1> LD_FS_Reserved4 equ 38 ;4 reserved word
156                             <1> LD_FS_VolumeSerial equ 40
157                             <1> LD_FS_VolumeName equ 44
158                             <1> LD_FS_BeginSector equ 108
159                             <1> LD_FS_VolumeSize equ 112
160                             <1> LD_FS_FreeSectors equ 116
161                             <1> LD_FS_FirstFreeSector equ 120
162                             <1> LD_FS_PartitionEntry equ 124
163                             <1> LD_FS_DParamEntry equ 125
164                             <1> LD_FS_MediaChanged equ 126
165                             <1> LD_FS_CDirLevel equ 127
166                             <1> LD_FS_CDIR_Converted equ 128
167                             <1>
168                             <1> ; Valid FAT Types
169                             <1> FS_FAT12 equ 1
170                             <1> FS_FAT16_CHS equ 2
171                             <1> FS_FAT32_CHS equ 3
172                             <1> FS_FAT16_LBA equ 4
173                             <1> FS_FAT32_LBA equ 5
174                             <1>
175                             <1> ; Cursor Location
176                             <1> CCCpointer equ  0450h   ; BIOS data, current cursor column
177                             <1> ; FAT Clusters EOC sign
178                             <1> FAT12EOC equ 0FFFh
179                             <1> FAT16EOC equ 0FFFFh
180                             <1> ;FAT32EOC equ 0FFFFFFFh ; It is not direct usable for 8086 code
181                             <1> ; BAD Cluster
182                             <1> FAT12BADC equ 0FF7h
183                             <1> FAT16BADC equ 0FFF7h
184                             <1> ;FAT32BADC equ 0FFFFFF7h ; It is not direct usable for 8086 code
185                             <1> ; MS-DOS FAT16 FS (Maximum Possible) Last Cluster Number= 0FFF6h
186                             <1>
187                             <1> ; TRFS
188                             <1>
189                             <1> bs_FS_JmpBoot equ 0 ; jmp short bsBootCode
190                             <1>                     ; db 0EBh, db 3Fh, db 90h
191                             <1> bs_FS_Identifier equ 3  ; db 'FS', db 0
192                             <1> bs_FS_BytesPerSec equ 6 ; dw 512
193                             <1> bs_FS_MediaAttrib equ 8 ; db 3
194                             <1> bs_FS_PartitionID equ 9 ; db 0A1h
195                             <1> bs_FS_VersionMaj equ 10 ; db 01h
196                             <1> bs_FS_VersionMin equ 11 ; db 0
197                             <1> bs_FS_BeginSector equ 12   ; dd 0
198                             <1> bs_FS_VolumeSize equ 16 ; dd 2880
199                             <1> bs_FS_StartupFD equ 20 ; dd 0
200                             <1> bs_FS_MATLocation equ 24 ; dd 1
201                             <1> bs_FS_RootDirD equ 28 ; dd 8
202                             <1> bs_FS_SystemConfFD equ 32 ; dd 0
203                             <1> bs_FS_SwapFD equ 36 ; dd 0
204                             <1> bs_FS_UnDelDirD equ 40 ; dd 0
205                             <1> bs_FS_DriveNumber equ 44 ; db 0
206                             <1> bs_FS_LBA_Ready equ 45 ; db 0
207                             <1> bs_FS_MagicWord equ 46
208                             <1> bs_FS_SecPerTrack equ 46 ; db 0A1h
209                             <1> bs_FS_Heads equ 47 ; db 01h
210                             <1> bs_FS_OperationSys equ 48 ; db "TR-SINGLIX v1.0b"
211                             <1> bs_FS_Terminator equ 64 ; db 0
212                             <1> bs_FS_BootCode equ 65
213                             <1>
214                             <1> FS_MAT_DATLocation equ 12
215                             <1> FS_MAT_DATScount equ 16
216                             <1> FS_MAT_FreeSectors equ 20
217                             <1> FS_MAT_FirstFreeSector equ 24
218                             <1> FS_RDT_VolumeSerialNo equ 28
219                             <1> FS_RDT_VolumeName equ 64
220                             <1>
221                             <1> ; FAT12 + FAT16 + FAT32
222                             <1> BS_JmpBoot equ 0
223                             <1> BS_OEMName equ 3
224                             <1> BPB_BytsPerSec equ 11
225                             <1> BPB_SecPerClust equ 13
226                             <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 - v2.0.3) - SYS INIT : trdosk1.s
                        <1> ; ----------------------------------------------------------------------------
                        <1> ; Last Update: 06/12/2020
                        <1> ; ----------------------------------------------------------------------------
                        <1> ; Beginning: 04/01/2016
                        <1> ; ----------------------------------------------------------------------------
                        <1> ; Assembler: NASM version 2.15 (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>       ; 20/01/2018  (v2.0.1)
                        <1>       ; 23/01/2017  (v2.0.0)
                        <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)
00007436 B036           <1>       mov   al, 00110110b ; 36h
00007438 E643           <1>       out   43h, al
0000743A 31C0           <1>       xor   eax, eax  ; sub     al, al ; 0
0000743C E640           <1>       out   40h, al ; LB
0000743E E640           <1>       out   40h, al ; HB
                        <1>       ;
                        <1>       ; 30/03/2016
                        <1>       ; Clear Logical DOS Disk Description Tables Area
                        <1>       ;xor   eax, eax
00007440 BF00010900     <1>       mov   edi, Logical_DOSDisks
00007445 B980060000     <1>       mov   ecx, 6656/4 ; 26*256 = 6656 bytes
0000744A F3AB           <1>       rep   stosd ; 1664 times 4 bytes
                        <1>
0000744C B83F3A2F00     <1>       mov   eax, '?:/'
00007451 A3[878A0100]   <1>       mov   [Current_Dir_Drv], eax
                        <1>
                        <1>       ; Logical DRV INIT (only for hard disks)
```

```
00007456 E825040000          <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>
0000745B BE00010900          <1>       mov    esi, Logical_DOSDisks
00007460 B001                <1>       mov    al, 1 ; Initialization sign (invalid_fd_parameter)
00007462 83C67E              <1>       add    esi, LD_MediaChanged ; Media Change Status = 1 (init needed)
00007465 8806                <1>       mov    [esi], al ; A:
00007467 81C600010000        <1>       add    esi, 100h
0000746D 8806                <1>       mov    [esi], al ; B:
                             <1>
                             <1> _current_drive_bootdisk:
0000746F 8A15[FA6D0000]      <1>       mov    dl, [boot_drv] ; physical drive number
00007475 80FAFF              <1>       cmp    dl, 0FFh
00007478 740A                <1>       je     short _last_dos_diskno_check
                             <1> _boot_drive_check:
0000747A 80FA80              <1>       cmp    dl, 80h
0000747D 7218                <1>       jb     short _current_drive_a
0000747F 80EA7E              <1>       sub    dl, 7Eh ; C = 2 , D = 3
00007482 EB13                <1>       jmp    short _current_drive_a
                             <1>
                             <1> _last_dos_diskno_check:
00007484 8A15[50400100]      <1>       mov    dl, [Last_DOS_DiskNo]
0000748A 80FA02              <1>       cmp    dl, 2
0000748D 7706                <1>       ja     short _current_drive_c
0000748F 7406                <1>       je     short _current_drive_a
00007491 30D2                <1>       xor    dl, dl ; A:
00007493 EB02                <1>       jmp    short _current_drive_a
                             <1>
                             <1> _current_drive_c:
00007495 B202                <1>       mov    dl, 2 ; C:
                             <1>
                             <1> _current_drive_a:
00007497 8815[FB6D0000]      <1>       mov    [drv], dl
0000749D BE[52400100]        <1>         mov     esi, msg_CRLF_temp
000074A2 E8AE000000          <1>       call   print_msg
                             <1>
000074A7 8A15[FB6D0000]      <1>       mov    dl, [drv]
                             <1> _default_drive_c:
000074AD E82C0C0000          <1>       call   change_current_drive
000074B2 731C                <1>       jnc    short _start_mainprog
                             <1>
                             <1> _drv_not_ready_error:
000074B4 BE[0D430100]        <1>       mov    esi, msgl_drv_not_ready
000074B9 E897000000          <1>       call   print_msg
                             <1>         ;jmp _end_of_mainprog
                             <1>
                             <1>       ; 20/01/2018
000074BE B202                <1>       mov    dl, 2
000074C0 3815[FB6D0000]      <1>       cmp    [drv], dl
000074C6 736B                <1>       jnb    short _end_of_mainprog
000074C8 8815[FB6D0000]      <1>       mov    [drv], dl
000074CE EBDD                <1>       jmp    short _default_drive_c
                             <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
000074D0 66B80100            <1>       mov    ax, 1
000074D4 A2[B3030300]        <1>       mov    [u.uno], al
000074D9 66A3[4E030300]      <1>       mov    [mpid], ax
000074DF 66A3[20000300]      <1>       mov    [p.pid], ax
000074E5 A2[B0000300]        <1>       mov    [p.stat], al
000074EA C605[A8030300]04    <1>       mov    byte [u.quant], time_count  ; 07/01/2017
                             <1>       ;
000074F1 A1[C0890100]        <1>       mov    eax, [k_page_dir]
000074F6 A3[B8030300]        <1>       mov    [u.pgdir], eax ; reset
                             <1>       ;
000074FB E8D3E6FFFF          <1>       call   allocate_page
00007500 0F82B5000000        <1>       jc     panic
00007506 A3[B4030300]        <1>       mov    [u.upage], eax ; user structure page
0000750B A3[C0000300]        <1>       mov    [p.upage], eax
00007510 E838E7FFFF          <1>       call   clear_page
                             <1>       ;
                             <1>       ; 24/08/2015
00007515 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
0000751B BF00300900          <1>       mov    edi, Env_Page ; 93000h
00007520 B980000000          <1>       mov    ecx, Env_Page_Size / 4    ; 512/4  (4096/4)

00007525 31C0                <1>       xor    eax, eax
00007527 F3AB                <1>       rep    stosd
                             <1>
                             <1>       ; 14/04/2016
00007529 E807350000          <1>       call   mainprog_startup_configuration
                             <1>
0000752E E8EC0C0000          <1>         call    dos_prompt
                             <1>
                             <1> _end_of_mainprog:
00007533 BE[52400100]        <1>         mov     esi, msg_CRLF_temp
00007538 E818000000          <1>       call   print_msg
0000753D BE[58400100]        <1>       mov    esi, mainprog_Version
00007542 E80E000000          <1>       call   print_msg
                             <1>       ; 24/01/2016
00007547 28E4                <1>       sub    ah, ah
```

```
00007549 E89799FFFF          <1>       call   int16h ; call getch
0000754E E9779EFFFF          <1>       jmp    cpu_reset
                             <1>
00007553 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>       ;
00007555 8A3D[EE890100]      <1>       mov    bh, [ACTIVE_PAGE] ; 04/01/2016 (ptty)
                             <1>       ;mov   bl, 07h ; Black background, light gray forecolor
                             <1>
0000755B AC                  <1>       lodsb
                             <1> pmsg1:
0000755C 56                  <1>       push   esi
                             <1>       ;mov   bh, [ACTIVE_PAGE] ; 04/01/2016 (ptty)
0000755D B307                <1>       mov    bl, 07h ; Black background, light gray forecolor
0000755F E899ADFFFF          <1>       call   _write_tty
00007564 5E                  <1>       pop    esi
00007565 AC                  <1>       lodsb
00007566 20C0                <1>       and    al, al
00007568 75F2                <1>       jnz    short pmsg1
0000756A C3                  <1>       retn
                             <1>
                             <1> clear_screen:
                             <1>       ; 06/12/2020
                             <1>       ; 03/12/2020 (TRDOS 386 v2.0.3)
                             <1>       ; 13/05/2016
                             <1>       ; 30/01/2016
                             <1>       ; 24/01/2016
                             <1>       ; 04/01/2016
0000756B 0FB61D[EE890100]    <1>       movzx  ebx, byte [ACTIVE_PAGE] ; video page number (0 to 7)
00007572 8AA3[DB6F0000]      <1>       mov    ah, [ebx+vmode] ; default = 03h (80x25 text)
00007578 80FC04              <1>       cmp    ah, 4
0000757B 7205                <1>       jb     short cls1
0000757D 80FC07              <1>       cmp    ah, 7
00007580 7530                <1>       jne    short vga_clear
                             <1> cls1:
                             <1>       ;mov   bh, bl
                             <1>       ;mov   bl, 7
00007582 3A25[CA6F0000]      <1>       cmp    ah, [CRT_MODE] ; current video mode ?
00007588 740E                <1>       je     short cls2 ; yes (current video mode = 3)
                             <1>       ;;call set_mode_3 ; set video mode to 3 (& clear screen)
                             <1>       ;;retn
                             <1>       ; 06/12/2020
0000758A 803D[1C120300]00    <1>       cmp    byte [pmi32], 0
00007591 771F                <1>       ja     short vga_clear
00007593 E96FADFFFF          <1>       jmp    set_mode_3
                             <1> cls2:
00007598 88DF                <1>       mov    bh, bl ; video page (0 to 7)
0000759A B307                <1>       mov    bl, 07h ; attribute to be used on blanked line
0000759C 28C0                <1>       sub    al, al ; 0 =  entire window
0000759E 6631C9              <1>       xor    cx, cx
000075A1 66BA4F18            <1>       mov    dx, 184Fh
000075A5 E893AAFFFF          <1>       call   _scroll_up ; 24/01/2016
                             <1>       ;
                             <1>       ;mov   bh, [ACTIVE_PAGE] ; video page number (0 to 7)
000075AA 6631D2              <1>       xor    dx, dx
                             <1>       ;call  _set_cpos ; 24/01/2016
                             <1>       ;;retn
                             <1>       ; 03/12/2020
000075AD E9E5ADFFFF          <1>       jmp    _set_cpos ; returns to the caller of this proc
                             <1> ;cls3:
                             <1> ;     retn
                             <1>       ; 06/12/2020
                             <1> vga_clear:
                             <1>       ; 03/12/2020
                             <1>       ; set mode by using _int10h
                             <1>       ; (also clears screen)
000075B2 88E0                <1>       mov    al, ah
000075B4 28E4                <1>       sub    ah, ah  ; set current video mode
                             <1>       ;call  _int10h ; simulates int 10h in TRDOS 386 kernel
                             <1>       ;jmp   short cls3
000075B6 E9AEA1FFFF          <1>       jmp    _int10h ; returns to the caller of this proc
                             <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)
000075BB BE[174D0100]        <1>       mov    esi, panic_msg
000075C0 E890FFFFFF          <1>       call   print_msg
                             <1> key_to_reboot:
                             <1>         ; 24/01/2016
000075C5 28E4                <1>         sub     ah, ah
000075C7 E81999FFFF          <1>         call    int16h ; call   getch
                             <1>         ; wait for a character from the current tty
                             <1>       ;
000075CC B00A                <1>       mov    al, 0Ah
000075CE 8A3D[EE890100]      <1>       mov    bh, [ptty] ; [ACTIVE_PAGE]
000075D4 B307                <1>       mov    bl, 07h ; Black background,
                             <1>                    ; light gray forecolor
000075D6 E822ADFFFF          <1>       call   _write_tty
000075DB E9EA9DFFFF          <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>             ;
000075E0 66833D[AA030300]00 <1>       cmp   word [u.intr], 0
000075E8 7645               <1>       jna   short cbrk4
                            <1> cbrk0:
                            <1>       ; 12/11/2015
                            <1>       ; 06/12/2013
000075EA 66833D[AC030300]00 <1>       cmp   word [u.quit], 0
000075F2 743B               <1>       jz    short cbrk4
                            <1>       ;
                            <1>       ; 20/09/2013
000075F4 6650               <1>       push  ax
000075F6 A0[EE890100]       <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>       ;
000075FB 3C08               <1>       cmp   al, 8 ; serial port tty nums > 7
000075FD 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>       ;
000075FF 52                 <1>       push  edx
00007600 0FB615[B3030300]   <1>       movzx edx, byte [u.uno]
00007607 3A82[7F000300]     <1>       cmp     al, [edx+p.ttyc-1] ; console tty (rw)
0000760D 5A                 <1>       pop   edx
0000760E 7412               <1>       je    short cbrk2
                            <1> cbrk1:
00007610 FEC0               <1>       inc   al  ; [u.ttyp] : 1 based tty number
                            <1>       ; 06/12/2013
00007612 3A05[94030300]     <1>       cmp   al, [u.ttyp] ; recent open tty (r)
00007618 7408               <1>       je    short cbrk2
0000761A 3A05[95030300]     <1>         cmp     al, [u.ttyp+1] ; recent open tty (w)
00007620 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>       ;
00007622 6631C0             <1>       xor   ax, ax ; 0
00007625 6648               <1>       dec   ax
                            <1>       ; 0FFFFh = 'ctrl+brk' keystroke
00007627 66A3[AC030300]     <1>       mov   [u.quit], ax
                            <1> cbrk3:
0000762D 6658               <1>       pop   ax
                            <1> cbrk4:
0000762F 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>
00007630 E875F4FFFF         <1>       call  get_rtc_time        ; Return Current Time
00007635 86E9               <1>         xchg      ch,cl
00007637 66890D[8E860100]   <1>         mov [hour], cx
0000763E 86F2               <1>         xchg      dh,dl
00007640 668915[92860100]   <1>         mov [second], dx
                            <1>       ;
00007647 E8CFF4FFFF         <1>         call      get_rtc_date        ; Return Current Date
0000764C 86E9               <1>         xchg      ch,cl
0000764E 66890D[88860100]   <1>         mov [year], cx
00007655 86F2               <1>         xchg      dh,dl
00007657 668915[8A860100]   <1>         mov [month], dx
                            <1>       ;
0000765E 66B93030           <1>       mov   cx, 3030h
                            <1>       ;
00007662 A0[8E860100]       <1>       mov   al, [hour] ; Hour
                            <1>             ; AL <= BCD number)
00007667 D410               <1>         db  0D4h,10h            ; Undocumented inst. AAM
                            <1>                                 ; AH = AL / 10h
```

```
                                <1>                                     ; AL = AL MOD 10h
00007669 D50A                   <1>         aad ; AX= AH*10+AL
0000766B A2[8E860100]           <1>       mov   [hour], al
00007670 A0[8F860100]           <1>       mov   al, [hour+1] ; Minute
                                <1>             ; AL <= BCD number)
00007675 D410                   <1>         db  0D4h,10h            ; Undocumented inst. AAM
                                <1>                                 ; AH = AL / 10h
                                <1>                                 ; AL = AL MOD 10h
00007677 D50A                   <1>         aad ; AX= AH*10+AL
00007679 A2[90860100]           <1>       mov   [minute], al
0000767E A0[92860100]           <1>       mov   al, [second] ; Second
                                <1>             ; AL <= BCD number)
00007683 D410                   <1>         db  0D4h,10h            ; Undocumented inst. AAM
                                <1>                                 ; AH = AL / 10h
                                <1>                                 ; AL = AL MOD 10h
00007685 D50A                   <1>         aad ; AX= AH*10+AL
00007687 A2[92860100]           <1>       mov   [second], al
0000768C 66A1[88860100]         <1>       mov   ax, [year] ; Year (century)
00007692 6650                   <1>         push      ax
                                <1>             ; AL <= BCD number)
00007694 D410                   <1>         db  0D4h,10h            ; Undocumented inst. AAM
                                <1>                                 ; AH = AL / 10h
                                <1>                                 ; AL = AL MOD 10h
00007696 D50A                   <1>         aad ; AX= AH*10+AL
00007698 B464                   <1>       mov   ah, 100
0000769A F6E4                   <1>       mul   ah
0000769C 66A3[88860100]         <1>       mov   [year], ax
000076A2 6658                   <1>       pop   ax
000076A4 88E0                   <1>       mov   al, ah
                                <1>             ; AL <= BCD number)
000076A6 D410                   <1>         db  0D4h,10h            ; Undocumented inst. AAM
                                <1>                                 ; AH = AL / 10h
                                <1>                                 ; AL = AL MOD 10h
000076A8 D50A                   <1>         aad ; AX= AH*10+AL
000076AA 660105[88860100]       <1>       add   [year], ax
000076B1 A0[8A860100]           <1>       mov   al, [month] ; Month
                                <1>             ; AL <= BCD number)
000076B6 D410                   <1>         db  0D4h,10h            ; Undocumented inst. AAM
                                <1>                                 ; AH = AL / 10h
                                <1>                                 ; AL = AL MOD 10h
000076B8 D50A                   <1>         aad ; AX= AH*10+AL
000076BA A2[8A860100]           <1>       mov   [month], al
000076BF A0[8B860100]           <1>         mov     al, [month+1]         ; Day
                                <1>             ; AL <= BCD number)
000076C4 D410                   <1>         db  0D4h,10h            ; Undocumented inst. AAM
                                <1>                                 ; AH = AL / 10h
                                <1>                                 ; AL = AL MOD 10h
000076C6 D50A                   <1>         aad ; AX= AH*10+AL
000076C8 A2[8C860100]           <1>         mov     [day], al
                                <1>
000076CD C3                     <1>       retn  ; 30/12/2017
                                <1>
                                <1> epoch:
000076CE 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
000076D3 29C0                   <1>       sub   eax, eax
000076D5 66A1[88860100]         <1>       mov   ax, [year]
000076DB 662DB207               <1>       sub   ax, 1970
000076DF BA6D010000             <1>       mov   edx, 365
000076E4 F7E2                   <1>       mul   edx
000076E6 31DB                   <1>       xor   ebx, ebx
000076E8 8A1D[8A860100]         <1>       mov   bl, [month]
000076EE FECB                   <1>       dec   bl
000076F0 D0E3                   <1>       shl   bl, 1
                                <1>       ;sub  edx, edx
000076F2 668B93[94860100]       <1>       mov   dx, [EBX+DMonth]
000076F9 8A1D[8C860100]         <1>         mov     bl, [day]
000076FF FECB                   <1>       dec   bl
00007701 01D0                   <1>       add   eax, edx
00007703 01D8                   <1>       add   eax, ebx
                                <1>                     ; EAX = days since 1/1/1970
00007705 668B15[88860100]       <1>       mov   dx, [year]
0000770C 6681EAB107             <1>       sub   dx, 1969
00007711 66D1EA                 <1>       shr   dx, 1
00007714 66D1EA                 <1>       shr   dx, 1
                                <1>             ; (year-1969)/4
00007717 01D0                   <1>       add   eax, edx
                                <1>                     ; + leap days since 1/1/1970
00007719 803D[8A860100]02       <1>       cmp   byte [month], 2     ; if past february
00007720 7610                   <1>       jna   short cte1
00007722 668B15[88860100]       <1>       mov   dx, [year]
00007729 6683E203               <1>       and   dx, 3 ; year mod 4
0000772D 7503                   <1>       jnz   short cte1
                                <1>                     ; and if leap year
0000772F 83C001                 <1>       add   eax, 1        ; add this year's leap day (february 29)
                                <1> cte1:                       ; compute seconds since 1/1/1970
00007732 BA18000000             <1>       mov   edx, 24
00007737 F7E2                   <1>       mul   edx
00007739 8A15[8E860100]         <1>       mov   dl, [hour]
0000773F 01D0                   <1>       add   eax, edx
                                <1>             ; EAX = hours since 1/1/1970 00:00:00
                                <1>       ;mov  ebx, 60
00007741 B33C                   <1>       mov   bl, 60
00007743 F7E3                   <1>       mul   ebx
00007745 8A15[90860100]         <1>       mov   dl, [minute]
```

```
0000774B 01D0                <1>       add    eax, edx
                             <1>              ; EAX = minutes since 1/1/1970 00:00:00
                             <1>       ;mov   ebx, 60
0000774D F7E3                <1>       mul    ebx
0000774F 8A15[92860100]      <1>       mov    dl, [second]
00007755 01D0                <1>       add    eax, edx
                             <1>              ; EAX -> seconds since 1/1/1970 00:00:00
00007757 C3                  <1>       retn
                             <1>
                             <1> ;set_date_time:
                             <1> convert_from_epoch:
                             <1>       ; 31/12/2017 (v2.0.0)
                             <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>       ;
00007758 31D2                <1>       xor    edx, edx
0000775A B93C000000          <1>       mov    ecx, 60
0000775F F7F1                <1>       div    ecx
                             <1>       ;mov   [imin], eax   ; whole minutes
                             <1>                            ; since 1/1/1970
00007761 668915[92860100]    <1>       mov    [second], dx  ; leftover seconds
00007768 29D2                <1>       sub    edx, edx
0000776A F7F1                <1>       div    ecx
                             <1>       ;mov   [ihrs], eax   ; whole hours
                             <1>       ;                   ; since 1/1/1970
0000776C 668915[90860100]    <1>       mov    [minute], dx  ; leftover minutes
00007773 31D2                <1>       xor    edx, edx
                             <1>       ;mov   cx, 24
00007775 B118                <1>       mov    cl, 24
00007777 F7F1                <1>       div    ecx
                             <1>       ;mov   [iday], ax   ; whole days
                             <1>                          ; since 1/1/1970
00007779 668915[8E860100]    <1>       mov    [hour], dx   ; leftover hours
00007780 05DB020000          <1>       add    eax, 365+366 ; whole day since
                             <1>                          ; 1/1/1968
                             <1>       ;mov   [iday], ax
00007785 50                  <1>       push   eax
00007786 29D2                <1>       sub    edx, edx
00007788 B9B5050000          <1>       mov    ecx, (4*365)+1 ; 4 years = 1461 days
0000778D F7F1                <1>       div    ecx
0000778F 59                  <1>       pop    ecx
                             <1>       ;mov   [lday], ax   ; count of quadyrs (4 years)
00007790 6652                <1>       push   dx
                             <1>       ;mov   [qday], dx   ; days since quadyr began
00007792 6683FA3C            <1>       cmp    dx, 31 + 29  ; if past feb 29 then
00007796 F5                  <1>       cmc                 ; add this quadyr's leap day
00007797 83D000              <1>       adc    eax, 0       ; to # of qadyrs (leap days)
                             <1>       ;mov   [lday], ax   ; since 1968
                             <1>       ;mov   cx, [iday]
0000779A 91                  <1>       xchg   ecx, eax     ; ECX = lday, EAX = iday
0000779B 29C8                <1>       sub    eax, ecx     ; iday - lday
0000779D B96D010000          <1>       mov    ecx, 365
000077A2 31D2                <1>       xor    edx, edx
                             <1>       ; EAX = iday-lday, EDX = 0
000077A4 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
000077A6 6605B007            <1>       add    ax, 1968     ; compute year
000077AA 66A3[88860100]      <1>       mov    [year], ax
000077B0 6689D1              <1>       mov    cx, dx
                             <1>       ;mov   dx, [qday]
000077B3 665A                <1>       pop    dx
000077B5 6681FA6D01          <1>       cmp    dx, 365           ; if qday <= 365 and qday >= 60
000077BA 7709                <1>       ja     short cfe1   ; jday = jday +1
000077BC 6683FA3C            <1>       cmp    dx, 60       ; if past 2/29 and leap year then
000077C0 F5                  <1>         cmc              ; add a leap day to the # of whole
000077C1 6683D100            <1>       adc    cx, 0        ; days since 1/1 of current year
                             <1> cfe1:
                             <1>       ;mov   [jday], cx
000077C5 66BB0C00            <1>       mov    bx, 12       ; estimate month
000077C9 66BA6E01            <1>       mov    dx, 366      ; mday, max. days since 1/1 is 365
000077CD 6683E003            <1>       and    ax, 11b      ; year mod 4 (and dx, 3)
                             <1> cfe2: ; Month calculation  ; 0 to 11  (11 to 0)
000077D1 6639D1              <1>       cmp    cx, dx       ; mday = # of days passed from 1/1
000077D4 731D                <1>       jnb    short cfe3
000077D6 664B                <1>       dec    bx           ; month = month - 1
000077D8 66D1E3              <1>       shl    bx, 1
000077DB 668B93[94860100]    <1>       mov    dx, [EBX+DMonth] ; # elapsed days at 1st of month
000077E2 66D1EB              <1>       shr    bx, 1        ; bx = month - 1 (0 to 11)
000077E5 6683FB01            <1>       cmp    bx, 1        ; if month > 2 and year mod 4  = 0
000077E9 76E6                <1>       jna    short cfe2   ; then mday = mday + 1
000077EB 08C0                <1>       or     al, al       ; if past 2/29 and leap year then
000077ED 75E2                <1>       jnz    short cfe2   ; add leap day (to mday)
000077EF 6642                <1>       inc    dx           ; mday = mday + 1
000077F1 EBDE                <1>       jmp    short cfe2
                             <1> cfe3:
000077F3 6643                <1>       inc    bx          ; -> bx = month, 1 to 12
000077F5 66891D[8A860100]    <1>       mov    [month], bx
000077FC 6629D1              <1>       sub    cx, dx      ; day = jday - mday + 1
000077FF 6641                <1>       inc    cx
00007801 66890D[8C860100]    <1>       mov    [day], cx
```

```
                                <1>
                                <1>       ; eax, ebx, ecx, edx is changed at return
                                <1>       ; output ->
                                <1>       ; [year], [month], [day], [hour], [minute], [second]
                                <1>
00007808 C3                     <1>       retn   ; 31/12/2017 (TRDOS 386)
                                <1>
                                <1> set_rtc_date_time:
                                <1>       ; 31/12/2017 (v2.0.0)
                                <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)
00007809 E80F000000             <1>       call   set_date_bcd
                                <1>       ; Set real-time clock date
0000780E E835F3FFFF             <1>       call   set_rtc_date ; RTC_50
                                <1>       ; Set real-time clock time
00007813 E832000000             <1>       call   set_time_bcd
00007818 E9BCF2FFFF             <1>       jmp    set_rtc_time ; RTC_30
                                <1>
                                <1> ; 31/12/2017
                                <1> set_date_bcd:
0000781D A0[89860100]           <1>         mov     al, [year+1]
00007822 D40A                   <1>       aam    ; ah = al / 10, al = al mod 10
00007824 D510                   <1>       db     0D5h,10h     ; Undocumented inst. AAD
                                <1>                           ; AL = AH * 10h + AL
00007826 88C5                   <1>       mov    ch, al ; century (BCD)
00007828 A0[88860100]           <1>       mov    al, [year]
0000782D D40A                   <1>       aam    ; ah = al / 10, al = al mod 10
0000782F D510                   <1>       db     0D5h,10h     ; Undocumented inst. AAD
                                <1>                           ; AL = AH * 10h + AL
00007831 88C1                   <1>       mov    cl, al ; year (BCD)
00007833 A0[8A860100]           <1>         mov  al, [month]
00007838 D40A                   <1>       aam    ; ah = al / 10, al = al mod 10
0000783A D510                   <1>       db     0D5h,10h     ; Undocumented inst. AAD
                                <1>                           ; AL = AH * 10h + AL
0000783C 88C6                   <1>       mov    dh, al ; month (BCD)
0000783E A0[8C860100]           <1>       mov    al, [day]
00007843 D40A                   <1>       aam    ; ah = al / 10, al = al mod 10
00007845 D510                   <1>       db     0D5h,10h     ; Undocumented inst. AAD
                                <1>                           ; AL = AH * 10h + AL
00007847 88C6                   <1>       mov    dh, al ; day (BCD)
00007849 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)
0000784A FA                     <1>       cli
0000784B E84AF4FFFF             <1>       CALL   UPD_IPR             ; CHECK FOR UPDATE IN PROCESS
                                <1>       ; cf = 1 -> al = 0
00007850 7207                   <1>         jc      short stime1
00007852 B00B                   <1>       MOV    AL,CMOS_REG_B       ; ADDRESS ALARM REGISTER
00007854 E85CF4FFFF             <1>       CALL   CMOS_READ           ; READ CURRENT VALUE OF DSE BIT
                                <1> stime1:
00007859 FB                     <1>       sti
0000785A 2401                   <1>       AND    AL,00000001B        ; MASK FOR VALID DSE BIT
0000785C 88C2                   <1>       MOV    DL,AL               ; SET [DL] TO ZERO FOR NO DSE BIT
                                <1>       ; DL = 1 or 0 (day light saving time)
                                <1>       ;
0000785E A0[8E860100]           <1>       mov    al, [hour]
00007863 D40A                   <1>       aam    ; ah = al / 10, al = al mod 10
00007865 D510                   <1>       db     0D5h,10h     ; Undocumented inst. AAD
                                <1>                           ; AL = AH * 10h + AL
00007867 88C5                   <1>       mov    ch, al ; hour (BCD)
00007869 A0[90860100]           <1>         mov     al, [minute]
0000786E D40A                   <1>       aam    ; ah = al / 10, al = al mod 10
00007870 D510                   <1>       db     0D5h,10h     ; Undocumented inst. AAD
                                <1>                           ; AL = AH * 10h + AL
00007872 88C1                   <1>       mov    cl, al       ; minute (BCD)
00007874 A0[92860100]           <1>         mov     al, [second]
00007879 D40A                   <1>       aam    ; ah = al / 10, al = al mod 10
0000787B D510                   <1>       db     0D5h,10h     ; Undocumented inst. AAD
                                <1>                           ; AL = AH * 10h + AL
0000787D 88C6                   <1>       mov    dh, al      ; second (BCD)
0000787F C3                     <1>       retn   ; 30/12/2017
                                    %include 'trdosk2.s' ; 04/01/2016
                                <1> ; ****************************************************************************
                                <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.2) - DRV INIT : trdosk2.s
                                <1> ; ----------------------------------------------------------------------------
                                <1> ; Last Update: 30/08/2020
                                <1> ; ----------------------------------------------------------------------------
                                <1> ; Beginning: 04/01/2016
                                <1> ; ----------------------------------------------------------------------------
                                <1> ; Assembler: NASM version 2.14 (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>       ; 30/08/2020
                                <1>       ; 25/08/2020
                                <1>       ; 11/08/2020, 13/08/2020
                                <1>       ; 17/07/2020, 20/07/2020
                                <1>       ; 14/07/2020, 15/07/2020
                                <1>       ; 30/01/2018
                                <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
                        <1>
                        <1>       ; 15/07/2020
                        <1>       ;movzx ecx, byte [HF_NUM] ; number of fixed disks
                        <1>       ;cmp   cl, 1
                        <1>       ;jnb   short load_hd_partition_tables
                        <1>
00007880 A0[5C8A0100]   <1>       mov    al, [HF_NUM] ; number of fixed disks
00007885 20C0           <1>       and    al, al
00007887 7501           <1>       jnz    short load_hd_partition_tables
                        <1>
                        <1>       ; no any hard disks
00007889 C3             <1>       retn
                        <1>
                        <1> load_hd_partition_tables:
                        <1>       ;mov   esi, [HDPM_TBL_VEC] ; primary master disk FDPT
                        <1>       ; 15/07/2020
0000788A BE[608A0100]   <1>       mov    esi, HDPM_TBL_VEC
0000788F BF[868E0100]   <1>       mov    edi, PTable_hd0
00007894 B280           <1>       mov    dl, 80h
                        <1>       ; 15/07/2020
00007896 A2[FD6D0000]   <1>       mov    [hdc], al
                        <1>       ;xor   ecx, ecx ; 0
                        <1> load_next_hd_partition_table:
                        <1>       ; 20/07/2020
0000789B 31C9           <1>       xor    ecx, ecx ; 0
                        <1>       ;push  ecx
0000789D 57             <1>       push   edi ; *
                        <1>       ;push  esi ; FDPT (+ DPTE) address
                        <1>       ; 15/07/2020
0000789E AD             <1>       lodsd
0000789F 56             <1>       push   esi ; ** ; next FDPT (+ DPTE) address ptr
                        <1>
                        <1>       ;mov   al, [esi+20] ; DPTE offset 4
                        <1>       ;and   al, 40h ;  LBA bit (bit 6)
                        <1>       ;;shr  al, 6
                        <1>       ;mov   [HD_LBA_yes], al
                        <1>
                        <1>       ; 15/07/2020
000078A0 8A4814         <1>       mov    cl, [eax+20]
000078A3 80E140         <1>       and    cl, 40h
000078A6 880D[8A8F0100] <1>       mov    [HD_LBA_yes], cl
                        <1>
000078AC E844040000     <1>       call   load_masterboot
                        <1>       ;jc    short pass_pt_this_hard_disk
                        <1>       ; 13/08/2020
000078B1 0F828A000000   <1>       jc     pass_pt_this_hard_disk
                        <1>
000078B7 BB[448E0100]   <1>       mov    ebx, PartitionTable
000078BC 89DE           <1>       mov    esi, ebx
                        <1>       ;mov   ecx, 16
000078BE B110           <1>       mov    cl, 16
000078C0 F3A5           <1>       rep    movsd
000078C2 89DE           <1>       mov    esi, ebx
                        <1>       ;mov   byte [hdc], 4 ; 4 - partition index
                        <1>       ; 15/07/2020
000078C4 C605[8B8F0100]04 <1>     mov    byte [PP_Counter], 4
                        <1> loc_validate_hdp_partition:
                        <1>       ;cmp   byte [esi+ptFileSystemID], 0
                        <1>       ;jna   short loc_validate_next_hdp_partition2
                        <1>       ; 13/08/2020
000078CB 8A4604         <1>       mov    al, [esi+ptFileSystemID]
000078CE 20C0           <1>       and    al, al
000078D0 7457           <1>       jz     short loc_validate_next_hdp_partition2
                        <1>
000078D2 56             <1>       push   esi ; *** ; Masterboot partition table offset
000078D3 52             <1>       push   edx ; **** ; dl = Physical drive number
                        <1>
                        <1>       ; 13/08/2020
000078D4 3C05           <1>       cmp    al, 05h  ; Extended partition CHS
000078D6 7404           <1>       je     short loc_set_ep_counter
000078D8 3C0F           <1>       cmp    al, 0Fh  ; Extended partition LBA
000078DA 7511           <1>       jne    short loc_validate_next_hdp_partition0
                        <1>
                        <1>       ;;inc  byte [PP_Counter]
                        <1>       ; 15/07/2020
                        <1>       ;inc   byte [EP_Counter] ; disk has valid partition(s)
                        <1>
                        <1> loc_set_ep_counter:
                        <1>       ; 13/08/2020
000078DC 803D[8C8F0100]80 <1>     cmp    byte [EP_Counter], 80h
000078E3 7342           <1>       jnb    short loc_validate_next_hdp_partition1
                        <1>
000078E5 8815[8C8F0100] <1>       mov    byte [EP_Counter], dl ; disk drv has extd. part.
                        <1>
000078EB EB3A           <1>       jmp    short loc_validate_next_hdp_partition1
                        <1>
                        <1> loc_validate_next_hdp_partition0:
000078ED 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
000078EF E885010000     <1>       call   validate_hd_fat_partition
000078F4 730E           <1>       jnc    short loc_set_valid_hdp_partition_entry
                        <1>
                        <1>       ;pop   edx
                        <1>       ;push  edx
000078F6 8B1424         <1>       mov    edx, [esp]  ; ****
000078F9 8B742404       <1>       mov    esi, [esp+4] ; *** ; 30/01/2018
000078FD E8D1020000     <1>       call   validate_hd_fs_partition
00007902 7223           <1>       jc     short loc_validate_next_hdp_partition1
                        <1> loc_set_valid_hdp_partition_entry:
00007904 8A0D[50400100] <1>       mov    cl, [Last_DOS_DiskNo]
0000790A 80C141         <1>       add    cl, 'A'
```

```
                                <1>       ; ESI = Logical dos drive description table address
0000790D 880E                   <1>       mov   [esi+LD_Name], cl
                                <1>       ; 15/07/2020
0000790F 8A4602                 <1>       mov   al, [esi+LD_PhyDrvNo] ; Physical drive number
                                <1>       ;mov  al, [esp] ; ****
00007912 2C7F                   <1>       sub   al, 7Fh
                                <1>             ; AL = 1 to 4
00007914 C0E002                 <1>       shl   al, 2 ; AL = 4 to 16
                                <1>
00007917 8A15[8B8F0100]         <1>       mov   dl, [PP_Counter]
                                <1>
                                <1>       ;sub  al, [PP_Counter]
0000791D 28D0                   <1>       sub   al, dl ; [PP_Counter] ; 4 - partition index
                                <1>
                                <1>       ; AL = Partition entry/index, 0 based
                                <1>       ;  0 -> hd 0, Partition Table offset = 0
                                <1>       ; 15 -> hd 3, Partition Table offset = 3
                                <1>
                                <1>       ;mov  [esi+LD_PartitionEntry], al
                                <1>
                                <1>       ; 15/07/2020
0000791F B404                   <1>       mov   ah, 4
                                <1>       ;sub  ah, [PP_Counter]
00007921 28D4                   <1>       sub   ah, dl
                                <1>
                                <1>       ; AH = Primary partition index, 0 to 3 ; pt entry
                                <1>       ;            (4 to 7 for logical disk partitions)
                                <1>
                                <1>       ;mov  [esi+LD_DParamEntry], ah
00007923 6689467C               <1>       mov   [esi+LD_PartitionEntry], ax
                                <1>
                                <1> loc_validate_next_hdp_partition1:
00007927 5A                     <1>       pop   edx ; **** ; dl = Physical drive number
00007928 5E                     <1>       pop   esi ; *** ; Masterboot partition table offset
                                <1>
                                <1> loc_validate_next_hdp_partition2:
                                <1>       ; ESI = PartitionTable offset
                                <1>       ; DL = Hard/Fixed disk drive number
                                <1>
                                <1>       ;dec  byte [hdc] ; 4 - partition index
                                <1>       ;jz   short pass_pt_this_hard_disk
                                <1>       ; 15/07/2020
00007929 FE0D[8B8F0100]         <1>       dec   byte [PP_Counter] ; 4 - partition index
0000792F 7410                   <1>       jz    short pass_pt_this_hard_disk
                                <1>
00007931 83C610                 <1>       add   esi, 16 ; 10h
00007934 EB95                   <1>       jmp   short loc_validate_hdp_partition
                                <1>
                                <1> loc_not_any_extd_partitions:
                                <1>       ; 15/07/2020
00007936 C3                     <1>       retn
                                <1>
                                <1> loc_next_hd_partition_table:
00007937 FEC2                   <1>       inc   dl
                                <1>       ; 15/07/2020
                                <1>       ;add  esi, 32 ; next FDPT address
00007939 83C740                 <1>       add   edi, 64 ; next partition table destination
0000793C E95AFFFFFF             <1>         jmp     load_next_hd_partition_table
                                <1>
                                <1> pass_pt_this_hard_disk:
                                <1>       ;pop  esi ; FDPT (+ DPTE) address
                                <1>       ; 15/07/2020
00007941 5E                     <1>       pop   esi ; ** ; next FDPT (+ DPTE) address ptr
00007942 5F                     <1>       pop   edi ; * ; Ptable_hd?
                                <1>       ;pop  ecx
                                <1>       ;loop loc_next_hd_partition_table
00007943 FE0D[FD6D0000]         <1>       dec   byte [hdc]
00007949 75EC                   <1>       jnz   short loc_next_hd_partition_table
                                <1>
                                <1>       ;cmp  byte [PP_Counter], 1
                                <1>       ;jnb  short load_extended_dos_partitions
                                <1>       ;; Empty partition table
                                <1>       ;retn
                                <1>
                                <1>       ; 11/08/2020
                                <1>       ; 17/07/2020
                                <1> check_extended_partitions:
                                <1>       ; 15/07/2020
                                <1>       ;cmp  byte [EP_Counter], 0
                                <1>       ;jna  short loc_not_any_extd_partitions
                                <1>       ; 13/08/2020
0000794B A0[8C8F0100]           <1>       mov   al, [EP_Counter] ; 1st disk drv has extd partition
00007950 08C0                   <1>       or    al, al ; 0 ?
00007952 74E2                   <1>       jz    short loc_not_any_extd_partitions
                                <1>
                                <1> load_extended_dos_partitions:
                                <1>       ;mov  byte [hdc], 80h
                                <1>       ; 13/08/2020
00007954 A2[FD6D0000]           <1>       mov   byte [hdc], al ; 1st disk drv has extd partition
                                <1>       ; 25/08/2020
00007959 2C80                   <1>       sub   al, 80h
0000795B 740E                   <1>       jz    short loc_set_ext_ptable_hd0
0000795D C0E006                 <1>       shl   al, 6 ; * 64
00007960 0FB6F0                 <1>       movzx esi, al
00007963 81C6[868E0100]         <1>       add   esi, PTable_hd0
00007969 EB05                   <1>       jmp   short next_hd_extd_partition
                                <1>
                                <1>       ; 25/08/2020
                                <1> loc_set_ext_ptable_hd0:
0000796B BE[868E0100]           <1>       mov   esi, PTable_hd0
                                <1>
                                <1> next_hd_extd_partition:
                                <1>       ; 17/07/2020
                                <1>       ;mov  byte [EP_Counter], 0 ; Reset for each physical disk
                                <1>       ; 13/08/2020
```

```
                                <1>       ;mov   byte [LD_Counter], 0 ; Reset logical drive index
00007970 66C705[8C8F0100]00-    <1>       mov    word [EP_Counter], 0 ; Reset EP index and LD index
00007978 00                     <1>
                                <1>
00007979 56                     <1>       push   esi ; **** ; PTable_hd? offset
                                <1>
0000797A C605[8B8F0100]04       <1>       mov    byte [PP_Counter], 4
                                <1>                           ; set for each extd partition table
                                <1>       ;;mov  ecx, 4
                                <1>       ;mov   cl, 4
00007981 8A15[FD6D0000]         <1>       mov    dl, [hdc]
                                <1> hd_check_fs_id_05h:
00007987 8A4604                 <1>       mov    al, [esi+ptFileSystemID]
0000798A 3C05                   <1>       cmp    al, 05h ; Is it an extended dos partition ?
0000798C 7411                   <1>       je     short loc_set_ep_start_sector ; yes
                                <1> hd_check_fs_id_0Fh:
0000798E 3C0F                   <1>       cmp    al, 0Fh ; Is it an extended win4 (LBA mode) partition ?
00007990 740D                   <1>       je     short loc_set_ep_start_sector ; yes
                                <1>
                                <1> continue_to_check_ep:
                                <1>       ;add   esi, 16
                                <1>       ;loop  hd_check_fs_id_05h
                                <1>       ; 15/07/2020
                                <1>       ;dec   cl
                                <1>       ;jz    short continue_check_ep_next_disk
00007992 FE0D[8B8F0100]         <1>       dec    byte [PP_Counter] ; 4 --> 0
00007998 742D                   <1>       jz     short continue_check_ep_next_disk
0000799A 83C610                 <1>       add    esi, 16
0000799D EBE8                   <1>       jmp    short hd_check_fs_id_05h
                                <1>
                                <1> loc_set_ep_start_sector:
                                <1>       ; dl = [hdc] ; Drive number
                                <1>       ; 15/07/2020
0000799F 8B4E08                 <1>       mov    ecx, [esi+ptStartSector]
                                <1>       ; 30/08/2020
000079A2 890D[8E8F0100]         <1>       mov    [MBR_EP_StartSector], ecx
                                <1>       ; 20/07/2020
                                <1> loc_validate_hde_partition_next:
000079A8 890D[928F0100]         <1>       mov    [EP_StartSector], ecx ; Extended partition's start sector
000079AE BB[868C0100]           <1>         mov  ebx, MasterBootBuff
000079B3 803D[8A8F0100]01       <1>       cmp    byte [HD_LBA_yes], 1 ; LBA ready = Yes
000079BA 7227                   <1>       jb     short loc_hd_load_ep_05h ; cf = 1 ; 20/07/2020
                                <1>       ; 11/08/2020
                                <1>       ; (BugFix for extended partition type 05h beyond CHS limit)
                                <1>       ; (Infact if extended partition starts at the beyond of CHS limit,
                                <1>       ;  it's partition ID must be 0Fh but they/somebodies had used 05h.)
                                <1>       ;cmp   al, 05h
                                <1>       ;je    short loc_hd_load_ep_05h
                                <1> loc_hd_load_ep_0Fh:
                                <1>       ; 04/01/2016
                                <1>       ;push  ecx
                                <1>       ; 15/07/2020
                                <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)
                                <1>       ;mov   ah, 1Bh ; LBA read
                                <1>       ;mov   al, 1 ; sector count
000079BC 66B8011B               <1>       mov    ax, 1B01h
000079C0 E864D8FFFF             <1>       call   int13h
                                <1>       ;pop   ecx
                                <1>       ;jnc   short loc_hd_move_ep_table
                                <1>       ; 15/07/2020
000079C5 732E                   <1>       jnc    short loc_validate_hde_partition
                                <1>
                                <1> continue_check_ep_next_disk:
                                <1>       ; 15/07/2020
                                <1>       ;pop   edi ; PTable_ep?
000079C7 5E                     <1>       pop    esi ; **** ; PTable_hd?
000079C8 A0[5C8A0100]           <1>       mov    al, [HF_NUM] ; number of hard disks
000079CD 047F                   <1>       add    al, 7Fh
000079CF 3805[FD6D0000]         <1>       cmp    [hdc], al
000079D5 730B                   <1>       jnb    short loc_validating_hd_partitions_ok
000079D7 83C640                 <1>       add    esi, 64
                                <1>       ; 15/07/2020
                                <1>       ;add   edi, 64
000079DA FE05[FD6D0000]         <1>       inc    byte [hdc]
000079E0 EB8E                   <1>       jmp    short next_hd_extd_partition
                                <1>
                                <1> loc_validating_hd_partitions_ok:
                                <1>       ; 15/07/2020
                                <1>       ;mov   al, [Last_DOS_DiskNo]
                                <1> loc_drv_init_retn:
000079E2 C3                     <1>       retn
                                <1>
                                <1> loc_hd_load_ep_05h:
                                <1>       ; 20/07/2020 ('diskio.s', int13h, cf = 1 -> bugfix)
                                <1>       ;clc    ; (Bug: int13h would not clear carry flag bit,
                                <1>       ;      ; even if there would not be an error)
                                <1>       ;      ; ((Fix: now, int13h procedure clears carry flag
                                <1>       ;      ;  at the entrance of it.. 20/07/2020))
                                <1>       ; 15/07/2020
                                <1>       ;push  ecx
000079E3 8A7601                 <1>       mov    dh, [esi+ptBeginHead]
000079E6 668B4E02               <1>         mov     cx, [esi+ptBeginSector]
000079EA 66B80102               <1>       mov    ax, 0201h ; Read 1 sector
                                <1>       ;mov   ebx, MasterBootBuff
000079EE E836D8FFFF             <1>       call   int13h ; 20/07/2020
                                <1>                     ; 'diskio.s' modification, 'clc'
                                <1>       ;pop   ecx
000079F3 72D2                   <1>       jc     short continue_check_ep_next_disk
                                <1>       ; 15/07/2020
                                <1>       ;jmp   short loc_validate_hde_partition
                                <1>
```

```
                            <1>       ; 15/07/2020
                            <1> ;loc_hd_move_ep_table:
                            <1>       ;;pop  edi
                            <1>       ;;push edi  ; PTable_ep?
                            <1>       ;mov   edi, [esp]
                            <1>         ;mov esi, PartitionTable ; Extended
                            <1>       ;mov   ebx, esi
                            <1>       ;;mov  ecx, 16
                            <1>       ;mov   cl, 16
                            <1>              ;rep   movsd
                            <1>       ;mov   esi, ebx
                            <1> ;loc_set_hde_sub_partition_count:
                            <1>       ;mov   byte [PP_Counter], 4
                            <1>       ;mov   byte [EP_Counter], 0
                            <1>
                            <1> loc_validate_hde_partition:
                            <1>       ; 13/08/2020
                            <1>       ; 15/07/2020
                            <1>       ;mov   byte [PP_Counter], 4
000079F5 BE[448E0100]       <1>       mov    esi, PartitionTable ; (in MasterBootBuff)
                            <1>       ; 13/08/2020
                            <1>       ;jmp   short get_minidisk_partition_entry
                            <1>
                            <1> ;get_minidisk_partition_entry:
                            <1> ;     ; 20/07/2020
                            <1> ;     cmp    byte [esi+ptFileSystemID], 0
                            <1> ;     ja     short loc_validate_minidisk_partition
                            <1> ;     ; 13/08/2020
                            <1> ;     jmp    short continue_check_ep_next_disk
                            <1>
                            <1> ;     ; 11/08/2020
                            <1> ;get_minidisk_partition_entry_next:
                            <1> ;     ; 13/08/2020
                            <1> ;     cmp    esi, PartitionTable+64
                            <1> ;     jnb    short continue_check_ep_next_disk
                            <1> ;
                            <1> ;     add    esi, 16 ; 10h
                            <1> ;     ;jmp   short get_minidisk_partition_entry
                            <1>
                            <1>       ; 13/08/2020
                            <1> get_minidisk_partition_entry:
                            <1>       ; 20/07/2020
000079FA 807E0400           <1>       cmp    byte [esi+ptFileSystemID], 0
000079FE 76C7               <1>       jna    short continue_check_ep_next_disk ; 13/08/2020
                            <1>
                            <1> loc_validate_minidisk_partition:
                            <1>       ; 13/08/2020
                            <1>       ; 20/07/2020
                            <1>       ;push  esi ; *** ; Extended partition table offset
                            <1>
                            <1>       ; 13/08/2020
00007A00 FE05[8C8F0100]     <1>       inc    byte [EP_Counter] ; current (sub partition) index
                            <1>                                ; in current extended partition
                            <1>
00007A06 BF[928F0100]       <1>       mov    edi, EP_StartSector
                            <1>
                            <1>       ; Input -> ESI = PartitionTable offset
                            <1>       ; DL = Hard disk drive number
                            <1>       ; EDI = Extended partition start sector pointer
00007A0B E869000000         <1>       call   validate_hd_fat_partition
                            <1>       ;pop   ecx ; *
00007A10 7308               <1>       jnc    short loc_set_valid_hde_partition_entry
                            <1>                      ; jump down to deep !!!
                            <1>
                            <1>       ;pop   esi ; *** ; Extended partition table offset
                            <1>       ; 13/08/2020
                            <1>       ;mov   esi, PartitionTable
                            <1>
                            <1>       ; 11/08/2020
                            <1>       ; ESI = Extended partition table offset
00007A12 8A15[FD6D0000]     <1>       mov    dl, [hdc]
                            <1>
                            <1>       ;; DL = Hard disk drive number
                            <1>       ;dec   byte [PP_Counter]
                            <1>       ;jz    short continue_check_ep_next_disk
                            <1>       ;add   esi, 16 ; 10h
                            <1>       ;mov   dl, [hdc]
                            <1>       ;jmp   short get_minidisk_partition_entry
                            <1>
                            <1>       ; 11/08/2020
                            <1>       ;jmp   short get_minidisk_partition_entry_next
                            <1>
                            <1>       ; 23/08/2020
00007A18 EB3D               <1>       jmp    short validate_next_minidisk_partition_ok
                            <1>
                            <1>       ; 17/07/2020
                            <1>       ;; jumping down to deep levels !!!
                            <1>       ; ((That is a pitty microsoft preferred ep table chain
                            <1>       ; instead of a single table as mbr partition table!?))
                            <1>
                            <1> loc_set_valid_hde_partition_entry:
                            <1>       ; 15/07/2020
00007A1A A0[FD6D0000]       <1>       mov    al, [hdc] ; Hard disk drive number (>=80h)
00007A1F 88C2               <1>       mov    dl, al ; mov dl, [hdc]
00007A21 2C7F               <1>       sub    al, 7Fh
                            <1>                ; 1 to 4
00007A23 C0E002             <1>       shl    al, 2 ; 4 to 16
00007A26 2A05[8B8F0100]     <1>       sub    al, [PP_Counter] ; al - (4 - partition index)
                            <1>                  ; (disk number * 4) + partition index
                            <1>
```

```
                                <1>       ; AL = Partition entry/index, 0 based
                                <1>         ;  0 -> hd 0, Partition Table offset = 0
                                <1>         ; 15 -> hd 3, Partition Table offset = 3
                                <1>
                                <1>       ;mov   ah, 4 ; Logical dos partition (>= 4)
                                <1>       ;add   ah, [EP_Counter]
                                <1>              ; Logical disk partition index = 4 to 7
                                <1>              ; (Primary disk partition index = 0 to 3)
                                <1>
                                <1>       ; 13/08/2020
00007A2C 8A25[8D8F0100]         <1>       mov    ah, [LD_Counter] ; Logical drive index number
                                <1>                               ; (in current extended partition)
00007A32 80C404                 <1>       add    ah, 4 ; 4 to 7
                                <1>
                                <1>       ; 15/07/2020
                                <1>       ; CX -> AX
                                <1>       ;; 06/01/2016 (TRDOS v2.0)
                                <1>       ;; BUGFIX *
                                <1>       ;;mov  [esi+LD_PartitionEntry], cl
                                <1>       ;;mov  [esi+LD_DParamEntry], ch
                                <1>       ;mov   [esi+LD_PartitionEntry], cx
00007A35 6689467C               <1>       mov    [esi+LD_PartitionEntry], ax
                                <1>
00007A39 8A0D[50400100]         <1>       mov    cl, [Last_DOS_DiskNo]
00007A3F 80C141                 <1>       add    cl, 'A'
00007A42 880E                   <1>       mov    [esi+LD_Name], cl
                                <1>
                                <1>       ; 17/07/2020
                                <1>       ;cmp   cl, 'Z'
                                <1>       ;jb    short logical_drive_count_ok_for_next
                                <1>       ;pop   esi ; ***
                                <1>       ;pop   esi ; ****
                                <1>       ;retn
                                <1>
                                <1> ;logical_drive_count_ok_for_next:
                                <1>
                                <1>       ;; 15/07/2020
                                <1>       ;inc   byte [EP_Counter]
                                <1>       ; 13/08/2020
00007A44 FE05[8D8F0100]         <1>       inc    byte [LD_Counter]
                                <1>
                                <1>       ;mov   dl, [hdc]
                                <1>
                                <1>       ; 17/07/2020
                                <1>       ;; Now,
                                <1>       ;; we are swimming in deep of an extended partition !!!
                                <1>       ; (! sub or chained extended partition tables !)
                                <1>       ; ((Logical dos partitions in extended partition were called
                                <1>       ;  as 'mini disk partition' in msdos 6.0 source code.))
                                <1>
                                <1> validate_next_minidisk_partition:
                                <1>       ; 13/08/2020
                                <1>       ;pop   esi ; *** ; Extended partition table offset
                                <1>
                                <1>       ; 17/07/2020
                                <1>       ;cmp   byte [EP_Counter], 4
                                <1>       ; 13/08/2020
00007A4A 803D[8D8F0100]04       <1>       cmp    byte [LD_Counter], 4 ; maximum 4 logical disks
                                <1>                                   ; per extended partition
00007A51 0F8370FFFFFF           <1>       jnb    continue_check_ep_next_disk
                                <1>
                                <1> validate_next_minidisk_partition_ok:
                                <1>       ; 13/08/2020
                                <1>       ;dec   byte [PP_Counter] ; 4 --> 0
                                <1>       ;jz    continue_check_ep_next_disk
                                <1>
                                <1>       ;cmp   esi, PartitionTable+64
                                <1>       ;jnb   continue_check_ep_next_disk
                                <1>
                                <1>       ;add   esi, 16
                                <1>       ; 13/08/2020
00007A57 BE[548E0100]           <1>       mov    esi, PartitionTable+16
                                <1>
                                <1>       ; 20/07/2020
00007A5C 8A4604                 <1>       mov    al, [esi+ptFileSystemID]
                                <1>
                                <1>       ; 20/07/2020
00007A5F 3C05                   <1>       cmp    al, 05h ; Is it an extended dos partition ?
00007A61 7408                   <1>       je     short loc_minidisk_next_ep_lba_chs ; 17/07/2020
00007A63 3C0F                   <1>       cmp    al, 0Fh ; Is it an extended win4 (LBA mode) partition ?
00007A65 0F855CFFFFFF           <1>       jne    continue_check_ep_next_disk ; AL must be 0 here
                                <1>                                        ; (when it is not 05h or 0Fh)
                                <1>                                        ; If AL is not ZERO -> EP Bug!
                                <1>                                        ; (!Microsoft DOS convention!)
                                <1> loc_minidisk_next_ep_lba_chs:
                                <1>       ; 17/07/2020
00007A6B 8B4E08                 <1>       mov    ecx, [esi+ptStartSector] ; relative start sector number
                                <1>       ;add   ecx, [EP_StartSector]
                                <1>       ; 30/08/2020
00007A6E 030D[8E8F0100]         <1>       add    ecx, [MBR_EP_StartSector]
                                <1>       ; 20/07/2020
00007A74 E92FFFFFFF             <1>       jmp    loc_validate_hde_partition_next
                                <1>
                                <1> validate_hd_fat_partition:
                                <1>       ; 17/07/2020
                                <1>       ; 15/07/2020
                                <1>       ;      (optimization)
                                <1>       ; 14/07/2020
                                <1>       ;      (fat16 -big- partition search bugfix)
                                <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
00007A79 8A6604         <1>       mov    ah, [esi+ptFileSystemID]
00007A7C B002           <1>       mov    al, 2 ; 27/12/2017
00007A7E 80FC06         <1>       cmp    ah, 06h ; FAT16 CHS partition (>=32MB)
                        <1>       ; 12/02/2016
                        <1>       ;;jb   short loc_not_a_valid_fat_partition2
                        <1>       ;jnb   short vhdp_FAT16_32
                        <1>       ; 14/07/2020 (BugFix)
00007A81 7711           <1>       ja     short vhdp_FAT16_32
00007A83 7425           <1>       je     short loc_set_valid_hd_partition_params
                        <1>
                        <1> vhdp_FAT12_16:
                        <1>       ; 27/12/2017
00007A85 FEC8           <1>       dec    al ; mov al, 1
00007A87 38C4           <1>       cmp    ah, al ; 1 ; FAT12 partition
00007A89 741F           <1>       je     short loc_set_valid_hd_partition_params
                        <1>       ;
00007A8B FEC0           <1>       inc    al ; mov al, 2
00007A8D 80FC04         <1>       cmp    ah, 04h ; FAT16 CHS partition (< 32MB)
00007A90 7418           <1>       je     short loc_set_valid_hd_partition_params
                        <1>
                        <1>       ; 15/07/2020
                        <1>       ; (ah = 05h, 02h or 03h)
                        <1> loc_not_a_valid_fat_partition1:
00007A92 F9             <1>       stc
                        <1>       ; cf=1
00007A93 C3             <1>       retn
                        <1>
                        <1> vhdp_FAT16_32:
                        <1>       ; 15/07/2020
                        <1>       ;mov   al, 3
00007A94 FEC0           <1>       inc    al
00007A96 80FC0C         <1>       cmp    ah, 0Ch ; FAT32 LBA partition
00007A99 740F           <1>       je     short loc_set_valid_hd_partition_params
00007A9B 7706           <1>       ja     short vhdp_check_FAT16_lba
                        <1>
                        <1> vhdp_check_FAT32_chs:
00007A9D 80FC0B         <1>       cmp    ah, 0Bh ; FAT32 CHS partition
00007AA0 7408           <1>       je     short loc_set_valid_hd_partition_params
                        <1>       ;jne   short loc_not_a_valid_fat_partition1
                        <1>
                        <1>       ;stc
                        <1> loc_not_a_valid_fat_partition2:
00007AA2 C3             <1>       retn
                        <1>
                        <1> vhdp_check_FAT16_lba:
00007AA3 80FC0E         <1>       cmp    ah, 0Eh ; FAT16 LBA partition
00007AA6 75EA           <1>       jne    short loc_not_a_valid_fat_partition1
                        <1>
                        <1>       ;mov   al, 2
00007AA8 FEC8           <1>       dec    al
                        <1>
                        <1> loc_set_valid_hd_partition_params:
                        <1>       ; 15/07/2020
                        <1>       ;inc   byte [Last_DOS_DiskNo] ; > 1
                        <1>       ;
00007AAA 31DB           <1>       xor    ebx, ebx
00007AAC 8A3D[50400100] <1>       mov    bh, [Last_DOS_DiskNo] ; * 256
00007AB2 FEC7           <1>       inc    bh ; 15/07/2020
00007AB4 81C300010900   <1>       add    ebx, Logical_DOSDisks
                        <1>       ;
00007ABA C6430102       <1>       mov    byte [ebx+LD_DiskType], 2
00007ABE 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
00007AC1 66894303       <1>       mov    word [ebx+LD_FATType], ax
                        <1>       ;
00007AC5 8B4E08         <1>       mov    ecx, [esi+ptStartSector]
00007AC8 09FF           <1>       or     edi, edi
00007ACA 7402           <1>       jz     short pass_hd_FAT_ep_start_sector_adding
                        <1> loc_add_hd_FAT_ep_start_sector:
                        <1>       ; 17/07/2020
00007ACC 030F           <1>       add    ecx, [edi]
                        <1> pass_hd_FAT_ep_start_sector_adding:
00007ACE 894B6C         <1>       mov    [ebx+LD_StartSector], ecx
                        <1> loc_hd_FAT_logical_drv_init:
00007AD1 89DD           <1>       mov    ebp, ebx
                        <1>       ;mov   dl, [ebx+LD_PhyDrvNo]
00007AD3 A0[8A8F0100]   <1>       mov    al, [HD_LBA_yes] ; 07/01/2016
00007AD8 884305         <1>       mov    [ebx+LD_LBAYes], al
00007ADB BB[968F0100]   <1>       mov    ebx, DOSBootSectorBuff ; buffer address
00007AE0 08C0           <1>       or     al, al
00007AE2 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)
00007AE4 B41B           <1>       mov    ah, 1Bh ; LBA read
00007AE6 B001           <1>       mov    al, 1 ; sector count
```

```
659 00007AE8 E83CD7FFFF          <1>       call   int13h
660 00007AED 7313                <1>       jnc    short loc_hd_drv_FAT_boot_validation
661                              <1> loc_not_a_valid_fat_partition3:
662 00007AEF C3                  <1>       retn
663                              <1> loc_hd_FAT_drv_init_load_bs_chs:
664 00007AF0 8A7601              <1>       mov    dh, [esi+ptBeginHead]
665 00007AF3 668B4E02            <1>       mov    cx, [esi+ptBeginSector]
666 00007AF7 66B80102            <1>       mov    ax, 0201h ; Read 1 sector
667                              <1>       ;mov   ebx, DOSBootSectorBuff
668 00007AFB E829D7FFFF          <1>       call   int13h
669 00007B00 72ED                <1>       jc     short loc_not_a_valid_fat_partition3
670                              <1> loc_hd_drv_FAT_boot_validation:
671                              <1>       ;mov   esi, DOSBootSectorBuff
672 00007B02 89DE                <1>       mov    esi, ebx
673 00007B04 6681BEFE01000055AA  <1>       cmp    word [esi+BS_Validation], 0AA55h
674 00007B0D 7512                <1>       jne    short loc_not_a_valid_fat_partition4
675 00007B0F 807E15F8            <1>       cmp    byte [esi+BPB_Media], 0F8h
676 00007B13 750C                <1>       jne    short loc_not_a_valid_fat_partition4
677                              <1>
678                              <1>       ; 27/12/2017
679 00007B15 807D0303            <1>       cmp    byte [ebp+LD_FATType], 3
680 00007B19 7508                <1>       jne    short loc_hd_FAT16_BPB
681                              <1>
682                              <1> loc_hd_drv_FAT32_boot_validation:
683 00007B1B 807E4229            <1>       cmp    byte [esi+BS_FAT32_BootSig], 29h
684 00007B1F 7416                <1>       je     short loc_hd_FAT32_BPB
685                              <1>
686                              <1> loc_not_a_valid_fat_partition4:
687 00007B21 F9                  <1>       stc
688 00007B22 C3                  <1>       retn
689                              <1>
690                              <1> loc_hd_FAT16_BPB:
691 00007B23 807E2629            <1>       cmp    byte [esi+BS_BootSig], 29h
692 00007B27 75F8                <1>       jne    short loc_not_a_valid_fat_partition4
693                              <1>
694 00007B29 66837E1600          <1>       cmp    word [esi+BPB_FATSz16], 0
695 00007B2E 7607                <1>       jna    short loc_hd_big_FAT16_BPB
696 00007B30 B920000000          <1>       mov    ecx, 32
697 00007B35 EB05                <1>       jmp    short loc_hd_move_FAT_BPB
698                              <1>
699                              <1> loc_hd_FAT32_BPB:
700                              <1>       ;cmp   word [esi+BPB_FATSz16], 0
701                              <1>       ;ja    short loc_not_a_valid_fat_partition4
702                              <1> loc_hd_big_FAT16_BPB:
703 00007B37 B92D000000          <1>       mov    ecx, 45
704                              <1>
705                              <1> loc_hd_move_FAT_BPB:
706 00007B3C 89EF                <1>       mov    edi, ebp
707                              <1>       ;mov   esi, ebx ; Boot sector
708 00007B3E 57                  <1>       push   edi
709 00007B3F 83C706              <1>       add    edi, LD_BPB
710 00007B42 F366A5              <1>       rep    movsw
711 00007B45 5E                  <1>       pop    esi
712 00007B46 0FB74614            <1>       movzx  eax, word [esi+LD_BPB+BPB_RsvdSecCnt]
713 00007B4A 03466C              <1>       add    eax, [esi+LD_StartSector]
714 00007B4D 894660              <1>       mov    [esi+LD_FATBegin], eax
715 00007B50 807E0303            <1>       cmp    byte [esi+LD_FATType], 3
716 00007B54 7224                <1>       jb     short loc_set_FAT16_RootDirLoc
717                              <1> loc_set_FAT32_RootDirLoc:
718 00007B56 8B462A              <1>       mov    eax, [esi+LD_BPB+BPB_FATSz32]
719 00007B59 0FB65E16            <1>         movzx      ebx, byte [esi+LD_BPB+BPB_NumFATs]
720 00007B5D F7E3                <1>       mul    ebx
721 00007B5F 034660              <1>       add    eax, [esi+LD_FATBegin]
722                              <1> loc_set_FAT32_data_begin:
723 00007B62 894668              <1>       mov    [esi+LD_DATABegin], eax
724 00007B65 894664              <1>       mov    [esi+LD_ROOTBegin], eax
725                              <1>       ; If Root Directory Cluster <> 2 then
726                              <1>       ; change the beginning sector value
727                              <1>       ; of the root dir by adding sector offset.
728 00007B68 8B4632              <1>       mov    eax, [esi+LD_BPB+BPB_RootClus]
729 00007B6B 83E802              <1>       sub    eax, 2
730 00007B6E 7435                <1>       jz     short short loc_set_32bit_FAT_total_sectors
731                              <1>       ;movzx ebx, byte [esi+LD_BPB+BPB_SecPerClust]
732 00007B70 8A5E13              <1>       mov    bl, [esi+LD_BPB+BPB_SecPerClust]
733 00007B73 F7E3                <1>       mul    ebx
734 00007B75 014664              <1>       add    [esi+LD_ROOTBegin], eax
735 00007B78 EB2B                <1>       jmp    short loc_set_32bit_FAT_total_sectors
736                              <1>       ;
737                              <1> loc_set_FAT16_RootDirLoc:
738 00007B7A 0FB64616            <1>       movzx  eax, byte [esi+LD_BPB+BPB_NumFATs]
739 00007B7E 0FB7561C            <1>       movzx  edx, word [esi+LD_BPB+BPB_FATSz16]
740 00007B82 F7E2                <1>       mul    edx
741 00007B84 034660              <1>       add    eax, [esi+LD_FATBegin]
742 00007B87 894664              <1>       mov    [esi+LD_ROOTBegin], eax
743                              <1> loc_set_FAT16_data_begin:
744 00007B8A 894668              <1>       mov    [esi+LD_DATABegin], eax
745                              <1>       ;mov   eax, 20h  ; Size of a directory entry
746                              <1>       ;;movzx      edx, word [esi+LD_BPB+BPB_RootEntCnt]
747                              <1>         ;mov     dx, [esi+LD_BPB+BPB_RootEntCnt]
748                              <1>         ;mul edx
749                              <1>       ;;mov  ecx, 511
750                              <1>       ;mov   cx, 511
751                              <1>       ;add   eax, ecx
752                              <1>       ;inc   ecx ; 512
753                              <1>       ;div   ecx
754                              <1>       ; 14/07/2020
755 00007B8D 0FB74617            <1>       movzx  eax, word [esi+LD_BPB+BPB_RootEntCnt]
756 00007B91 6683C00F            <1>       add    ax, 15
757 00007B95 66C1E804            <1>       shr    ax, 4 ; / 16 ; (16 entries per sector)
758 00007B99 014668              <1>       add    [esi+LD_DATABegin], eax
759                              <1>       ;movzx eax, word [esi+LD_BPB+BPB_TotalSec16]
760 00007B9C 668B4619            <1>       mov    ax, [esi+LD_BPB+BPB_TotalSec16]
761 00007BA0 6685C0              <1>       test   ax, ax
762                              <1>       ;jz    short loc_set_32bit_FAT_total_sectors
763                              <1> ;loc_set_16bit_FAT_total_sectors:
```

```
                                <1>       ;mov   [esi+LD_TotalSectors], eax
                                <1>       ;jmp   short loc_set_hd_FAT_cluster_count
                                <1>       ; 14/07/2020
00007BA3 7503                   <1>       jnz    short loc_set_hd_FAT_cluster_count
                                <1> loc_set_32bit_FAT_total_sectors:
00007BA5 8B4626                 <1>       mov    eax, [esi+LD_BPB+BPB_TotalSec32]
                                <1>       ;mov   [esi+LD_TotalSectors], eax
                                <1> loc_set_hd_FAT_cluster_count:
00007BA8 894670                 <1>       mov    [esi+LD_TotalSectors], eax ; 14/07/2020
00007BAB 8B5668                 <1>       mov    edx, [esi+LD_DATABegin]
00007BAE 2B566C                 <1>       sub    edx, [esi+LD_StartSector]
00007BB1 29D0                   <1>       sub    eax, edx
00007BB3 31D2                   <1>       xor    edx, edx ; 0
00007BB5 0FB64E13               <1>         movzx   ecx, byte [esi+LD_BPB+BPB_SecPerClust]
00007BB9 F7F1                   <1>         div ecx
00007BBB 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
00007BBE E855010000             <1>       call   get_free_FAT_sectors
00007BC3 720D                   <1>       jc     short loc_validate_hd_FAT_partition_retn
00007BC5 894674                 <1>       mov    [esi+LD_FreeSectors], eax
00007BC8 C6467E06               <1>       mov    byte [esi+LD_MediaChanged], 6  ; Volume Name Reset
                                <1>
                                <1>       ; 15/07/2020
00007BCC FE05[50400100]         <1>       inc    byte [Last_DOS_DiskNo] ; > 1
                                <1>
                                <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:
00007BD2 C3                     <1>       retn
                                <1>
                                <1> validate_hd_fs_partition:
                                <1>       ; 03/02/2018
                                <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
00007BD3 8A6604                 <1>       mov    ah, [esi+ptFileSystemID]
00007BD6 80FCA1                 <1>       cmp    ah, 0A1h ; SINGLIX FS1 (trfs1) partition
00007BD9 7549                   <1>       jne    short loc_validate_hd_fs_partition_stc_retn
                                <1> loc_set_valid_hd_fs_partition_params:
00007BDB FE05[50400100]         <1>       inc    byte [Last_DOS_DiskNo] ; > 1
00007BE1 30C0                   <1>       xor    al, al ; mov al, 0
                                <1>       ;mov   [drv], dl
00007BE3 29DB                   <1>       sub    ebx, ebx ; 0
00007BE5 8A3D[50400100]         <1>       mov    bh, [Last_DOS_DiskNo]
00007BEB 81C300010900           <1>       add    ebx, Logical_DOSDisks
00007BF1 C6430102               <1>       mov    byte [ebx+LD_DiskType], 2
00007BF5 885302                 <1>       mov    [ebx+LD_PhyDrvNo], dl
                                <1>       ;mov   [ebx+LD_FATType], al ; 0
                                <1>       ;mov   [ebx+LD_FSType], ah
00007BF8 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:
00007BFC 89DD                   <1>       mov    ebp, ebx ; 10/01/2016
                                <1>       ;mov   dl, [ebx+LD_PhyDrvNo]
00007BFE A0[8A8F0100]           <1>       mov    al, [HD_LBA_yes] ; 10/01/2016
00007C03 884305                 <1>       mov    [ebx+LD_LBAYes], al
00007C06 89DE                   <1>       mov    esi, ebx
00007C08 BB[968F0100]           <1>       mov    ebx, DOSBootSectorBuff ; buffer address
00007C0D 08C0                   <1>       or     al, al
00007C0F 7515                   <1>       jnz    short loc_hd_fs_drv_init_load_bs_lba
                                <1> loc_hd_fs_drv_init_load_bs_chs:
00007C11 8A7601                 <1>       mov    dh, [esi+ptBeginHead]
00007C14 668B4E02               <1>       mov    cx, [esi+ptBeginSector]
00007C18 66B80102               <1>       mov    ax, 0201h ; Read 1 sector
                                <1>       ;mov   ebx, DOSBootSectorBuff
00007C1C E808D6FFFF             <1>       call   int13h
00007C21 7311                   <1>       jnc    short loc_hd_drv_fs_boot_validation
                                <1> loc_validate_hd_fs_partition_err_retn:
00007C23 C3                     <1>       retn
                                <1> loc_validate_hd_fs_partition_stc_retn:
00007C24 F9                     <1>       stc
00007C25 C3                     <1>       retn
                                <1> loc_hd_fs_drv_init_load_bs_lba:
                                <1>       ; DL = Physical drive number
                                <1>       ;mov   esi, ebx
00007C26 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)
00007C29 B41B                   <1>       mov    ah, 1Bh ; LBA read
00007C2B B001                   <1>       mov    al, 1 ; sector count
00007C2D E8F7D5FFFF             <1>       call   int13h
00007C32 72EF                   <1>       jc     short loc_validate_hd_fs_partition_err_retn
```

```
                            <1> loc_hd_drv_fs_boot_validation:
                            <1>       ;mov   esi, DOSBootSectorBuff
00007C34 89DE               <1>       mov    esi, ebx ; Boot sector buffer
00007C36 6681BEFE01000055AA <1>       cmp    word [esi+BS_Validation], 0AA55h
00007C3F 75E3               <1>       jne    short loc_validate_hd_fs_partition_stc_retn
                            <1>         ;
                            <1>       ;Singlix FS Extensions to TR-DOS (7/6/2009)
00007C41 66817E034653       <1>       cmp    word [esi+bs_FS_Identifier], 'FS' ; 03/02/2018
00007C47 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.
00007C49 807E09A1           <1>       cmp    byte [esi+bs_FS_PartitionID], 0A1h
00007C4D 75D5               <1>       jne    short loc_validate_hd_fs_partition_stc_retn
                            <1>       ;
00007C4F 89EF               <1>       mov    edi, ebp ; 10/01/2016
                            <1>       ;
00007C51 8A462D             <1>       mov    al, byte [esi+bs_FS_LBA_Ready]
00007C54 884705             <1>       mov    [edi+LD_FS_LBAYes], al
                            <1>       ;
                            <1>       ; 03/01/2010 CHS -> DOS FAT/BPB compatibility fix
00007C57 8A4608             <1>       mov    al, [esi+bs_FS_MediaAttrib]
00007C5A 884706             <1>       mov    byte [edi+LD_FS_MediaAttrib], al
                            <1>       ;
00007C5D 8A460A             <1>       mov    al, [esi+bs_FS_VersionMaj]
00007C60 884707             <1>       mov    [edi+LD_FS_VersionMajor], al
                            <1>       ;
00007C63 668B4606           <1>       mov    ax, [esi+bs_FS_BytesPerSec]
00007C67 66894711           <1>       mov    [edi+LD_FS_BytesPerSec], ax
00007C6B 8A462E             <1>       mov    al, [esi+bs_FS_SecPerTrack]
00007C6E 30E4               <1>       xor    ah, ah ; 09/12/2017
00007C70 6689471E           <1>       mov    [edi+LD_FS_SecPerTrack], ax
00007C74 8A462F             <1>       mov    al, [esi+bs_FS_Heads]
00007C77 66894720           <1>       mov    [edi+LD_FS_NumHeads], ax
                            <1>       ;
00007C7B 8B4628             <1>       mov    eax, [esi+bs_FS_UnDelDirD]
00007C7E 894722             <1>       mov    [edi+LD_FS_UnDelDirD], eax
00007C81 8B5618             <1>       mov    edx, [esi+bs_FS_MATLocation]
00007C84 89570C             <1>       mov    [edi+LD_FS_MATLocation], edx
00007C87 8B461C             <1>       mov    eax, [esi+bs_FS_RootDirD]
00007C8A 894708             <1>       mov    [edi+LD_FS_RootDirD], eax
00007C8D 8B460C             <1>       mov    eax, [esi+bs_FS_BeginSector]
00007C90 89476C             <1>       mov    [edi+LD_FS_BeginSector], eax
00007C93 8B4710             <1>       mov    eax, [edi+bs_FS_VolumeSize]
00007C96 894770             <1>       mov    [edi+LD_FS_VolumeSize], eax
                            <1>       ;
00007C99 89D0               <1>       mov    eax, edx ; [edi+LD_FS_MATLocation]
00007C9B 03476C             <1>       add    eax, [edi+LD_FS_BeginSector]
00007C9E 89FE               <1>       mov    esi, edi
                            <1> mread_hd_fs_MAT_sector:
                            <1>        ;mov  ebx, DOSBootSectorBuff
00007CA0 B901000000         <1>       mov    ecx, 1
00007CA5 E8A1AE0000         <1>       call   disk_read
00007CAA 7248               <1>       jc     short loc_validate_hd_fs_partition_retn
                            <1>       ; EDI will not be changed
00007CAC 89DE               <1>       mov    esi, ebx
                            <1> use_hdfs_mat_sector_params:
00007CAE 8B460C             <1>       mov    eax, [esi+FS_MAT_DATLocation]
00007CB1 894714             <1>       mov    [edi+LD_FS_DATLocation], eax
00007CB4 8B4610             <1>       mov    eax, [esi+FS_MAT_DATScount]
00007CB7 894718             <1>       mov    [edi+LD_FS_DATSectors], eax
00007CBA 8B4614             <1>       mov    eax, [esi+FS_MAT_FreeSectors]
00007CBD 894774             <1>         mov     [edi+LD_FS_FreeSectors], eax
00007CC0 8B4618             <1>       mov    eax, [esi+FS_MAT_FirstFreeSector]
00007CC3 894778             <1>       mov    [edi+LD_FS_FirstFreeSector], eax
00007CC6 8B4708             <1>       mov    eax, [edi+LD_FS_RootDirD]
00007CC9 03476C             <1>       add    eax, [edi+LD_FS_BeginSector]
00007CCC 89FE               <1>       mov    esi, edi
                            <1> read_hd_fs_RDT_sector:
00007CCE BB[968F0100]       <1>       mov    ebx, DOSBootSectorBuff
                            <1>       ;mov   ecx, 1
00007CD3 B101               <1>       mov    cl, 1
00007CD5 E871AE0000         <1>       call   disk_read
00007CDA 7218               <1>       jc     short loc_validate_hd_fs_partition_retn
                            <1>       ; EDI will not be changed
00007CDC 89DE               <1>       mov    esi, ebx
                            <1> use_hdfs_RDT_sector_params:
00007CDE 8B461C             <1>       mov    eax, [esi+FS_RDT_VolumeSerialNo]
00007CE1 894728             <1>       mov    [edi+LD_FS_VolumeSerial], eax
00007CE4 57                 <1>       push   edi
                            <1>       ;mov   ecx, 16
00007CE5 B110               <1>       mov    cl, 16
00007CE7 83C640             <1>       add    esi, FS_RDT_VolumeName
00007CEA 83C72C             <1>       add    edi, LD_FS_VolumeName
00007CED F3A5               <1>       rep    movsd ; 64 bytes
00007CEF 5E                 <1>       pop    esi
                            <1>              ; Volume Name Reset
00007CF0 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:
00007CF4 C3                 <1>       retn
                            <1>
                            <1> load_masterboot:
                            <1>       ; 14/07/2020 (Reset function has been removed)
                            <1>       ;
                            <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
                            <1>       ; 2005 - 2011
                            <1>       ; input -> DL = drive number
                            <1> ;     mov    ah, 0Dh ; Alternate disk reset
                            <1> ;     call   int13h
                            <1> ;     jnc    short pass_reset_error
                            <1> ;harddisk_error:
```

```
 974                                    <1> ;       retn
 975                                    <1> ;pass_reset_error:
 976 00007CF5 BB[868C0100]              <1>         mov     ebx, MasterBootBuff
 977 00007CFA 66B80102                  <1>         mov     ax, 0201h
 978 00007CFE 66B90100                  <1>         mov     cx, 1
 979 00007D02 30F6                      <1>         xor     dh, dh
 980 00007D04 E820D5FFFF                <1>         call    int13h
 981 00007D09 720C                      <1>         jc      short harddisk_error
 982                                    <1>         ;
 983 00007D0B 66813D[848E0100]55-       <1>         cmp     word [MBIDCode], 0AA55h
 983 00007D13 AA                        <1>
 984 00007D14 7401                      <1>         je      short load_masterboot_ok
 985 00007D16 F9                        <1>         stc
 986                                    <1> harddisk_error:
 987                                    <1> load_masterboot_ok:
 988 00007D17 C3                        <1>         retn
 989                                    <1>
 990                                    <1> get_free_FAT_sectors:
 991                                    <1>         ; 21/12/2017
 992                                    <1>         ; 29/02/2016
 993                                    <1>         ; 13/02/2016
 994                                    <1>         ; 04/02/2016
 995                                    <1>         ; 07/01/2016 (TRDOS 386 = TRDOS v2.0)
 996                                    <1>         ; 11/07/2010
 997                                    <1>         ; 21/06/2009
 998                                    <1>         ; INPUT: ESI = Logical DOS Drive Description Table address
 999                                    <1>         ; OUTPUT: STC => Error
1000                                    <1>           ;    cf = 0 and EAX = Free FAT sectors
1001                                    <1>         ; Also, related parameters and FAT buffer will be reset and updated
1002                                    <1>
1003 00007D18 31C0                      <1>         xor     eax, eax
1004                                    <1>         ;mov   [esi+LD_FreeSectors], eax ; Reset
1005                                    <1>
1006 00007D1A 807E0302                  <1>           cmp     byte [esi+LD_FATType], 2
1007 00007D1E 7654                      <1>         jna     short loc_gfc_get_fat_free_clusters
1008                                    <1>
1009                                    <1>         ; 29/02/2016
1010 00007D20 48                        <1>         dec     eax ; 0FFFFFFFFh
1011 00007D21 89463A                    <1>         mov     [esi+LD_BPB+BPB_Reserved], eax ; Free cluster count (reset)
1012 00007D24 89463E                    <1>         mov     [esi+LD_BPB+BPB_Reserved+4], eax ; First Free Cluster (reset)
1013 00007D27 40                        <1>         inc     eax ; 0
1014                                    <1>         ;
1015 00007D28 668B4636                  <1>         mov     ax, [esi+LD_BPB+BPB_FSInfo]
1016 00007D2C 03466C                    <1>         add     eax, [esi+LD_StartSector]
1017                                    <1>
1018 00007D2F BB[968F0100]              <1>         mov     ebx, DOSBootSectorBuff
1019 00007D34 B901000000                <1>         mov     ecx, 1
1020 00007D39 E80DAE0000                <1>         call    disk_read
1021 00007D3E 7301                      <1>         jnc     short loc_gfc_check_fsinfo_signs
1022                                    <1> retn_gfc_get_fsinfo_sec:
1023 00007D40 C3                        <1>         retn
1024                                    <1>
1025                                    <1> loc_gfc_check_fsinfo_signs:
1026 00007D41 BB[968F0100]              <1>         mov     ebx, DOSBootSectorBuff ; 13/02/2016
1027 00007D46 813B52526141              <1>           cmp     dword [ebx], 41615252h
1028 00007D4C 7524                      <1>         jne     short retn_gfc_get_fsinfo_stc
1029                                    <1>         ;add    ebx, 484
1030                                    <1>         ;cmp    dword [ebx], 61417272h
1031 00007D4E 81BBE4010000727241-       <1>         cmp     dword [ebx+484], 61417272h
1031 00007D57 61                        <1>
1032 00007D58 7518                      <1>         jne     short retn_gfc_get_fsinfo_stc
1033                                    <1>         ;add    ebx, 4
1034                                    <1>         ;mov    eax, [ebx]
1035 00007D5A 8B83E8010000              <1>         mov     eax, [ebx+488]
1036                                    <1>         ; 29/02/2016
1037 00007D60 89463A                    <1>         mov     [esi+LD_BPB+BPB_Reserved], eax ; Free cluster count
1038 00007D63 8B93EC010000              <1>         mov     edx,  [ebx+492]
1039 00007D69 89463E                    <1>         mov     [esi+LD_BPB+BPB_Reserved+4], eax ; First Free Cluster
1040                                    <1>         ; 21/12/2017
1041 00007D6C 89C3                      <1>         mov     ebx, eax ; (initial value = 0FFFFFFFFh)
1042 00007D6E 43                        <1>         inc     ebx ; 0FFFFFFFFh -> 0
1043 00007D6F 7513                      <1>         jnz     short short retn_from_get_free_fat32_clusters
1044 00007D71 C3                        <1>         retn
1045                                    <1>
1046                                    <1> retn_gfc_get_fsinfo_stc:
1047 00007D72 F9                        <1>         stc
1048 00007D73 C3                        <1>         retn
1049                                    <1>
1050                                    <1> loc_gfc_get_fat_free_clusters:
1051                                    <1>         ;mov   eax, 2
1052 00007D74 B002                      <1>         mov     al, 2
1053                                    <1>         ;mov   [FAT_CurrentCluster], eax
1054                                    <1> loc_gfc_loop_get_next_cluster:
1055 00007D76 E8EB4F0000                <1>         call    get_next_cluster
1056 00007D7B 730E                      <1>         jnc     short loc_gfc_free_fat_clusters_cont
1057 00007D7D 21C0                      <1>         and     eax, eax
1058 00007D7F 7411                      <1>         jz      short loc_gfc_pass_inc_free_cluster_count
1059                                    <1>
1060                                    <1> retn_from_get_free_fat_clusters:
1061 00007D81 8B4674                    <1>         mov     eax, [esi+LD_FreeSectors] ; Free clusters !
1062                                    <1> retn_from_get_free_fat32_clusters:
1063 00007D84 0FB65E13                  <1>           movzx     ebx, byte [esi+LD_BPB+BPB_SecPerClust]
1064 00007D88 F7E3                      <1>               mul     ebx
1065                                    <1>         ;mov   [esi+LD_FreeSectors], eax ; Free sectors
1066                                    <1> retn_get_free_sectors_calc:
1067 00007D8A C3                        <1>         retn
1068                                    <1>
1069                                    <1> loc_gfc_free_fat_clusters_cont:
1070 00007D8B 09C0                      <1>         or      eax, eax
1071 00007D8D 7503                      <1>         jnz     short loc_gfc_pass_inc_free_cluster_count
1072 00007D8F FF4674                    <1>         inc     dword [esi+LD_FreeSectors] ; Free clusters !
1073                                    <1>
1074                                    <1> loc_gfc_pass_inc_free_cluster_count:
1075                                    <1>         ;mov   eax, [FAT_CurrentCluster]
1076 00007D92 89C8                      <1>         mov     eax, ecx ; [FAT_CurrentCluster]
```

```
1077 00007D94 3B4678              <1>       cmp   eax, [esi+LD_Clusters]
1078 00007D97 77E8                <1>       ja    short retn_from_get_free_fat_clusters
1079 00007D99 40                  <1>       inc   eax
1080                              <1>       ;mov  [FAT_CurrentCluster], eax
1081 00007D9A EBDA                <1>       jmp   short loc_gfc_loop_get_next_cluster
1082                              <1>
1083                              <1> floppy_drv_init:
1084                              <1>       ; 09/12/2017
1085                              <1>       ; 06/07/2016
1086                              <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
1087                              <1>       ; 24/07/2011
1088                              <1>       ; 04/07/2009
1089                              <1>       ; INPUT ->
1090                              <1>       ;     DL = Drive number (0,1)
1091                              <1>       ; OUTPUT ->
1092                              <1>       ;     BL = drive name
1093                              <1>       ;     BH = drive number
1094                              <1>       ;     ESI = Logical DOS drv description table
1095                              <1>       ;     EAX = Volume serial number
1096                              <1>
1097 00007D9C BE[FE6D0000]        <1>       mov   esi, fd0_type ; 10/01/2016
1098 00007DA1 BF00010900          <1>       mov   edi, Logical_DOSDisks
1099 00007DA6 08D2                <1>       or    dl, dl
1100 00007DA8 7407                <1>       jz    short loc_drv_init_fd0_fd1
1101 00007DAA 81C700010000        <1>       add   edi, 100h
1102 00007DB0 46                  <1>       inc   esi ; fd1_type ; 10/01/2016
1103                              <1> loc_drv_init_fd0_fd1:
1104 00007DB1 C6477E00            <1>       mov   byte [edi+LD_MediaChanged], 0
1105 00007DB5 803E01              <1>       cmp   byte [esi], 1 ; type (>0 if it is existing)
1106                              <1>             ; 4 = 1.44 MB, 80 track, 3 1/2"
1107 00007DB8 7221                <1>       jb    short read_fd_boot_sector_retn
1108 00007DBA 885702              <1>       mov   [edi+LD_PhyDrvNo], dl
1109                              <1> read_fd_boot_sector:
1110 00007DBD 30F6                <1>       xor   dh, dh
1111 00007DBF B904000000          <1>       mov   ecx, 4 ; Retry Count
1112                              <1> read_fd_boot_sector_again:
1113 00007DC4 51                  <1>       push  ecx
1114                              <1>       ;mov  cx, 1
1115 00007DC5 B101                <1>       mov   cl, 1
1116 00007DC7 66B80102            <1>       mov   ax, 0201h ; Read 1 sector
1117 00007DCB BB[968F0100]        <1>       mov   ebx, DOSBootSectorBuff
1118 00007DD0 E854D4FFFF          <1>       call  int13h
1119 00007DD5 59                  <1>       pop   ecx
1120 00007DD6 7304                <1>       jnc   short use_fd_boot_sector_params
1121 00007DD8 E2EA                <1>       loop  read_fd_boot_sector_again
1122                              <1>
1123                              <1> read_fd_boot_sector_stc_retn:
1124 00007DDA F9                  <1>       stc
1125                              <1> read_fd_boot_sector_retn:
1126 00007DDB C3                  <1>       retn
1127                              <1>
1128                              <1> use_fd_boot_sector_params:
1129                              <1>       ;mov  esi, DOSBootSectorBuff
1130 00007DDC 89DE                <1>       mov   esi, ebx
1131 00007DDE 6681BEFE01000055AA  <1>       cmp   word [esi+BS_Validation], 0AA55h
1132 00007DE7 75F1                <1>       jne   short read_fd_boot_sector_stc_retn
1133 00007DE9 66817E035346        <1>         cmp     word [esi+bs_FS_Identifier], 'SF'
1134 00007DEF 0F85A2000000        <1>         jne     use_fd_fatfs_boot_sector_params
1135                              <1>       ;
1136 00007DF5 8A462D              <1>       mov   al, [esi+bs_FS_LBA_Ready]
1137 00007DF8 884705              <1>       mov   [edi+LD_FS_LBAYes], al
1138                              <1>       ;
1139                              <1>       ; 03/01/2010 CHS -> DOS FAT/BPB compatibility fix
1140 00007DFB 8A4608              <1>       mov   al, [esi+bs_FS_MediaAttrib]
1141 00007DFE 884706              <1>       mov   [edi+LD_FS_MediaAttrib], al
1142                              <1>       ;
1143 00007E01 8A460A              <1>         mov al, [esi+bs_FS_VersionMaj]
1144 00007E04 884707              <1>       mov   byte [edi+LD_FS_VersionMajor], al
1145 00007E07 668B4606            <1>       mov   ax, [esi+bs_FS_BytesPerSec]
1146 00007E0B 66894711            <1>       mov   [edi+LD_FS_BytesPerSec], ax
1147 00007E0F 8A462E              <1>       mov   al, [esi+bs_FS_SecPerTrack]
1148 00007E12 28E4                <1>       sub   ah, ah ; 09/12/2017
1149 00007E14 6689471E            <1>       mov   [edi+LD_FS_SecPerTrack], ax
1150 00007E18 8A462F              <1>       mov   al, [esi+bs_FS_Heads]
1151 00007E1B 66894720            <1>       mov   [edi+LD_FS_NumHeads], ax
1152                              <1>       ;
1153 00007E1F 8B4628              <1>       mov   eax, [esi+bs_FS_UnDelDirD]
1154 00007E22 894722              <1>       mov   [edi+LD_FS_UnDelDirD], eax
1155 00007E25 8B4618              <1>       mov   eax, [esi+bs_FS_MATLocation]
1156 00007E28 89470C              <1>       mov   [edi+LD_FS_MATLocation], eax
1157 00007E2B 8B461C              <1>       mov   eax, [esi+bs_FS_RootDirD]
1158 00007E2E 894708              <1>       mov   [edi+LD_FS_RootDirD], eax
1159 00007E31 8B460C              <1>       mov   eax, [esi+bs_FS_BeginSector]
1160 00007E34 89476C              <1>       mov   [edi+LD_FS_BeginSector], eax
1161 00007E37 8B4610              <1>       mov   eax, [esi+bs_FS_VolumeSize]
1162 00007E3A 894770              <1>       mov   [edi+LD_FS_VolumeSize], eax
1163                              <1>       ;
1164 00007E3D 89FE                <1>       mov   esi, edi
1165 00007E3F 8B460C              <1>       mov   eax, [esi+LD_FS_MATLocation]
1166                              <1>       ;add  eax, [edi+LD_FS_BeginSector]
1167                              <1> read_fd_MAT_sector_again:
1168                              <1>       ;mov  ebx, DOSBootSectorBuff
1169                              <1>       ;mov  ecx, 1
1170 00007E42 B101                <1>       mov   cl, 1
1171 00007E44 E808AD0000          <1>       call  chs_read
1172 00007E49 89DE                <1>       mov   esi, ebx
1173 00007E4B 7301                <1>       jnc   short use_fdfs_mat_sector_params
1174                              <1>       ;jmp  short read_fd_boot_sector_retn
1175 00007E4D C3                  <1>       retn
1176                              <1> use_fdfs_mat_sector_params:
1177 00007E4E 8B460C              <1>       mov   eax, [esi+FS_MAT_DATLocation]
1178 00007E51 894714              <1>       mov   [edi+LD_FS_DATLocation], eax
1179 00007E54 8B4610              <1>       mov   eax, [esi+FS_MAT_DATScount]
1180 00007E57 894718              <1>       mov   [edi+LD_FS_DATSectors], eax
1181 00007E5A 8B4714              <1>       mov   eax, [edi+FS_MAT_FreeSectors]
```

```
1182 00007E5D 894774            <1>       mov    [edi+LD_FS_FreeSectors], eax
1183 00007E60 8B4618            <1>       mov    eax, [esi+FS_MAT_FirstFreeSector]
1184 00007E63 894778            <1>       mov    [edi+LD_FS_FirstFreeSector], eax
1185                            <1>       ;
1186 00007E66 89FE              <1>       mov    esi, edi
1187 00007E68 8B4608            <1>       mov    eax, [esi+LD_FS_RootDirD]
1188                            <1> read_fd_RDT_sector_again:
1189                            <1>       ;mov   ebx, DOSBootSectorBuff
1190                            <1>       ;mov   cx, 1
1191 00007E6B B101              <1>       mov    cl, 1
1192 00007E6D E8DFAC0000        <1>       call   chs_read
1193 00007E72 89DE              <1>       mov    esi, ebx
1194 00007E74 7220              <1>       jc     short read_fd_RDT_sector_retn
1195                            <1> use_fdfs_RDT_sector_params:
1196 00007E76 8B461C            <1>       mov    eax, [esi+FS_RDT_VolumeSerialNo]
1197 00007E79 894728            <1>       mov    [edi+LD_FS_VolumeSerial], eax
1198 00007E7C 57                <1>       push   edi
1199                            <1>       ;mov   ecx, 16
1200 00007E7D B110              <1>       mov    cl, 16
1201 00007E7F 83C640            <1>       add    esi, FS_RDT_VolumeName
1202 00007E82 83C72C            <1>       add    edi, LD_FS_VolumeName
1203 00007E85 F3A5              <1>       rep    movsd ; 64 bytes
1204 00007E87 5E                <1>       pop    esi
1205 00007E88 C6460300          <1>       mov    byte [esi+LD_FATType], 0
1206 00007E8C C64604A1          <1>       mov    byte [esi+LD_FSType], 0A1h
1207 00007E90 E9A5000000        <1>         jmp     loc_cont_use_fd_boot_sector_params
1208                            <1>
1209                            <1> read_fd_RDT_sector_stc_retn:
1210 00007E95 F9                <1>       stc
1211                            <1> read_fd_RDT_sector_retn:
1212 00007E96 C3                <1>       retn
1213                            <1>
1214                            <1> use_fd_fatfs_boot_sector_params:
1215 00007E97 807E2629          <1>       cmp    byte [esi+BS_BootSig], 29h
1216 00007E9B 75F8              <1>       jne    short read_fd_RDT_sector_stc_retn
1217 00007E9D 807E15F0          <1>       cmp    byte [esi+BPB_Media], 0F0h
1218 00007EA1 72F3              <1>       jb     short read_fd_RDT_sector_retn
1219 00007EA3 57                <1>       push   edi
1220 00007EA4 83C706            <1>       add    edi, LD_BPB
1221                            <1>       ;mov   ecx, 16
1222 00007EA7 B110              <1>       mov    cl, 16
1223 00007EA9 F3A5              <1>       rep    movsd ; 64 bytes
1224 00007EAB 5E                <1>       pop    esi
1225 00007EAC 31C0              <1>       xor    eax, eax
1226 00007EAE 89466C            <1>       mov    [esi+LD_StartSector], eax ; 0
1227 00007EB1 668B461C          <1>       mov    ax, [esi+LD_BPB+BPB_FATSz16]
1228 00007EB5 8A4E16            <1>       mov    cl, [esi+LD_BPB+BPB_NumFATs]
1229 00007EB8 F7E1              <1>       mul    ecx
1230                            <1>       ; edx = 0 !
1231 00007EBA 668B5614          <1>       mov    dx, [esi+LD_BPB+BPB_RsvdSecCnt]
1232 00007EBE 66895660          <1>       mov    [esi+LD_FATBegin], dx
1233                            <1>       ;add   eax, edx
1234 00007EC2 6601D0            <1>       add    ax, dx
1235 00007EC5 894664            <1>       mov    [esi+LD_ROOTBegin], eax
1236 00007EC8 894668            <1>       mov    [esi+LD_DATABegin], eax
1237 00007ECB 668B5617          <1>       mov    dx, [esi+LD_BPB+BPB_RootEntCnt]
1238                            <1>       ;;shl  edx, 5 ; * 32 (Size of a directory entry)
1239                            <1>       ;shl   dx, 5
1240                            <1>       ;;add  edx, 511
1241                            <1>       ;add   dx, 511
1242                            <1>       ;;shr  edx, 9 ; edx = ((edx*32)+511) / 512
1243                            <1>       ;shr   dx, 9
1244 00007ECF 6683C20F          <1>       add    dx, 15 ; 06/07/2016 (+(512/32)-1)
1245 00007ED3 66C1EA04          <1>       shr    dx, 4 ; / 16 (==16 entries per sector)
1246 00007ED7 015668            <1>       add    [esi+LD_DATABegin], edx ; + rd sectors
1247                            <1>       ;movzx eax, word [esi+LD_BPB+BPB_TotalSec16]
1248 00007EDA 668B4619          <1>       mov    ax, [esi+LD_BPB+BPB_TotalSec16]
1249 00007EDE 894670            <1>       mov    [esi+LD_TotalSectors], eax
1250 00007EE1 2B4668            <1>       sub    eax, [esi+LD_DATABegin]
1251                            <1>       ;movzx ecx, byte [esi+LD_BPB+BPB_SecPerClust]
1252 00007EE4 8A4E13            <1>       mov    cl, [esi+LD_BPB+BPB_SecPerClust]
1253 00007EE7 80F901            <1>       cmp    cl, 1
1254 00007EEA 7605              <1>       jna    short save_fd_fatfs_cluster_count
1255                            <1>       ;sub   edx, edx
1256 00007EEC 6629D2            <1>       sub    dx, dx ; 0
1257                            <1>       ;sub   dl, dl ; 06/07/2016
1258 00007EEF F7F1              <1>       div    ecx
1259                            <1> save_fd_fatfs_cluster_count:
1260 00007EF1 894678            <1>       mov    [esi+LD_Clusters], eax
1261                            <1>
1262                            <1>       ; Maximum Valid Cluster Number = EAX +1
1263                            <1>       ; with 2 reserved clusters= EAX +2
1264                            <1>
1265                            <1> reset_FAT_buffer_decriptors:
1266 00007EF4 29C0              <1>       sub    eax, eax ; 0
1267 00007EF6 A2[9A910100]      <1>       mov    [FAT_BuffValidData], al ; 0
1268 00007EFB A2[9B910100]      <1>       mov    [FAT_BuffDrvName], al ; 0
1269 00007F00 A3[9E910100]      <1>       mov    [FAT_BuffSector], eax ; 0
1270                            <1>
1271                            <1> read_fd_FAT_sectors:
1272 00007F05 BB001C0900        <1>       mov    ebx, FAT_Buffer
1273 00007F0A 668B4614          <1>       mov    ax, [esi+LD_BPB+BPB_RsvdSecCnt]
1274                            <1>       ;mov   ecx, 3
1275 00007F0E B103              <1>       mov    cl, 3 ; 3 sectors
1276 00007F10 E83CAC0000        <1>       call   chs_read
1277 00007F15 7240              <1>       jc     short read_fd_FAT_sectors_retn
1278                            <1> use_fd_FAT_sectors:
1279 00007F17 8A4602            <1>       mov    al, [esi+LD_PhyDrvNo]
1280 00007F1A 0441              <1>       add    al, 'A'
1281 00007F1C A2[9B910100]      <1>       mov    [FAT_BuffDrvName], al
1282 00007F21 C605[9A910100]01  <1>       mov    byte [FAT_BuffValidData], 1
1283 00007F28 E82B000000        <1>       call   fd_init_calculate_free_clusters
1284 00007F2D 7228              <1>       jc     short read_fd_FAT_sectors_retn
1285                            <1>
1286                            <1> loc_use_fd_boot_sector_params_FAT:
```

```
1287 00007F2F C6460301          <1>       mov    byte [esi+LD_FATType], 1 ; FAT 12
1288 00007F33 C6460401          <1>       mov    byte [esi+LD_FSType], 1
1289 00007F37 8B462D            <1>         mov     eax, [esi+LD_BPB+VolumeID]
1290                            <1> loc_cont_use_fd_boot_sector_params:
1291 00007F3A 8A7E02            <1>       mov    bh, [esi+LD_PhyDrvNo]
1292 00007F3D 887E7D            <1>       mov    [esi+LD_DParamEntry], bh
1293 00007F40 88FB              <1>       mov    bl, bh
1294 00007F42 80C341            <1>       add    bl, 'A'
1295 00007F45 881E              <1>       mov    byte [esi+LD_Name], bl
1296 00007F47 C6460101          <1>       mov    byte [esi+LD_DiskType], 1
1297 00007F4B C6460500          <1>       mov    byte [esi+LD_LBAYes], 0
1298 00007F4F C6467C00          <1>       mov    byte [esi+LD_PartitionEntry], 0
1299 00007F53 C6467E06          <1>       mov    byte [esi+LD_MediaChanged], 6 ; Volume Name Reset
1300                            <1>
1301                            <1> read_fd_FAT_sectors_retn:
1302 00007F57 C3                <1>       retn
1303                            <1>
1304                            <1> fd_init_calculate_free_clusters:
1305                            <1>       ; 09/12/2017
1306                            <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
1307                            <1>       ; 04/07/2009
1308                            <1>       ; INPUT ->
1309                            <1>       ;     ESI = Logical DOS drive description table address
1310                            <1>       ; OUTPUT ->
1311                            <1>       ;    [ESI+LD_FreeSectors] will be set
1312                            <1>
1313 00007F58 29C0              <1>       sub    eax, eax
1314 00007F5A 894674            <1>       mov    [esi+LD_FreeSectors], eax ; 0
1315 00007F5D B002              <1>       mov    al, 2 ; eax = 2
1316                            <1>
1317                            <1> fd_init_loop_get_next_cluster:
1318 00007F5F E830000000        <1>       call   fd_init_get_next_cluster
1319 00007F64 722D              <1>       jc     short fd_init_calculate_free_clusters_retn
1320                            <1>
1321                            <1> fd_init_free_fat_clusters:
1322                            <1>       ;cmp   eax, 0
1323                            <1>       ;ja    short fd_init_pass_inc_free_cluster_count
1324                            <1>       ;and   eax, eax
1325                            <1>       ;jnz   short fd_init_pass_inc_free_cluster_count
1326 00007F66 6621C0            <1>       and    ax, ax
1327 00007F69 7504              <1>       jnz    short fd_init_pass_inc_free_cluster_count
1328                            <1>       ;inc   dword [esi+LD_FreeSectors]
1329 00007F6B 66FF4674          <1>         inc  word [esi+LD_FreeSectors]
1330                            <1>
1331                            <1> fd_init_pass_inc_free_cluster_count:
1332                            <1>       ;mov   eax, [FAT_CurrentCluster]
1333 00007F6F 66A1[96910100]    <1>       mov    ax, [FAT_CurrentCluster]
1334                            <1>       ;cmp   eax, [esi+LD_Clusters]
1335 00007F75 663B4678          <1>       cmp    ax, [esi+LD_Clusters]
1336 00007F79 7704              <1>       ja     short short retn_from_fd_init_calculate_free_clusters
1337                            <1>       ;inc   eax
1338 00007F7B 6640              <1>       inc    ax
1339 00007F7D EBE0              <1>       jmp    short fd_init_loop_get_next_cluster
1340                            <1>
1341                            <1> retn_from_fd_init_calculate_free_clusters:
1342 00007F7F 8A4613            <1>       mov    al, [esi+LD_BPB+BPB_SecPerClust]
1343 00007F82 3C01              <1>       cmp    al, 1
1344 00007F84 760D              <1>       jna    short fd_init_calculate_free_clusters_retn
1345                            <1>       ;movzx eax, al
1346 00007F86 30E4              <1>       xor    ah, ah ; 09/12/2017
1347                            <1>       ;mov   ecx, [esi+LD_FreeSectors]
1348 00007F88 668B4E74          <1>       mov    cx, [esi+LD_FreeSectors] ; Count of free clusters
1349                            <1>       ;mul   ecx
1350 00007F8C 66F7E1            <1>       mul    cx
1351                            <1>       ;mov   [esi+LD_FreeSectors], eax
1352 00007F8F 66894674          <1>       mov    [esi+LD_FreeSectors], ax
1353                            <1> fd_init_calculate_free_clusters_retn:
1354 00007F93 C3                <1>       retn
1355                            <1>
1356                            <1> fd_init_get_next_cluster:
1357                            <1>       ; 04/02/2016
1358                            <1>       ; 02/02/2016
1359                            <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
1360                            <1>       ; 04/07/2009
1361                            <1>       ; INPUT ->
1362                            <1>       ;    EAX = Current cluster
1363                            <1>       ;    ESI = Logical DOS drive description table address
1364                            <1>       ;    EDX = 0
1365                            <1>       ; OUTPUT ->
1366                            <1>       ;    EAX = Next cluster
1367                            <1>
1368 00007F94 A3[96910100]      <1>       mov    [FAT_CurrentCluster], eax
1369                            <1> fd_init_get_next_cluster_readnext:
1370 00007F99 29D2              <1>       sub    edx, edx ; 0
1371 00007F9B BB00040000        <1>       mov    ebx, 1024 ; 400h
1372 00007FA0 F7F3              <1>       div    ebx
1373                            <1>       ; EAX = Count of 3 FAT sectors
1374                            <1>       ; EDX = Buffer entry index
1375 00007FA2 89C1              <1>       mov    ecx, eax
1376                            <1>       ;mov   eax, 3
1377 00007FA4 B003              <1>       mov    al, 3
1378 00007FA6 F7E2              <1>       mul    edx ; Multiply by 3
1379 00007FA8 66D1E8            <1>       shr    ax, 1 ; Divide by 2
1380 00007FAB 89C3              <1>       mov    ebx, eax ; Buffer byte offset
1381 00007FAD 81C3001C0900      <1>       add    ebx, FAT_Buffer
1382 00007FB3 89C8              <1>       mov    eax, ecx
1383                            <1>       ;mov   edx, 3
1384 00007FB5 66BA0300          <1>       mov    dx, 3
1385 00007FB9 F7E2              <1>       mul    edx
1386                            <1>       ; EAX = FAT Beginning Sector
1387                            <1>       ; EDX = 0
1388 00007FBB 8A0E              <1>       mov    cl, [esi+LD_Name]
1389                            <1>       ;cmp   byte [FAT_BuffValidData], 0
1390                            <1>       ;jna   short fd_init_load_FAT_sectors0
1391 00007FBD 3A0D[9B910100]    <1>       cmp    cl, [FAT_BuffDrvName]
```

```
1392 00007FC3 751E                <1>       jne    short fd_init_load_FAT_sectors0
1393 00007FC5 3B05[9E910100]      <1>       cmp    eax, [FAT_BuffSector]
1394 00007FCB 751C                <1>       jne    short fd_init_load_FAT_sectors1
1395                              <1>       ;mov   eax, [FAT_CurrentCluster]
1396 00007FCD A0[96910100]        <1>       mov    al, [FAT_CurrentCluster]
1397                              <1>       ;shr   eax, 1
1398 00007FD2 D0E8                <1>       shr    al, 1
1399 00007FD4 668B03              <1>       mov    ax, [ebx]
1400 00007FD7 7306                <1>       jnc    short fd_init_gnc_even
1401 00007FD9 66C1E804            <1>       shr    ax, 4
1402                              <1> fd_init_gnc_clc_retn:
1403 00007FDD F8                  <1>       clc
1404 00007FDE C3                  <1>       retn
1405                              <1>
1406                              <1> fd_init_gnc_even:
1407 00007FDF 80E40F              <1>       and    ah, 0Fh
1408 00007FE2 C3                  <1>       retn
1409                              <1>
1410                              <1> fd_init_load_FAT_sectors0:
1411 00007FE3 880D[9B910100]      <1>       mov    [FAT_BuffDrvName], cl
1412                              <1> fd_init_load_FAT_sectors1:
1413 00007FE9 C605[9A910100]00    <1>       mov    byte [FAT_BuffValidData], 0
1414 00007FF0 A3[9E910100]        <1>       mov    [FAT_BuffSector], eax
1415 00007FF5 034660              <1>       add    eax, [esi+LD_FATBegin]
1416 00007FF8 BB001C0900          <1>       mov    ebx, FAT_Buffer
1417                              <1>       ;movzx ecx, word [esi+LD_BPB+BPB_FATSz16]
1418 00007FFD 668B4E1C            <1>       mov    cx, [esi+LD_BPB+BPB_FATSz16]
1419 00008001 662B0D[9E910100]    <1>       sub    cx, [FAT_BuffSector]
1420                              <1>         ;cmp ecx, 3
1421 00008008 6683F903            <1>       cmp    cx, 3
1422 0000800C 7605                <1>       jna    short fdinit_pass_fix_sector_count_3
1423                              <1>       ;mov   ecx, 3
1424 0000800E B903000000          <1>       mov    ecx, 3
1425                              <1> fdinit_pass_fix_sector_count_3:
1426 00008013 E839AB0000          <1>       call   chs_read
1427 00008018 730D                <1>       jnc    short fd_init_FAT_sectors_no_load_error
1428 0000801A C605[9A910100]00    <1>       mov    byte [FAT_BuffValidData], 0
1429                              <1>              ; Drv not ready or read Error !
1430 00008021 B80F000000          <1>       mov    eax, ERR_DRV_NOT_RDY ; 15
1431                              <1>       ;xor   edx, edx
1432 00008026 C3                  <1>       retn
1433                              <1>
1434                              <1> fd_init_FAT_sectors_no_load_error:
1435 00008027 C605[9A910100]01    <1>       mov    byte [FAT_BuffValidData], 1
1436 0000802E A1[96910100]        <1>       mov    eax, [FAT_CurrentCluster]
1437 00008033 E961FFFFFF          <1>         jmp     fd_init_get_next_cluster_readnext
1438                              <1>
1439                              <1> get_FAT_volume_name:
1440                              <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
1441                              <1>       ; 12/09/2009
1442                              <1>       ; INPUT ->
1443                              <1>       ;      BH = Logical DOS drive number (0,1,2,3,4 ...)
1444                              <1>       ;       BL = 0
1445                              <1>       ; OUTPUT ->
1446                              <1>       ;      CF = 0 -> ESI = Volume name address
1447                              <1>       ;      CF = 1 -> Root volume name not found
1448                              <1>
1449                              <1>       ;mov   ah, 0FFh
1450                              <1>       ;mov   al, [Last_Dos_DiskNo]
1451                              <1>       ;cmp   al, bh
1452                              <1>       ;jb     short loc_gfvn_dir_load_err
1453                              <1>
1454 00008038 89DE                <1>       mov    esi, ebx
1455 0000803A 81E600FF0000        <1>       and    esi, 0FF00h ; esi = bh
1456 00008040 81C600010900        <1>       add    esi, Logical_DOSDisks
1457 00008046 8A06                <1>       mov     al, [esi+LD_Name]
1458 00008048 8A6603              <1>       mov     ah, [esi+LD_FATType]
1459 0000804B 80FC01              <1>       cmp     ah, 1
1460 0000804E 7210                <1>       jb     short loc_gfvn_dir_load_err
1461 00008050 3C41                <1>       cmp    al, 'A'
1462 00008052 720C                <1>       jb      short loc_gfvn_dir_load_err
1463 00008054 80FC02              <1>       cmp    ah, 2
1464 00008057 7708                <1>       ja      short get_FAT32_root_cluster
1465                              <1>
1466 00008059 E8634E0000          <1>       call    load_FAT_root_directory
1467 0000805E 730B                <1>       jnc     short loc_get_volume_name
1468                              <1>
1469                              <1> loc_gfvn_dir_load_err:
1470 00008060 C3                  <1>       retn
1471                              <1>
1472                              <1> get_FAT32_root_cluster:
1473 00008061 8B4632              <1>       mov    eax, [esi+LD_BPB+BPB_RootClus]
1474 00008064 E8E34E0000          <1>       call    load_FAT_sub_directory
1475 00008069 7224                <1>       jc     short loc_get_volume_name_retn
1476                              <1>
1477                              <1> loc_get_volume_name:
1478 0000806B BE00000800          <1>         mov     esi, Directory_Buffer
1479 00008070 6631C9              <1>       xor    cx, cx ; 0
1480                              <1> check_root_volume_name:
1481 00008073 8A06                <1>       mov    al, [esi]
1482 00008075 08C0                <1>       or      al, al
1483 00008077 7416                <1>       jz      short loc_get_volume_name_retn
1484 00008079 807E0B08            <1>       cmp     byte [esi+0Bh], 08h
1485 0000807D 7410                <1>       je      short loc_get_volume_name_retn
1486 0000807F 663B0D[AF910100]    <1>       cmp     cx, [DirBuff_LastEntry]
1487 00008086 7308                <1>       jnb     short pass_check_root_volume_name
1488 00008088 6641                <1>       inc     cx
1489 0000808A 83C620              <1>       add     esi, 32
1490 0000808D EBE4                <1>       jmp     short check_root_volume_name
1491                              <1>
1492                              <1> loc_get_volume_name_retn:
1493 0000808F C3                  <1>       retn
1494                              <1>
1495                              <1> pass_check_root_volume_name:
1496 00008090 803D[AB910100]03    <1>       cmp    byte [DirBuff_FATType], 3
```

```
1497 00008097 7230              <1>       jb     short loc_get_volume_name_retn_xor
1498                            <1>
1499 00008099 BB001C0900        <1>       mov    ebx, FAT_Buffer
1500 0000809E BE00010900        <1>       mov    esi, Logical_DOSDisks
1501 000080A3 31C0              <1>       xor    eax, eax
1502 000080A5 8A25[AA910100]    <1>       mov    ah, [DirBuff_DRV]
1503 000080AB 80EC41            <1>       sub    ah, 'A'
1504 000080AE 01C6              <1>       add    esi, eax
1505 000080B0 A1[B1910100]      <1>       mov    eax, [DirBuff_Cluster]
1506 000080B5 E8AC4C0000        <1>       call   get_next_cluster
1507 000080BA 7305              <1>       jnc    short loc_gfvn_load_FAT32_dir_cluster
1508                            <1>
1509 000080BC 83F801            <1>       cmp     eax, 1
1510 000080BF F5                <1>       cmc
1511 000080C0 C3                <1>       retn
1512                            <1>
1513                            <1> loc_gfvn_load_FAT32_dir_cluster:
1514 000080C1 E8864E0000        <1>       call   load_FAT_sub_directory
1515 000080C6 73A3              <1>       jnc    short loc_get_volume_name
1516 000080C8 C3                <1>       retn
1517                            <1>
1518                            <1> loc_get_volume_name_retn_xor:
1519 000080C9 31C0              <1>       xor    eax, eax
1520 000080CB C3                <1>       retn
1521                            <1>
1522                            <1> get_media_change_status:
1523                            <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
1524                            <1>       ; 09/09/2009
1525                            <1>       ; INPUT:
1526                            <1>       ;     DL = Drive number (physical)
1527                            <1>       ; OUTPUT: clc & AH = 6 media changed
1528                            <1>       ;     clc & AH = 0 media not changed
1529                            <1>       ;     stc -> Drive not ready or an error
1530                            <1>
1531 000080CC B416              <1>       mov    ah, 16h
1532 000080CE E856D1FFFF        <1>       call   int13h
1533 000080D3 80FC06            <1>       cmp    ah, 06h
1534 000080D6 7405              <1>       je     short loc_gmc_status_retn
1535 000080D8 08E4              <1>       or     ah, ah
1536 000080DA 7401              <1>       jz     short loc_gmc_status_retn
1537                            <1> loc_gmc_status_stc_retn:
1538 000080DC F9                <1>       stc
1539                            <1> loc_gmc_status_retn:
1540 000080DD C3                <1>       retn
3081                                %include 'trdosk3.s' ; 06/01/2016
   1                            <1> ; ****************************************************************************
   2                            <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - MAIN PROGRAM : trdosk3.s
   3                            <1> ; ----------------------------------------------------------------------------
   4                            <1> ; Last Update: 31/12/2017
   5                            <1> ; ----------------------------------------------------------------------------
   6                            <1> ; Beginning: 06/01/2016
   7                            <1> ; ----------------------------------------------------------------------------
   8                            <1> ; Assembler: NASM version 2.11 (trdos386.s)
   9                            <1> ; ----------------------------------------------------------------------------
  10                            <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
  11                            <1> ; MAINPROG.ASM (09/11/2011)
  12                            <1> ; ****************************************************************************
  13                            <1> ; MAINPROG.ASM [ TRDOS KERNEL - COMMAND EXECUTER SECTION - MAIN PROGRAM ]
  14                            <1> ; (c) 2004-2011  Erdogan TAN  [ 17/01/2004 ]  Last Update: 09/11/2011
  15                            <1> ; CMD_INTR.ASM [ TRDOS Command Interpreter Procedure ] Last Update: 09/11/2011
  16                            <1> ; DIR.ASM [ DIRECTORY FUNCTIONS ] Last Update: 09/10/2011
  17                            <1> ; FILE.ASM [ FILE FUNCTIONS ] Last Update: 09/10/2011
  18                            <1>
  19                            <1> change_current_drive:
  20                            <1>       ; 16/10/2016
  21                            <1>       ; 02/02/2016
  22                            <1>       ; 15/01/2016 (TRDOS 386 = TRDOS v2.0)
  23                            <1>       ; 18/08/2011
  24                            <1>       ; 09/09/2009
  25                            <1>       ; INPUT:
  26                            <1>       ;   DL = Logical DOS Drive Number
  27                            <1>       ; OUTPUT:
  28                            <1>       ;  cf=1 -> Not successful
  29                            <1>       ;   EAX = Error code
  30                            <1>       ;  cf=0 ->
  31                            <1>       ;   EAX = 0 (successful)
  32                            <1>
  33 000080DE 31DB              <1>       xor    ebx, ebx
  34 000080E0 88D7              <1>       mov    bh, dl
  35                            <1>
  36                            <1>       ;cmp   dl, 1
  37                            <1>       ;jna   short loc_ccdrv_initial_media_change_check
  38                            <1>       ;cmp   bh, [Last_Dos_DiskNo]
  39                            <1>       ;ja    short loc_ccdrv_drive_not_ready_err
  40                            <1>
  41                            <1> loc_ccdrv_initial_media_change_check:
  42 000080E2 BE00010900        <1>       mov    esi, Logical_DOSDisks
  43 000080E7 01DE              <1>       add    esi, ebx
  44                            <1> loc_ccdrv_dos_drive_name_check:
  45 000080E9 80FA02            <1>       cmp    dl, 2
  46 000080EC 720F              <1>       jb     short loc_ccdrv_dos_drive_name_check_ok
  47                            <1>
  48 000080EE 8A06              <1>       mov    al, [esi+LD_Name]
  49 000080F0 2C41              <1>       sub    al, 'A'
  50 000080F2 38D0              <1>       cmp    al, dl
  51 000080F4 7407              <1>       je     short loc_ccdrv_dos_drive_name_check_ok
  52                            <1>
  53                            <1> loc_ccdrv_drive_not_ready_err:
  54                            <1>       ; 16/10/2016 (15h -> 15)
  55 000080F6 B80F000000        <1>       mov    eax, 15 ; Drive not ready
  56                            <1> loc_change_current_drive_stc_retn:
  57 000080FB F9                <1>       stc
  58 000080FC C3                <1>       retn
  59                            <1>
  60                            <1> loc_ccdrv_dos_drive_name_check_ok:
```

```
000080FD 8A667E             <1>       mov    ah, [esi+LD_MediaChanged]
00008100 80FC06             <1>       cmp    ah, 6  ; VOLUME NAME CHECK/MOVE SIGN
00008103 7455               <1>       je     short loc_ccdrv_get_FAT_volume_name_0
                            <1>
00008105 80FA01             <1>       cmp    dl, 1
00008108 777D               <1>       ja     short loc_gmcs_init_drv_hd
                            <1>
                            <1> loc_gmcs_init_drv_fd:
0000810A 08E4               <1>       or     ah, ah
                            <1>       ; AH = 1 is initialization sign (invalid_fd_parameter)
0000810C 7517               <1>       jnz    short loc_ccdrv_call_fd_init
                            <1>
0000810E E8B9FFFFFF         <1>       call   get_media_change_status
00008113 72E1               <1>       jc     short loc_ccdrv_drive_not_ready_err
                            <1>
00008115 20E4               <1>       and    ah, ah
00008117 7476               <1>       jz     short loc_change_current_drv3
                            <1>
00008119 80F406             <1>       xor    ah, 6
0000811C 75D8               <1>       jnz    short loc_ccdrv_drive_not_ready_err
                            <1>
                            <1> loc_ccdrv_call_fd_init_check_vol_id:
0000811E E8440A0000         <1>       call   get_volume_serial_number
00008123 730D               <1>       jnc    short loc_ccdrv_check_vol_serial
                            <1>
                            <1> loc_ccdrv_call_fd_init:
00008125 E872FCFFFF         <1>       call   floppy_drv_init
0000812A 731A               <1>       jnc    short loc_reset_drv_fd_current_dir
                            <1>
                            <1> loc_ccdrv_fdinit_fail_retn:
                            <1>       ; 16/10/2016
0000812C B80F000000         <1>       mov    eax, 15 ; Drive not ready
00008131 C3                 <1>       retn
                            <1>
                            <1> loc_ccdrv_check_vol_serial:
00008132 A3[7C8A0100]       <1>       mov    [Current_VolSerial], eax
                            <1>       ;mov   dl, bh
00008137 E860FCFFFF         <1>       call   floppy_drv_init
0000813C 72EE               <1>       jc     short loc_ccdrv_fdinit_fail_retn
                            <1>
0000813E 3B05[7C8A0100]     <1>       cmp    eax, [Current_VolSerial]
00008144 7445               <1>       je     short loc_change_current_drv2
                            <1>
                            <1> loc_reset_drv_fd_current_dir:
00008146 31C0               <1>       xor    eax, eax
00008148 88467F             <1>         mov  [esi+LD_CDirLevel], al
0000814B 89F7               <1>       mov    edi, esi
0000814D 81C780000000       <1>       add    edi, LD_CurrentDirectory
00008153 B920000000         <1>       mov    ecx, 32
00008158 F3AB               <1>       rep    stosd
                            <1>
                            <1> loc_ccdrv_get_FAT_volume_name_0:
0000815A 8A4603             <1>       mov    al, [esi+LD_FATType]
0000815D 08C0               <1>       or     al, al
0000815F 742A               <1>       jz     short loc_change_current_drv2
                            <1>
00008161 56                 <1>       push   esi
00008162 3C02               <1>       cmp    al, 2
00008164 7705               <1>       ja     short loc_ccdrv_get_FAT32_vol_name
                            <1>
                            <1> loc_ccdrv_get_FAT2_16_vol_name:
00008166 83C631             <1>       add    esi, LD_BPB + VolumeLabel
00008169 EB03               <1>       jmp    short loc_ccdrv_get_FAT_volume_name_1
                            <1>
                            <1> loc_ccdrv_get_FAT32_vol_name:
0000816B 83C64D             <1>       add    esi, LD_BPB + FAT32_VolLab
                            <1> loc_ccdrv_get_FAT_volume_name_1:
0000816E 53                 <1>       push   ebx
0000816F 56                 <1>       push   esi
00008170 E8C3FEFFFF         <1>       call   get_FAT_volume_name
00008175 5F                 <1>       pop    edi
00008176 5B                 <1>       pop    ebx
                            <1>       ; BL = 0
00008177 720B               <1>       jc     short loc_change_current_drv1
00008179 20C0               <1>       and    al, al
0000817B 7407               <1>       jz     short loc_change_current_drv1
                            <1>
                            <1> loc_ccdrv_move_FAT_volume_name:
0000817D B90B000000         <1>       mov    ecx, 11
00008182 F3A4               <1>       rep    movsb
                            <1>
                            <1> loc_change_current_drv1:
00008184 5E                 <1>       pop    esi
00008185 EB04               <1>       jmp    short loc_change_current_drv2
                            <1>
                            <1> loc_gmcs_init_drv_hd:
00008187 08E4               <1>       or     ah, ah
00008189 7404               <1>       jz     short loc_change_current_drv3
                            <1>       ; BL = 0, BH = Logical DOS drive number
                            <1> loc_change_current_drv2:
0000818B C6467E00           <1>       mov    byte [esi+LD_MediaChanged], 0
                            <1> loc_change_current_drv3:
0000818F 883D[868A0100]     <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>
00008195 8A4603                 <1>       mov    al, [esi+LD_FATType]
00008198 A2[858A0100]           <1>       mov    [Current_FATType], al
                                <1>
0000819D 8A26                   <1>       mov    ah, [esi+LD_Name]
0000819F 8825[878A0100]         <1>       mov    [Current_Dir_Drv], ah
                                <1>
000081A5 20C0                   <1>       and    al, al
000081A7 741D                   <1>       jz     short loc_restore_FS_current_directory
                                <1>
                                <1> loc_restore_FAT_current_directory:
000081A9 8A667F                 <1>       mov    ah, [esi+LD_CDirLevel]
000081AC 8825[848A0100]         <1>       mov    [Current_Dir_Level], ah
000081B2 08E4                   <1>       or     ah, ah
000081B4 7416                   <1>         jz   short loc_ccdrv_reset_cdir_FAT_12_16_32_fcluster
                                <1>
000081B6 0FB6D4                 <1>       movzx  edx, ah
000081B9 C0E204                 <1>       shl    dl, 4 ; * 16
000081BC 01F2                   <1>         add  edx, esi
000081BE 8B828C000000           <1>       mov    eax, [edx+LD_CurrentDirectory+12]
000081C4 EB2C                   <1>       jmp    short loc_ccdrv_reset_cdir_FAT_fcluster
                                <1>
                                <1> loc_restore_FS_current_directory:
000081C6 E8BC4D0000             <1>       call   load_current_FS_directory
000081CB C3                     <1>       retn
                                <1>
                                <1> loc_ccdrv_reset_cdir_FAT_12_16_32_fcluster:
000081CC 3C03                   <1>       cmp    al, 3
000081CE 7205                   <1>       jb     short loc_ccdrv_reset_cdir_FAT_12_16_fcluster
                                <1>
                                <1> loc_ccdrv_reset_cdir_FAT32_fcluster:
000081D0 8B4632                 <1>       mov    eax, [esi+LD_BPB+FAT32_RootFClust]
000081D3 EB04                   <1>       jmp    short loc_ccdrv_check_rootdir_sign
                                <1> loc_ccdrv_reset_cdir_FAT_12_16_fcluster:
000081D5 30C0                   <1>       xor    al, al  ; xor eax, eax
000081D7 31D2                   <1>       xor    edx, edx
                                <1> loc_ccdrv_check_rootdir_sign:
000081D9 80BE8000000000         <1>       cmp    byte [esi+LD_CurrentDirectory], 0
000081E0 7510                   <1>       jne    short loc_ccdrv_reset_cdir_FAT_fcluster
                                <1> loc_ccdrv_set_rootdir_FAT_fcluster:
000081E2 89868C000000           <1>         mov     [esi+LD_CurrentDirectory+12], eax
000081E8 C786800000000524F4F-   <1>       mov    dword [esi+LD_CurrentDirectory], 'ROOT'
000081F1 54                     <1>
                                <1>
                                <1> loc_ccdrv_reset_cdir_FAT_fcluster:
000081F2 A3[808A0100]           <1>       mov    [Current_Dir_FCluster], eax
                                <1>
000081F7 BF[E3910100]           <1>       mov    edi, PATH_Array
000081FC 89F2                   <1>       mov    edx, esi
000081FE 81C680000000           <1>       add    esi, LD_CurrentDirectory
00008204 B920000000             <1>       mov    ecx, 32
00008209 F3A5                   <1>       rep    movsd
                                <1>
0000820B E84C2D0000             <1>       call   change_prompt_dir_string
                                <1>
00008210 89D6                   <1>       mov    esi, edx
                                <1>
00008212 29C0                   <1>         sub  eax, eax
                                <1>        ;sub  edx, edx
00008214 BF[878A0100]           <1>       mov    edi, Current_Dir_Drv
                                <1>
00008219 A2[51400100]           <1>       mov    [Restore_CDIR], al ; 0
0000821E 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
0000821F C705[40960100]-        <1>       mov    dword [mainprog_return_addr], return_from_cmd_interpreter
00008225 [D3820000]             <1>
                                <1>
                                <1> loc_TRDOS_prompt:
00008229 BF[868B0100]           <1>       mov    edi, TextBuffer
0000822E C6075B                 <1>       mov    byte [edi], "["
00008231 47                     <1>       inc    edi
00008232 BE[A4400100]           <1>       mov    esi, TRDOSPromptLabel
                                <1> get_next_prompt_label_char:
00008237 803E20                 <1>       cmp    byte [esi], 20h
0000823A 7203                   <1>       jb     short pass_prompt_label
0000823C A4                     <1>       movsb
0000823D EBF8                   <1>       jmp    short get_next_prompt_label_char
                                <1> pass_prompt_label:
0000823F C6075D                 <1>       mov    byte [edi], "]"
00008242 47                     <1>       inc    edi
00008243 C60720                 <1>       mov    byte [edi], 20h
00008246 47                     <1>       inc    edi
00008247 BE[878A0100]           <1>       mov    esi, Current_Dir_Drv
0000824C 66A5                   <1>       movsw
0000824E A4                     <1>       movsb
                                <1> loc_prompt_current_directory:
0000824F 803E20                 <1>       cmp    byte [esi], 20h
00008252 7203                   <1>       jb     short pass_prompt_current_directory
00008254 A4                     <1>       movsb
00008255 EBF8                   <1>       jmp    short loc_prompt_current_directory
                                <1> pass_prompt_current_directory:
00008257 C6073E                 <1>       mov    byte [edi], '>'
```

```
0000825A 47                <1>       inc    edi
0000825B C60700            <1>       mov    byte [edi], 0
0000825E BE[868B0100]      <1>       mov    esi, TextBuffer
00008263 E8EDF2FFFF        <1>       call   print_msg
                           <1>
                           <1>       ;sub   bh, bh ; video page = 0
                           <1>       ;call  get_cpos ; get cursor position
00008268 668B15[DE890100]  <1>       mov    dx, [CURSOR_POSN] ; video page 0
0000826F 8815[E68A0100]    <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:
00008275 E899000000        <1>       call   rw_char
                           <1> loc_move_command:
0000827A BE[368B0100]      <1>       mov    esi, CommandBuffer
0000827F 89F7              <1>       mov    edi, esi
00008281 31C9              <1>       xor    ecx, ecx
                           <1> first_command_char:
00008283 AC                <1>       lodsb
00008284 3C20              <1>       cmp    al, 20h
00008286 772E              <1>       ja     short pass_space_control
00008288 7241              <1>       jb     short loc_move_cmd_arguments_ok
0000828A 81FE[858B0100]    <1>       cmp    esi, CommandBuffer + 79
00008290 72F1              <1>       jb     short first_command_char
00008292 EB37              <1>       jmp    short loc_move_cmd_arguments_ok
                           <1>
                           <1> next_command_char:
00008294 AC                <1>       lodsb
00008295 3C20              <1>       cmp    al, 20h
00008297 771D              <1>       ja     short pass_space_control
00008299 7230              <1>       jb     short loc_move_cmd_arguments_ok
                           <1>
                           <1> loc_1st_cmd_arg: ; 30/01/2016
0000829B AC                <1>       lodsb
0000829C 3C20              <1>       cmp    al, 20h
0000829E 74FB              <1>       je     short loc_1st_cmd_arg
000082A0 7229              <1>       jb     short loc_move_cmd_arguments_ok
                           <1>
000082A2 C60700            <1>         mov     byte [edi], 0
000082A5 47                <1>       inc    edi
                           <1>
                           <1> loc_move_cmd_arguments:
000082A6 AA                <1>       stosb
000082A7 81FE[858B0100]    <1>       cmp    esi, CommandBuffer + 79
000082AD 731C              <1>       jnb    short loc_move_cmd_arguments_ok
000082AF AC                <1>         lodsb
000082B0 3C20              <1>       cmp    al, 20h
000082B2 73F2              <1>       jnb    short loc_move_cmd_arguments
000082B4 EB15              <1>       jmp    short loc_move_cmd_arguments_ok
                           <1>
                           <1> pass_space_control:
000082B6 3C61              <1>       cmp    al, 61h
000082B8 7206              <1>       jb     short pass_capitalize
000082BA 3C7A              <1>       cmp    al, 7Ah
000082BC 7702              <1>       ja     short pass_capitalize
000082BE 24DF              <1>       and    al, 0DFh
                           <1> pass_capitalize:
000082C0 AA                <1>       stosb
000082C1 FEC1              <1>       inc     cl
000082C3 81FE[858B0100]    <1>         cmp     esi, CommandBuffer + 79
000082C9 72C9              <1>       jb      short next_command_char
                           <1>
                           <1> loc_move_cmd_arguments_ok:
000082CB C60700            <1>         mov     byte [edi], 0
                           <1>
                           <1> call_command_interpreter:
000082CE E8CF080000        <1>       call    command_interpreter
                           <1>
                           <1> return_from_cmd_interpreter:
000082D3 B950000000        <1>         mov ecx, 80
                           <1>       ;mov   cx, 80
000082D8 BF[368B0100]      <1>       mov    edi, CommandBuffer
000082DD 30C0              <1>       xor    al, al
000082DF F3AA              <1>       rep    stosb
                           <1>       ;cmp   byte [Program_Exit], 0
                           <1>       ;ja    short loc_terminate_trdos
                           <1>
                           <1>       ; 16/01/2016
000082E1 803D[CA6F0000]03  <1>       cmp    byte [CRT_MODE], 3 ; 80*25 color
000082E8 741D              <1>       je     short pass_set_txt_mode
                           <1>
000082EA E87798FFFF        <1>       call   set_txt_mode ; set vide mode to 03h
                           <1>       ; 07/01/2017
000082EF 30C0              <1>       xor    al, al
                           <1>
                           <1> loc_check_active_page:
                           <1>       ;xor   al, al
000082F1 3805[EE890100]    <1>       cmp    [ACTIVE_PAGE], al ; 0
000082F7 0F842CFFFFFF      <1>         je      loc_TRDOS_prompt
                           <1>       ; AL = 0 = video page 0
```

```
000082FD E8AD9CFFFF          <1>       call   set_active_page
00008302 E922FFFFFF          <1>         jmp     loc_TRDOS_prompt ; infinitive loop
                             <1>
                             <1> pass_set_txt_mode:
00008307 BE[EF4C0100]        <1>       mov    esi, nextline
0000830C E844F2FFFF          <1>       call   print_msg
00008311 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:
00008313 30E4                <1>       xor     ah, ah
00008315 E8CB8BFFFF          <1>       call   int16h
0000831A 20C0                <1>       and    al, al
0000831C 7432                <1>       jz     short loc_arrow
0000831E 3CE0                <1>       cmp    al, 0E0h
00008320 742E                <1>       je     short loc_arrow
00008322 3C08                <1>       cmp    al, 08h
00008324 7542                <1>       jne    short char_return
                             <1> loc_back:
00008326 3A15[E68A0100]      <1>       cmp    dl, [CursorColumn]
0000832C 76E5                <1>       jna     short readnextchar
                             <1> prev_column:
0000832E FECA                <1>       dec    dl
                             <1> set_cursor_pos:
                             <1>       ;push  dx
00008330 52                  <1>       push   edx ; 29/12/2017
                             <1>       ;xor   bh, bh ; 0 = video page 0
                             <1>       ; DH = Row, DL = Column
00008331 E861A0FFFF          <1>       call   _set_cpos ; 17/01/2016
00008336 5A                  <1>         pop  edx ; 29/12/2017
                             <1>       ;pop   dx
                             <1>       ;movzx ebx, dl
00008337 88D3                <1>       mov    bl, dl
00008339 2A1D[E68A0100]      <1>       sub    bl, [CursorColumn]
0000833F B020                <1>       mov    al, 20h
00008341 8883[368B0100]      <1>       mov    [CommandBuffer+ebx], al
                             <1>       ;sub   bh, bh ; video page 0
                             <1>       ;mov   cx, 1
00008347 B307                <1>       mov    bl, 7 ; color attribute
00008349 E8289FFFFF          <1>       call   _write_c_current ; 17/01/2016
                             <1>       ;mov   dx, [CURSOR_POSN]
0000834E EBC3                <1>       jmp    short readnextchar
                             <1> loc_arrow:
00008350 80FC4B              <1>       cmp    ah, 4Bh
00008353 74D1                <1>       je     short loc_back
00008355 80FC53              <1>       cmp    ah, 53h
00008358 74CC                <1>       je      short loc_back
0000835A 80FC4D              <1>       cmp    ah, 4Dh
0000835D 75B4                <1>       jne    short readnextchar
0000835F 80FA4F              <1>       cmp    dl, 79
00008362 73AF                <1>       jnb    short readnextchar
00008364 FEC2                <1>       inc    dl
00008366 EBC8                <1>       jmp    short set_cursor_pos
                             <1> char_return:
00008368 0FB6DA              <1>       movzx  ebx, dl
0000836B 2A1D[E68A0100]      <1>       sub    bl, [CursorColumn]
00008371 3C20                <1>       cmp    al, 20h
00008373 721D                <1>       jb     short loc_escape
00008375 8883[368B0100]      <1>       mov    [CommandBuffer+ebx], al
0000837B 80FA4F              <1>       cmp    dl, 79
0000837E 7393                <1>       jnb    short readnextchar
00008380 66BB0700            <1>       mov    bx, 7 ; color attribute
00008384 E8749FFFFF          <1>       call   _write_tty
00008389 668B15[DE890100]    <1>       mov    dx, [CURSOR_POSN] ; video page 0
00008390 EB81                <1>         jmp     readnextchar
                             <1> loc_escape:
00008392 3C1B                <1>       cmp    al, 1Bh
00008394 7418                <1>       je     short rw_char_retn
                             <1>       ;
00008396 3C0D                <1>       cmp    al, 0Dh ; CR
00008398 0F8575FFFFFF        <1>         jne     readnextchar
                             <1>       ; 13/05/2016
0000839E 66BB0700            <1>       mov    bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh=0)
000083A2 E8569FFFFF          <1>       call   _write_tty
                             <1>       ;mov   bx, 7  ; attribute/color
                             <1>                    ; video page 0 (bh=0)
000083A7 B00A                <1>       mov    al, 0Ah ; LF
000083A9 E84F9FFFFF          <1>       call   _write_tty
                             <1> rw_char_retn:
000083AE 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
000083AF E867E7FFFF          <1>       call   RTC_40 ; GET RTC DATE
                             <1>
000083B4 88D0                <1>       mov    al, dl
000083B6 E81D8BFFFF          <1>       call   bcd_to_ascii
000083BB 66A3[90410100]      <1>       mov    [Day], ax
                             <1>
```

```
000083C1 88F0                <1>       mov    al, dh
000083C3 E8108BFFFF          <1>       call   bcd_to_ascii
000083C8 66A3[93410100]      <1>       mov    [Month], ax
                             <1>
000083CE 88E8                <1>       mov    al, ch
000083D0 E8038BFFFF          <1>       call   bcd_to_ascii
000083D5 66A3[96410100]      <1>       mov    [Century], ax
                             <1>
000083DB 88C8                <1>       mov    al, cl
000083DD E8F68AFFFF          <1>       call   bcd_to_ascii
000083E2 66A3[98410100]      <1>       mov    word [Year], ax
                             <1>
000083E8 BE[80410100]        <1>       mov    esi, Msg_Show_Date
000083ED E863F1FFFF          <1>       call   print_msg
                             <1>
000083F2 C3                  <1>       retn
                             <1>
                             <1> set_date:
                             <1>       ; 13/05/2016
                             <1>       ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>         ; 2004-2005
                             <1>
000083F3 BE[64410100]        <1>       mov    esi, Msg_Enter_Date
000083F8 E858F1FFFF          <1>       call   print_msg
                             <1>
                             <1> loc_enter_day_1:
000083FD 30E4                <1>       xor     ah, ah
000083FF E8E18AFFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
00008404 3C0D                <1>       cmp    al, 13
00008406 0F84B7010000        <1>       je     loc_set_date_retn
0000840C 3C1B                <1>       cmp    al, 27
0000840E 0F84AF010000        <1>       je     loc_set_date_retn
00008414 A2[90410100]        <1>       mov    [Day], al
00008419 3C30                <1>       cmp    al, '0'
0000841B 0F82AD010000        <1>       jb     loc_set_date_stc_0
00008421 3C33                <1>       cmp    al, '3'
00008423 0F87A5010000        <1>       ja     loc_set_date_stc_0
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
00008429 B307                <1>       mov    bl, 7
0000842B E8CD9EFFFF          <1>       call   _write_tty
                             <1> loc_enter_day_2:
00008430 30E4                <1>       xor     ah, ah
00008432 E8AE8AFFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
00008437 3C1B                <1>       cmp    al, 27
00008439 0F8484010000        <1>         je       loc_set_date_retn
0000843F A2[91410100]        <1>       mov    [Day+1], al
00008444 3C30                <1>       cmp    al, '0'
00008446 0F828C010000        <1>         jb       loc_set_date_stc_1
0000844C 3C39                <1>       cmp    al, '9'
0000844E 0F8784010000        <1>         ja       loc_set_date_stc_1
00008454 803D[90410100]33    <1>       cmp    byte [Day], '3'
0000845B 7208                <1>       jb     short pass_set_day_31
0000845D 3C31                <1>       cmp    al, '1'
0000845F 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)
00008465 B307                <1>       mov    bl, 7
00008467 E8919EFFFF          <1>       call   _write_tty
                             <1> loc_enter_separator_1:
0000846C 28E4                <1>       sub     ah, ah ; 0
0000846E E8728AFFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
00008473 3C1B                <1>       cmp    al, 27
00008475 0F8448010000        <1>         je       loc_set_date_retn
0000847B 3C2D                <1>       cmp    al, '-'
0000847D 7408                <1>       je     short pass_set_date_separator_1
0000847F 3C2F                <1>       cmp    al, '/'
00008481 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)
00008487 B307                <1>       mov    bl, 7
00008489 E86F9EFFFF          <1>       call   _write_tty
                             <1> loc_enter_month_1:
0000848E 30E4                <1>       xor     ah, ah ; 0
00008490 E8508AFFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
00008495 3C1B                <1>       cmp    al, 27
00008497 0F8426010000        <1>         je       loc_set_date_retn
0000849D A2[93410100]        <1>       mov    [Month], al
000084A2 3C30                <1>       cmp    al, '0'
000084A4 0F8264010000        <1>         jb       loc_set_date_stc_3
000084AA 3C31                <1>       cmp    al, '1'
000084AC 0F875C010000        <1>         ja       loc_set_date_stc_3
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
000084B2 B307                <1>       mov    bl, 7
000084B4 E8449EFFFF          <1>       call   _write_tty
                             <1> loc_enter_month_2:
000084B9 30E4                <1>       xor     ah, ah
000084BB E8258AFFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
000084C0 3C1B                <1>       cmp    al, 27
000084C2 0F84FB000000        <1>         je       loc_set_date_retn
000084C8 A2[94410100]        <1>       mov    [Month+1], al
000084CD 3C30                <1>       cmp    al, '0'
000084CF 0F8254010000        <1>         jb       loc_set_date_stc_4
```

```
000084D5 3C39               <1>       cmp    al, '9'
000084D7 0F874C010000       <1>         ja      loc_set_date_stc_4
000084DD 803D[93410100]31   <1>       cmp    byte [Month], '1'
000084E4 7208               <1>       jb     short pass_set_month_12
000084E6 3C32               <1>       cmp    al, '2'
000084E8 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)
000084EE B307               <1>       mov    bl, 7
000084F0 E8089EFFFF         <1>       call   _write_tty
                            <1> loc_enter_separator_2:
000084F5 28E4               <1>       sub     ah, ah
000084F7 E8E989FFFF         <1>       call   int16h
                            <1>       ; AL = ASCII Code of the Character
000084FC 3C1B               <1>       cmp    al, 27
000084FE 0F84BF000000       <1>         je      loc_set_date_retn
00008504 3C2D               <1>       cmp    al, '-'
00008506 7408               <1>       je     short pass_set_date_separator_2
00008508 3C2F               <1>       cmp    al, '/'
0000850A 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)
00008510 B307               <1>       mov    bl, 7
00008512 E8E69DFFFF         <1>       call   _write_tty
                            <1> loc_enter_year_1:
00008517 30E4               <1>       xor     ah, ah
00008519 E8C789FFFF         <1>       call   int16h
                            <1>       ; AL = ASCII Code of the Character
0000851E 3C1B               <1>       cmp    al, 27
00008520 0F849D000000       <1>         je      loc_set_date_retn
00008526 A2[98410100]       <1>       mov    [Year], al
0000852B 3C30               <1>       cmp    al, '0'
0000852D 0F822C010000       <1>         jb      loc_set_date_stc_6
00008533 3C39               <1>       cmp    al, '9'
00008535 0F8724010000       <1>         ja      loc_set_date_stc_6
                            <1>       ; 13/05/2016
                            <1>       ;mov   bx, 7 ; attribute/color (bl)
                            <1>                    ; video page 0 (bh)
0000853B B307               <1>       mov    bl, 7
0000853D E8BB9DFFFF         <1>       call   _write_tty
                            <1> loc_enter_year_2:
00008542 30E4               <1>       xor    ah, ah
00008544 E89C89FFFF         <1>       call   int16h
                            <1>       ; AL = ASCII Code of the Character
00008549 3C1B               <1>       cmp    al, 27
0000854B 7476               <1>       je     short loc_set_date_retn
0000854D A2[99410100]       <1>       mov    byte [Year+1], al
00008552 3C30               <1>       cmp    al, '0'
00008554 0F8220010000       <1>         jb      loc_set_date_stc_7
0000855A 3C39               <1>       cmp    al, '9'
0000855C 0F8718010000       <1>         ja      loc_set_date_stc_7
                            <1>       ; 13/05/2016
                            <1>       ;mov   bx, 7 ; attribute/color (bl)
                            <1>                      ; video page 0 (bh)
00008562 B307               <1>       mov    bl, 7
00008564 E8949DFFFF         <1>       call   _write_tty
                            <1> loc_set_date_get_lchar_again:
00008569 28E4               <1>       sub    ah, ah ; 0
0000856B E87589FFFF         <1>       call   int16h
                            <1>       ; AL = ASCII Code of the Character
00008570 3C0D               <1>       cmp    al, 13 ; ENTER key
00008572 7412               <1>       je     short loc_set_date_progress
00008574 3C1B               <1>       cmp    al, 27 ; ESC key
00008576 744B               <1>       je     short loc_set_date_retn
                            <1>       ;
00008578 E82A010000         <1>       call   check_for_backspace
0000857D 75EA               <1>       jne    short loc_set_date_get_lchar_again
                            <1>
                            <1> loc_set_date_bs_8:
0000857F E811010000         <1>       call   write_backspace
00008584 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
00008586 E890E5FFFF         <1>       call   RTC_40 ; GET RTC DATE
                            <1>       ; CH = century (in BCD)
                            <1>
0000858B 66A1[98410100]     <1>       mov    ax, [Year]
00008591 662D3030           <1>       sub    ax, '00'
00008595 C0E004             <1>       shl    al, 4 ; * 16
00008598 88C1               <1>       mov    cl, al
0000859A 00E1               <1>       add    cl, ah
0000859C 66A1[93410100]     <1>       mov    ax, [Month]
000085A2 662D3030           <1>       sub    ax, '00'
000085A6 C0E004             <1>       shl    al, 4 ; * 16
000085A9 88C6               <1>       mov    dh, al
000085AB 00E6               <1>       add    dh, ah
000085AD 66A1[90410100]     <1>       mov    ax, [Day]
000085B3 662D3030           <1>       sub    ax, '00'
000085B7 C0E004             <1>       shl    al, 4 ; * 16
000085BA 88C2               <1>       mov    dl, al
000085BC 00E2               <1>       add    dl, ah
                            <1>
                            <1>       ;mov   ah, 05h
                            <1>       ;call  int1Ah
000085BE E885E5FFFF         <1>       call   RTC_50 ; SET RTC DATE
                            <1>
                            <1> loc_set_date_retn:
000085C3 BE[EF4C0100]       <1>       mov    esi, nextline
```

```
000085C8 E888EFFFFF          <1>       call   print_msg
000085CD C3                  <1>       retn
                             <1>
                             <1> loc_set_date_stc_0:
                             <1>       ;xor   bh, bh ; video page 0
000085CE E8169EFFFF          <1>       call   beeper ; BEEP !
000085D3 E925FEFFFF          <1>         jmp     loc_enter_day_1
                             <1> loc_set_date_stc_1:
000085D8 E8CA000000          <1>       call   check_for_backspace
000085DD 740A                <1>       je     short loc_set_date_bs_1
                             <1>       ;xor   bh, bh ; video page 0
000085DF E8059EFFFF          <1>       call   beeper ; BEEP !
000085E4 E947FEFFFF          <1>         jmp     loc_enter_day_2
                             <1> loc_set_date_bs_1:
000085E9 E8A7000000          <1>       call   write_backspace
000085EE E90AFEFFFF          <1>         jmp     loc_enter_day_1
                             <1> loc_set_date_stc_2:
000085F3 E8AF000000          <1>       call   check_for_backspace
000085F8 740A                <1>       je     short loc_set_date_bs_2
                             <1>       ;xor   bh, bh ; video page 0
000085FA E8EA9DFFFF          <1>       call   beeper ; BEEP !
000085FF E968FEFFFF          <1>         jmp     loc_enter_separator_1
                             <1> loc_set_date_bs_2:
00008604 E88C000000          <1>       call   write_backspace
00008609 E922FEFFFF          <1>         jmp     loc_enter_day_2
                             <1> loc_set_date_stc_3:
0000860E E894000000          <1>       call   check_for_backspace
00008613 740A                <1>       je short loc_set_date_bs_3
                             <1>       ;xor   bh, bh ; video page 0
00008615 E8CF9DFFFF          <1>       call   beeper ; BEEP !
0000861A E96FFEFFFF          <1>         jmp     loc_enter_month_1
                             <1> loc_set_date_bs_3:
0000861F E871000000          <1>       call   write_backspace
00008624 E943FEFFFF          <1>         jmp     loc_enter_separator_1
                             <1> loc_set_date_stc_4:
00008629 E879000000          <1>       call   check_for_backspace
0000862E 740A                <1>       je     short loc_set_date_bs_4
                             <1>       ;xor   bh, bh ; video page 0
00008630 E8B49DFFFF          <1>       call   beeper ; BEEP !
00008635 E97FFEFFFF          <1>         jmp     loc_enter_month_2
                             <1> loc_set_date_bs_4:
0000863A E856000000          <1>       call   write_backspace
0000863F E94AFEFFFF          <1>         jmp     loc_enter_month_1
                             <1> loc_set_date_stc_5:
00008644 E85E000000          <1>       call   check_for_backspace
00008649 740A                <1>       je     short loc_set_date_bs_5
                             <1>       ;xor   bh, bh ; video page 0
0000864B E8999DFFFF          <1>       call   beeper ; BEEP !
00008650 E9A0FEFFFF          <1>         jmp     loc_enter_separator_2
                             <1> loc_set_date_bs_5:
00008655 E83B000000          <1>       call   write_backspace
0000865A E95AFEFFFF          <1>         jmp     loc_enter_month_2
                             <1> loc_set_date_stc_6:
0000865F E843000000          <1>       call   check_for_backspace
00008664 740A                <1>         je      short  loc_set_date_bs_6
                             <1>       ;xor   bh, bh ; video page 0
00008666 E87E9DFFFF          <1>       call   beeper ; BEEP !
0000866B E9A7FEFFFF          <1>         jmp     loc_enter_year_1
                             <1> loc_set_date_bs_6:
00008670 E820000000          <1>       call   write_backspace
00008675 E97BFEFFFF          <1>         jmp     loc_enter_separator_2
                             <1> loc_set_date_stc_7:
0000867A E828000000          <1>       call   check_for_backspace
0000867F 740A                <1>       je     short loc_set_date_bs_7
                             <1>       ;xor   bh, bh ; video page 0
00008681 E8639DFFFF          <1>       call   beeper ; BEEP !
00008686 E9B7FEFFFF          <1>         jmp     loc_enter_year_2
                             <1> loc_set_date_bs_7:
0000868B E805000000          <1>       call   write_backspace
00008690 E982FEFFFF          <1>         jmp     loc_enter_year_1
                             <1>
                             <1> write_backspace:
                             <1>       ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
00008695 B008                <1>       mov    al, 08h ; BACKSPACE
                             <1>       ; 13/05/2016
00008697 66BB0700            <1>       mov    bx, 7 ; bl = attribute/color
                             <1>                    ; bh = video page = 0
0000869B E85D9CFFFF          <1>       call   _write_tty
000086A0 B020                <1>       mov    al, 20h ; BLANK/SPACE char
                             <1>       ;mov   bx, 7 ; attribute/color
                             <1>       ;call  _write_c_current
                             <1>       ;retn
000086A2 E9CF9BFFFF          <1>       jmp    _write_c_current
                             <1>
                             <1> check_for_backspace:
                             <1>       ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
000086A7 663D080E            <1>       cmp    ax, 0E08h
000086AB 7410                <1>       je     short cfbs_retn
000086AD 663DE04B            <1>       cmp    ax, 4BE0h
000086B1 740A                <1>       je     short cfbs_retn
000086B3 663D004B            <1>       cmp    ax, 4B00h
000086B7 7404                <1>       je     short cfbs_retn
000086B9 663DE053            <1>       cmp    ax, 53E0h
                             <1> cfbs_retn:
000086BD 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
000086BE E8E7E3FFFF          <1>       call   RTC_20 ; GET RTC TIME
                             <1>
000086C3 88E8                <1>       mov    al, ch
```

```
000086C5 E80E88FFFF          <1>       call   bcd_to_ascii
000086CA 66A3[BE410100]      <1>       mov    [Hour], ax
                             <1>
000086D0 88C8                <1>       mov    al, cl
000086D2 E80188FFFF          <1>       call   bcd_to_ascii
000086D7 66A3[C1410100]      <1>       mov    [Minute], ax
                             <1>
000086DD 88F0                <1>       mov    al, dh
000086DF E8F487FFFF          <1>       call   bcd_to_ascii
000086E4 66A3[C4410100]      <1>       mov    [Second], ax
                             <1>
000086EA BE[AE410100]        <1>       mov    esi, Msg_Show_Time
000086EF E861EEFFFF          <1>       call   print_msg
000086F4 C3                  <1>       retn
                             <1>
                             <1> set_time:
                             <1>       ; 13/05/2016
                             <1>       ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>         ; 2004-2005
                             <1>
000086F5 BE[9D410100]        <1>       mov    esi, Msg_Enter_Time
000086FA E856EEFFFF          <1>       call   print_msg
                             <1>
                             <1> loc_enter_hour_1:
000086FF 30E4                <1>       xor     ah, ah
00008701 E8DF87FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
00008706 3C0D                <1>       cmp    al, 13 ; ENTER key
00008708 0F84AE010000        <1>         je      loc_set_time_retn
0000870E 3C1B                <1>       cmp    al, 27 ; ESC key
00008710 0F84A6010000        <1>         je      loc_set_time_retn
00008716 A2[BE410100]        <1>       mov    [Hour], al
0000871B 3C30                <1>       cmp    al, '0'
0000871D 0F82A4010000        <1>         jb      loc_set_time_stc_0
00008723 3C32                <1>       cmp    al, '2'
00008725 0F879C010000        <1>         ja      loc_set_time_stc_0
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
0000872B B307                <1>       mov    bl, 7
0000872D E8CB9BFFFF          <1>       call   _write_tty
                             <1> loc_enter_hour_2:
00008732 30E4                <1>       xor     ah, ah
00008734 E8AC87FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
00008739 3C1B                <1>       cmp    al, 27
0000873B 0F847B010000        <1>         je      loc_set_time_retn
00008741 A2[BF410100]        <1>       mov    [Hour+1], al
00008746 3C30                <1>       cmp    al, '0'
00008748 0F8283010000        <1>         jb      loc_set_time_stc_1
0000874E 3C39                <1>       cmp    al, '9'
00008750 0F877B010000        <1>       ja     loc_set_time_stc_1
00008756 803D[BE410100]32    <1>         cmp     byte [Hour], '2'
0000875D 7208                <1>       jb     short pass_set_time_24
0000875F 3C34                <1>       cmp    al, '4'
00008761 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)
00008767 B307                <1>       mov    bl, 7
00008769 E88F9BFFFF          <1>       call   _write_tty
                             <1> loc_enter_time_separator_1:
0000876E 28E4                <1>       sub    ah, ah ; 0
00008770 E87087FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
00008775 3C1B                <1>       cmp    al, 27
00008777 0F843F010000        <1>         je      loc_set_time_retn
0000877D 3C3A                <1>       cmp    al, ':'
0000877F 0F8567010000        <1>         jne     loc_set_time_stc_2
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
00008785 B307                <1>       mov    bl, 7
00008787 E8719BFFFF          <1>       call   _write_tty
                             <1> loc_enter_minute_1:
0000878C 30E4                <1>       xor     ah, ah
0000878E E85287FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
00008793 3C1B                <1>       cmp    al, 27
00008795 0F8421010000        <1>         je      loc_set_time_retn
0000879B A2[C1410100]        <1>       mov    [Minute], al
000087A0 3C30                <1>       cmp    al, '0'
000087A2 0F825F010000        <1>         jb      loc_set_time_stc_3
000087A8 3C35                <1>       cmp    al, '5'
000087AA 0F8757010000        <1>         ja      loc_set_time_stc_3
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
000087B0 B307                <1>       mov    bl, 7
000087B2 E8469BFFFF          <1>       call   _write_tty
                             <1> loc_enter_minute_2:
000087B7 30E4                <1>       xor     ah, ah
000087B9 E82787FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
000087BE 3C1B                <1>       cmp    al, 27
000087C0 0F84F6000000        <1>         je      loc_set_time_retn
000087C6 A2[C2410100]        <1>       mov    [Minute+1], al
000087CB 3C30                <1>       cmp    al, '0'
000087CD 0F824F010000        <1>         jb      loc_set_time_stc_4
000087D3 3C39                <1>       cmp    al, '9'
000087D5 0F8747010000        <1>         ja      loc_set_time_stc_4
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
```

```
000087DB B307                <1>       mov    bl, 7
000087DD E81B9BFFFF          <1>       call   _write_tty
                             <1> loc_enter_time_separator_2:
000087E2 66C705[C4410100]30- <1>       mov    word [Second], 3030h
000087EA 30                  <1>
000087EB 28E4                <1>       sub     ah, ah
000087ED E8F386FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
000087F2 3C0D                <1>       cmp    al, 13
000087F4 0F8485000000        <1>         je      loc_set_time_progress
000087FA 3C1B                <1>       cmp    al, 27
000087FC 0F84BA000000        <1>         je      loc_set_time_retn
00008802 3C3A                <1>       cmp    al, ':'
00008804 0F8533010000        <1>         jne     loc_set_time_stc_5
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
0000880A B307                <1>       mov    bl, 7
0000880C E8EC9AFFFF          <1>       call   _write_tty
                             <1> loc_enter_second_1:
00008811 30E4                <1>       xor     ah, ah
00008813 E8CD86FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
00008818 3C0D                <1>       cmp    al, 13
0000881A 7463                <1>       je     short loc_set_time_progress
0000881C 3C1B                <1>       cmp    al, 27
0000881E 0F8498000000        <1>         je      loc_set_time_retn
00008824 A2[C4410100]        <1>       mov    [Second], al
00008829 3C30                <1>       cmp    al, '0'
0000882B 0F8227010000        <1>         jb      loc_set_time_stc_6
00008831 3C35                <1>       cmp    al, '5'
00008833 0F871F010000        <1>         ja      loc_set_time_stc_6
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
00008839 B307                <1>       mov    bl, 7
0000883B E8BD9AFFFF          <1>       call   _write_tty
                             <1> loc_enter_second_2:
00008840 30E4                <1>       xor     ah, ah
00008842 E89E86FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
00008847 3C1B                <1>       cmp    al, 27
00008849 7471                <1>       je     short loc_set_time_retn
0000884B 3C30                <1>       cmp    al, '0'
0000884D 0F8229010000        <1>         jb      loc_set_time_stc_7
00008853 3C39                <1>       cmp    al, '9'
00008855 0F8721010000        <1>         ja      loc_set_time_stc_7
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
0000885B B307                <1>       mov    bl, 7
0000885D E89B9AFFFF          <1>       call   _write_tty
                             <1> loc_set_time_get_lchar_again:
00008862 28E4                <1>       sub    ah, ah ; 0
00008864 E87C86FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
00008869 3C0D                <1>       cmp    al, 13
0000886B 7412                <1>       je     short loc_set_time_progress
0000886D 3C1B                <1>       cmp    al, 27
0000886F 744B                <1>       je     short loc_set_time_retn
                             <1>       ;
00008871 E831FEFFFF          <1>       call   check_for_backspace
00008876 75EA                <1>       jne    short loc_set_time_get_lchar_again
                             <1>
                             <1> loc_set_time_bs_8:
00008878 E818FEFFFF          <1>       call   write_backspace
0000887D 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
0000887F E826E2FFFF          <1>       call   RTC_20 ; GET RTC TIME
                             <1>       ;DL = Daylight Savings Enable option (0-1)
                             <1>
00008884 66A1[BE410100]      <1>       mov    ax, [Hour]
0000888A 662D3030            <1>       sub    ax, '00'
0000888E C0E004              <1>       shl    al, 4 ; * 16
00008891 88C5                <1>       mov    ch, al
00008893 00E5                <1>       add    ch, ah
00008895 66A1[C1410100]      <1>       mov    ax, [Minute]
0000889B 662D3030            <1>       sub    ax, '00'
0000889F C0E004              <1>       shl    al, 4 ; * 16
000088A2 88C1                <1>       mov    cl, al
000088A4 00E1                <1>       add    cl, ah
000088A6 66A1[C4410100]      <1>       mov    ax, [Second]
000088AC 662D3030            <1>       sub    ax, '00'
000088B0 C0E004              <1>       shl    al, 4 ; * 16
000088B3 88C6                <1>       mov    dh, al
000088B5 00E6                <1>       add    dh, ah
                             <1>
                             <1>       ;mov   ah, 03h
                             <1>       ;call  int1Ah
000088B7 E81DE2FFFF          <1>       call   RTC_30 ; SET RTC TIME
                             <1>
                             <1> loc_set_time_retn:
000088BC BE[EF4C0100]        <1>       mov    esi, nextline
000088C1 E88FECFFFF          <1>       call   print_msg
000088C6 C3                  <1>       retn
                             <1>
                             <1> loc_set_time_stc_0:
                             <1>       ;xor   bh, bh ; video page 0
000088C7 E81D9BFFFF          <1>       call   beeper ; BEEP !
000088CC E92EFEFFFF          <1>         jmp     loc_enter_hour_1
                             <1> loc_set_time_stc_1:
```

```
1003 000088D1 E8D1FDFFFF          <1>       call   check_for_backspace
1004 000088D6 740A                <1>       je     short loc_set_time_bs_1
1005                              <1>       ;xor   bh, bh ; video page 0
1006 000088D8 E80C9BFFFF          <1>       call   beeper ; BEEP !
1007 000088DD E950FEFFFF          <1>         jmp     loc_enter_hour_2
1008                              <1> loc_set_time_bs_1:
1009 000088E2 E8AEFDFFFF          <1>       call   write_backspace
1010 000088E7 E913FEFFFF          <1>         jmp     loc_enter_hour_1
1011                              <1> loc_set_time_stc_2:
1012 000088EC E8B6FDFFFF          <1>       call   check_for_backspace
1013 000088F1 740A                <1>       je     short loc_set_time_bs_2
1014                              <1>       ;xor   bh, bh ; video page 0
1015 000088F3 E8F19AFFFF          <1>       call   beeper ; BEEP !
1016 000088F8 E971FEFFFF          <1>         jmp     loc_enter_time_separator_1
1017                              <1> loc_set_time_bs_2:
1018 000088FD E893FDFFFF          <1>       call   write_backspace
1019 00008902 E92BFEFFFF          <1>         jmp     loc_enter_hour_2
1020                              <1> loc_set_time_stc_3:
1021 00008907 E89BFDFFFF          <1>       call   check_for_backspace
1022 0000890C 740A                <1>       je     short loc_set_time_bs_3
1023                              <1>       ;xor   bh, bh ; video page 0
1024 0000890E E8D69AFFFF          <1>       call   beeper ; BEEP !6
1025 00008913 E974FEFFFF          <1>         jmp     loc_enter_minute_1
1026                              <1> loc_set_time_bs_3:
1027 00008918 E878FDFFFF          <1>       call   write_backspace
1028 0000891D E94CFEFFFF          <1>         jmp     loc_enter_time_separator_1
1029                              <1> loc_set_time_stc_4:
1030 00008922 E880FDFFFF          <1>       call   check_for_backspace
1031 00008927 740A                <1>       je     short loc_set_time_bs_4
1032                              <1>       ;xor   bh, bh ; video page 0
1033 00008929 E8BB9AFFFF          <1>       call   beeper ; BEEP !
1034 0000892E E984FEFFFF          <1>         jmp     loc_enter_minute_2
1035                              <1> loc_set_time_bs_4:
1036 00008933 E85DFDFFFF          <1>       call   write_backspace
1037 00008938 E94FFEFFFF          <1>         jmp     loc_enter_minute_1
1038                              <1> loc_set_time_stc_5:
1039 0000893D E865FDFFFF          <1>       call   check_for_backspace
1040 00008942 740A                <1>       je     short loc_set_time_bs_5
1041                              <1>       ;xor   bh, bh ; video page 0
1042 00008944 E8A09AFFFF          <1>       call   beeper ; BEEP !
1043 00008949 E994FEFFFF          <1>         jmp     loc_enter_time_separator_2
1044                              <1> loc_set_time_bs_5:
1045 0000894E E842FDFFFF          <1>       call   write_backspace
1046 00008953 E95FFEFFFF          <1>         jmp     loc_enter_minute_2
1047                              <1> loc_set_time_stc_6:
1048 00008958 E84AFDFFFF          <1>       call   check_for_backspace
1049 0000895D 7413                <1>       je     short loc_set_time_bs_6
1050                              <1>       ;xor   bh, bh ; video page 0
1051 0000895F E8859AFFFF          <1>       call   beeper ; BEEP !
1052 00008964 66C705[C4410100]30- <1>       mov    word [Second], 3030h
1052 0000896C 30                  <1>
1053 0000896D E99FFEFFFF          <1>         jmp     loc_enter_second_1
1054                              <1> loc_set_time_bs_6:
1055 00008972 E81EFDFFFF          <1>       call   write_backspace
1056 00008977 E966FEFFFF          <1>         jmp     loc_enter_time_separator_2
1057                              <1> loc_set_time_stc_7:
1058 0000897C E826FDFFFF          <1>       call   check_for_backspace
1059 00008981 740A                <1>       je     short loc_set_time_bs_7
1060                              <1>       ;xor   bh, bh ; video page 0
1061 00008983 E8619AFFFF          <1>       call   beeper ; BEEP !
1062 00008988 E9B3FEFFFF          <1>         jmp     loc_enter_second_2
1063                              <1> loc_set_time_bs_7:
1064 0000898D E803FDFFFF          <1>       call   write_backspace
1065 00008992 E97AFEFFFF          <1>         jmp     loc_enter_second_1
1066                              <1>
1067                              <1> print_volume_info:
1068                              <1>       ; 01/03/2016
1069                              <1>       ; 08/02/2016
1070                              <1>       ; 06/02/2016
1071                              <1>       ; 04/02/2016
1072                              <1>       ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
1073                              <1>       ; 25/10/2009
1074                              <1>       ;
1075                              <1>       ; "Volume Serial No: "
1076                              <1>       ;
1077                              <1>       ; INPUT  : AL = DOS Drive Number
1078                              <1>       ; OUTPUT : AH = FS Type
1079                              <1>       ;          AL = DOS Drive Name
1080                              <1>       ; CF = 0 -> OK
1081                              <1>       ; CF = 1 -> Drive not ready
1082                              <1>
1083 00008997 88C4                <1>       mov    ah, al
1084 00008999 28C0                <1>       sub    al, al
1085 0000899B 0FB7F0              <1>       movzx  esi, ax
1086 0000899E 81C600010900        <1>       add    esi, Logical_DOSDisks
1087 000089A4 8A06                <1>       mov    al, [esi]
1088 000089A6 3C41                <1>       cmp    al, 'A'
1089 000089A8 7304                <1>       jnb    short loc_pvi_set_vol_name
1090 000089AA 8A6604              <1>       mov    ah, [esi+LD_FSType]
1091 000089AD C3                  <1>       retn
1092                              <1>
1093                              <1> loc_pvi_set_vol_name:
1094 000089AE A2[F8410100]        <1>       mov    [Vol_Drv_Name], al
1095 000089B3 56                  <1>       push   esi
1096 000089B4 E858010000          <1>       call   move_volume_name_and_serial_no ;;;
1097 000089B9 7302                <1>       jnc    short loc_pvi_mvn_ok
1098 000089BB 5E                  <1>       pop    esi
1099 000089BC C3                  <1>       retn
1100                              <1>
1101                              <1> loc_pvi_mvn_ok:
1102 000089BD 8B3424              <1>       mov    esi, [esp]
1103 000089C0 807E04A1            <1>       cmp    byte [esi+LD_FSType], 0A1h
1104 000089C4 7509                <1>       jne    short loc_pvi_fat_vol_size
1105 000089C6 8B4670              <1>       mov    eax, [esi+LD_FS_VolumeSize]
1106 000089C9 0FB75E11            <1>       movzx  ebx, word [esi+LD_FS_BytesPerSec]
```

```
1107 000089CD EB07              <1>       jmp    short loc_vol_size_mul32
1108                            <1> loc_pvi_fat_vol_size:
1109 000089CF 8B4670            <1>       mov    eax, [esi+LD_TotalSectors]
1110 000089D2 0FB75E11          <1>       movzx  ebx, word [esi+LD_BPB+BPB_BytsPerSec]
1111                            <1> loc_vol_size_mul32:
1112 000089D6 F7E3              <1>       mul    ebx
1113 000089D8 09D2              <1>       or     edx, edx
1114 000089DA 7507              <1>       jnz    short loc_vol_size_in_kbytes
1115                            <1> loc_vol_size_in_bytes:
1116 000089DC B9[D6410100]      <1>       mov    ecx, VolSize_Bytes
1117 000089E1 EB0D              <1>       jmp    short loc_write_vol_size_str
1118                            <1> loc_vol_size_in_kbytes:
1119 000089E3 66BB0004          <1>       mov    bx, 1024
1120 000089E7 F7F3              <1>       div    ebx
1121 000089E9 B9[C9410100]      <1>       mov    ecx, VolSize_KiloBytes
1122 000089EE 31D2              <1>       xor    edx, edx ; 0
1123                            <1> loc_write_vol_size_str:
1124 000089F0 890D[BB910100]    <1>       mov    [VolSize_Unit1], ecx
1125                            <1>       ;
1126 000089F6 BF[D1910100]      <1>       mov    edi, Vol_Tot_Sec_Str_End
1127                            <1>         ;mov byte [edi], 0
1128 000089FB B90A000000        <1>       mov    ecx, 10
1129                            <1> loc_write_vol_size_chr:
1130 00008A00 F7F1              <1>       div    ecx
1131 00008A02 80C230            <1>       add    dl, '0'
1132 00008A05 4F                <1>       dec    edi
1133 00008A06 8817              <1>       mov    [edi], dl
1134 00008A08 85C0              <1>       test   eax, eax
1135 00008A0A 7404              <1>       jz     short loc_write_vol_size_str_ok
1136 00008A0C 28D2              <1>       sub    dl, dl ; 0
1137 00008A0E EBF0              <1>       jmp    short loc_write_vol_size_chr
1138                            <1>
1139                            <1> loc_write_vol_size_str_ok:
1140 00008A10 893D[C3910100]    <1>       mov    [Vol_Tot_Sec_Str_Start], edi
1141                            <1>       ;
1142 00008A16 BF[E1410100]      <1>       mov    edi, Vol_FS_Name
1143 00008A1B 8A4E03            <1>       mov    cl, [esi+LD_FATType]
1144 00008A1E 20C9              <1>       and    cl, cl ; 0 ?
1145 00008A20 7515              <1>       jnz    short loc_write_vol_FAT_str_1
1146 00008A22 66C7075452        <1>       mov    word [edi], 'TR'
1147 00008A27 C7470420465331    <1>       mov    dword [edi+4], ' FS1'
1148                            <1>       ;movzx ebx, word [esi+LD_FS_BytesPerSec]
1149 00008A2E 668B5E11          <1>       mov    bx, [esi+LD_FS_BytesPerSec]
1150 00008A32 8B4674            <1>       mov    eax, [esi+LD_FS_FreeSectors]
1151 00008A35 EB36              <1>       jmp    short loc_vol_freespace_mul32
1152                            <1>
1153                            <1> loc_write_vol_FAT_str_1:
1154 00008A37 66B83332          <1>       mov    ax, '32' ; FAT32
1155 00008A3B 80F902            <1>       cmp    cl, 2 ; [esi+LD_FATType]
1156 00008A3E 7708              <1>       ja     short loc_write_vol_FAT_str_2
1157 00008A40 66B83132          <1>       mov    ax, '12' ; FAT12
1158 00008A44 7202              <1>       jb     short loc_write_vol_FAT_str_2
1159 00008A46 B436              <1>       mov    ah, '6'  ; FAT16
1160                            <1> loc_write_vol_FAT_str_2:
1161 00008A48 C70746415420      <1>       mov    dword [edi], 'FAT '
1162 00008A4E 66894704          <1>       mov    word [edi+4], ax
1163                            <1>       ;
1164                            <1>       ;movzx ebx, word [esi+LD_BPB+BPB_BytsPerSec]
1165 00008A52 668B5E11          <1>       mov    bx, [esi+LD_BPB+BPB_BytsPerSec]
1166 00008A56 8B4674            <1>       mov    eax, [esi+LD_FreeSectors]
1167                            <1>
1168                            <1> loc_vol_freespace_recalc0:
1169                            <1>       ; 01/03/2016
1170 00008A59 83F8FF            <1>       cmp    eax, 0FFFFFFFFh
1171 00008A5C 720F              <1>       jb     short loc_vol_freespace_mul32
1172                            <1>       ;inc   eax ; 0
1173 00008A5E 20C9              <1>       and    cl, cl ; byte [esi+LD_FATType]
1174 00008A60 740B              <1>       jz     short loc_vol_freespace_mul32
1175 00008A62 53                <1>       push   ebx
1176 00008A63 66BB00FF          <1>       mov    bx, 0FF00h ; recalculate free sectors
1177 00008A67 E876490000        <1>       call   calculate_fat_freespace
1178 00008A6C 5B                <1>       pop    ebx
1179                            <1>
1180                            <1> loc_vol_freespace_mul32:
1181 00008A6D F7E3              <1>       mul    ebx
1182 00008A6F 09D2              <1>       or     edx, edx
1183 00008A71 7507              <1>       jnz    short loc_vol_fspace_in_kbytes
1184                            <1> loc_vol_fspace_in_bytes:
1185 00008A73 B9[D6410100]      <1>       mov    ecx, VolSize_Bytes
1186 00008A78 EB0D              <1>       jmp    short loc_write_vol_fspace_str
1187                            <1> loc_vol_fspace_in_kbytes:
1188 00008A7A 66BB0004          <1>       mov    bx, 1024
1189 00008A7E F7F3              <1>       div    ebx
1190 00008A80 B9[C9410100]      <1>       mov    ecx, VolSize_KiloBytes
1191 00008A85 31D2              <1>       xor    edx, edx ; 0
1192                            <1> loc_write_vol_fspace_str:
1193 00008A87 890D[BF910100]    <1>       mov    [VolSize_Unit2], ecx
1194                            <1>       ;
1195 00008A8D BF[E1910100]      <1>       mov    edi, Vol_Free_Sectors_Str_End
1196                            <1>         ;mov byte [edi], 0
1197 00008A92 B90A000000        <1>       mov    ecx, 10
1198                            <1> loc_write_vol_fspace_chr:
1199 00008A97 F7F1              <1>       div    ecx
1200 00008A99 80C230            <1>       add    dl, '0'
1201 00008A9C 4F                <1>       dec    edi
1202 00008A9D 8817              <1>       mov    [edi], dl
1203 00008A9F 85C0              <1>       test   eax, eax
1204 00008AA1 7404              <1>       jz     short loc_write_vol_fspace_str_ok
1205 00008AA3 28D2              <1>       sub    dl, dl ; 0
1206 00008AA5 EBF0              <1>       jmp    short loc_write_vol_fspace_chr
1207                            <1>
1208                            <1> loc_write_vol_fspace_str_ok:
1209 00008AA7 893D[D3910100]    <1>       mov    [Vol_Free_Sectors_Str_Start], edi
1210                            <1>       ;
1211 00008AAD BE[DF410100]      <1>       mov    esi, Volume_in_drive
```

```
1212 00008AB2 E89EEAFFFF          <1>       call   print_msg
1213 00008AB7 BE[1F420100]        <1>       mov    esi, Vol_Name
1214 00008ABC E894EAFFFF          <1>       call   print_msg
1215 00008AC1 BE[EF4C0100]        <1>       mov    esi, nextline
1216 00008AC6 E88AEAFFFF          <1>       call   print_msg
1217                              <1>       ;
1218 00008ACB BE[80420100]        <1>       mov    esi, Vol_Total_Sector_Header
1219 00008AD0 E880EAFFFF          <1>       call   print_msg
1220 00008AD5 8B35[C3910100]      <1>       mov    esi, [Vol_Tot_Sec_Str_Start]
1221 00008ADB E875EAFFFF          <1>       call   print_msg
1222 00008AE0 8B35[BB910100]      <1>       mov    esi, [VolSize_Unit1]
1223 00008AE6 E86AEAFFFF          <1>       call   print_msg
1224                              <1>       ;
1225 00008AEB BE[91420100]        <1>       mov    esi, Vol_Free_Sectors_Header
1226 00008AF0 E860EAFFFF          <1>       call   print_msg
1227 00008AF5 8B35[D3910100]      <1>       mov    esi, [Vol_Free_Sectors_Str_Start]
1228 00008AFB E855EAFFFF          <1>       call   print_msg
1229 00008B00 8B35[BF910100]      <1>       mov    esi, [VolSize_Unit2]
1230 00008B06 E84AEAFFFF          <1>       call   print_msg
1231                              <1>       ;
1232 00008B0B 5E                  <1>       pop    esi
1233                              <1>
1234                              <1>       ;mov   ah, [esi+LD_FSType]
1235                              <1>       ;mov   al, [esi+LD_FATType]
1236 00008B0C 668B4603            <1>       mov    ax, [esi+LD_FATType]
1237                              <1>
1238 00008B10 C3                  <1>       retn
1239                              <1>
1240                              <1> move_volume_name_and_serial_no:
1241                              <1>       ; 08/02/2016  (TRDOS 386 = TRDOS v2.0)
1242                              <1>       ; this routine will be called by
1243                              <1>       ; "print_volume_info" and "print_directory"
1244                              <1>       ; INPUT ->
1245                              <1>       ;      ESI = Logical DOS drv descripton table address
1246                              <1>       ; OUTPUT ->
1247                              <1>       ;      *Volume name will be moved to text area
1248                              <1>       ;      *Volume serial number will be converted to
1249                              <1>       ;       text and will be moved to text area
1250                              <1>       ;   cf = 1 -> invalid/unknown dos drive
1251                              <1>       ;   cf = 0 -> ecx = 0
1252                              <1>       ;
1253                              <1>       ; (eax, edx, ecx, esi, edi will be changed)
1254                              <1>
1255 00008B11 BF[1F420100]        <1>       mov    edi, Vol_Name
1256                              <1>
1257                              <1>       ;mov   ah, [esi+LD_FSType]
1258                              <1>       ;mov   al, [esi+LD_FATType]
1259 00008B16 668B4603            <1>       mov    ax, [esi+LD_FATType]
1260 00008B1A 80FCA1              <1>       cmp    ah, 0A1h
1261 00008B1D 7418                <1>       je     short mvn_2
1262 00008B1F 08E4                <1>       or     ah, ah
1263 00008B21 7404                <1>       jz     short mvn_0
1264 00008B23 08C0                <1>       or     al, al
1265 00008B25 7504                <1>       jnz    short mvn_1
1266                              <1> mvn_0:
1267 00008B27 8A06                <1>       mov    al, [esi]
1268 00008B29 F9                  <1>       stc
1269 00008B2A C3                  <1>       retn
1270                              <1> mvn_1:
1271 00008B2B 3C02                <1>       cmp    al, 2
1272 00008B2D 7717                <1>       ja     short mvn_3
1273                              <1>       ;or    al, al
1274                              <1>       ;jz    short mvn_2
1275 00008B2F 8B462D              <1>       mov    eax, [esi+LD_BPB+VolumeID]
1276 00008B32 83C631              <1>       add    esi, LD_BPB+VolumeLabel
1277 00008B35 EB15                <1>       jmp    short mvn_4
1278                              <1> mvn_2:
1279 00008B37 8B4628              <1>       mov    eax, [esi+LD_FS_VolumeSerial]
1280 00008B3A 83C62C              <1>       add    esi, LD_FS_VolumeName
1281 00008B3D B910000000          <1>       mov    ecx, 16
1282 00008B42 F3A5                <1>       rep    movsd
1283 00008B44 EB10                <1>       jmp    short mvn_5
1284                              <1> mvn_3:
1285 00008B46 8B4649              <1>       mov    eax, [esi+LD_BPB+FAT32_VolID]
1286 00008B49 83C64D              <1>       add    esi, LD_BPB+FAT32_VolLab
1287                              <1> mvn_4:
1288 00008B4C B90B000000          <1>       mov    ecx, 11
1289 00008B51 F3A4                <1>       rep    movsb
1290 00008B53 C60700              <1>       mov    byte [edi], 0
1291                              <1> mvn_5:
1292                              <1>       ;mov   [Current_VolSerial], eax
1293 00008B56 E8B2B7FFFF          <1>       call   dwordtohex
1294 00008B5B 8915[74420100]      <1>       mov    [Vol_Serial1], edx
1295 00008B61 A3[79420100]        <1>       mov    [Vol_Serial2], eax
1296                              <1>       ; ecx = 0
1297 00008B66 C3                  <1>       retn
1298                              <1>
1299                              <1> get_volume_serial_number:
1300                              <1>       ; 19/01/2016 (TRDOS 386 = TRDOS v2.0)
1301                              <1>       ; 08/08/2010
1302                              <1>       ;
1303                              <1>       ; INPUT -> DL = Logical DOS Drive number
1304                              <1>       ; OUTPUT -> EAX = Volume serial number
1305                              <1>       ;          BL= FAT Type
1306                              <1>       ;          BH = Logical DOS drv Number (DL input)
1307                              <1>       ; cf = 1 -> Drive not ready
1308                              <1>
1309 00008B67 31DB                <1>       xor    ebx, ebx
1310 00008B69 88D7                <1>       mov    bh, dl
1311 00008B6B 3815[50400100]      <1>       cmp    [Last_DOS_DiskNo], dl
1312 00008B71 7304                <1>       jnb    short loc_gvsn_start
1313                              <1> loc_gvsn_stc_retn:
1314 00008B73 31C0                <1>       xor    eax, eax
1315 00008B75 F9                  <1>       stc
1316 00008B76 C3                  <1>         retn
```

```
                        <1> loc_gvsn_start:
00008B77 56             <1>       push   esi
00008B78 BE00010900     <1>       mov    esi, Logical_DOSDisks
00008B7D 01DE           <1>       add    esi, ebx
00008B7F 8A5E03         <1>       mov    bl, [esi+LD_FATType]
00008B82 20DB           <1>       and    bl, bl
00008B84 740F           <1>       jz     short loc_gvsn_fs
00008B86 80FB02         <1>       cmp    bl, 2
00008B89 7705           <1>       ja     short loc_gvsn_fat32
                        <1> loc_gvsn_fat:
00008B8B 83C62D         <1>       add    esi, LD_BPB + VolumeID
00008B8E EB0E           <1>       jmp    short loc_gvsn_return
                        <1> loc_gvsn_fat32:
00008B90 83C649         <1>       add    esi, LD_BPB + FAT32_VolID
00008B93 EB09           <1>       jmp    short loc_gvsn_return
                        <1> loc_gvsn_fs:
00008B95 807E04A1       <1>       cmp    byte [esi+LD_FSType], 0A1h
00008B99 75D8           <1>       jne    short loc_gvsn_stc_retn
00008B9B 83C628         <1>       add    esi, LD_FS_VolumeSerial
                        <1> loc_gvsn_return:
00008B9E 8B06           <1>       mov    eax, [esi]
00008BA0 5E             <1>       pop    esi
00008BA1 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>
00008BA2 C605[74920100]00 <1>       mov    byte [Program_Exit],0
00008BA9 80F904         <1>       cmp    cl, 4
00008BAC 0F87B5020000   <1>         ja     c_6
00008BB2 0F8237010000   <1>         jb     c_2
                        <1> c_4:
                        <1>
                        <1> cmp_cmd_exit:
00008BB8 BF[BE400100]   <1>       mov    edi, Cmd_Exit
00008BBD E8C2030000     <1>       call   cmp_cmd
00008BC2 7208           <1>       jc     short cmp_cmd_date
                        <1>
00008BC4 C605[74920100]01 <1>         mov    byte [Program_Exit], 1
00008BCB C3             <1>         retn
                        <1>
                        <1> cmp_cmd_date:
00008BCC B104           <1>       mov    cl, 4
00008BCE BF[DA400100]   <1>       mov    edi, Cmd_Date
00008BD3 E8AC030000     <1>       call   cmp_cmd
00008BD8 720B           <1>         jc   short cmp_cmd_time
                        <1>
00008BDA E8D0F7FFFF     <1>       call   show_date
00008BDF E80FF8FFFF     <1>       call   set_date
00008BE4 C3             <1>       retn
                        <1>
                        <1> cmp_cmd_time:
00008BE5 B104           <1>       mov    cl, 4
00008BE7 BF[DF400100]   <1>       mov    edi, Cmd_Time
00008BEC E893030000     <1>       call   cmp_cmd
00008BF1 720B           <1>       jc     short cmp_cmd_show
                        <1>
00008BF3 E8C6FAFFFF     <1>       call   show_time
00008BF8 E8F8FAFFFF     <1>       call   set_time
00008BFD C3             <1>       retn
                        <1>
                        <1> cmp_cmd_show:
00008BFE B104           <1>       mov    cl, 4
00008C00 BF[F0400100]   <1>       mov    edi, Cmd_Show
00008C05 E87A030000     <1>       call   cmp_cmd
00008C0A 0F83050A0000   <1>         jnc     show_file
                        <1>
                        <1> cmp_cmd_echo:
00008C10 B104           <1>       mov    cl, 4
00008C12 BF[2C410100]   <1>       mov    edi, Cmd_Echo
00008C17 E868030000     <1>       call   cmp_cmd
00008C1C 7224           <1>       jc     short cmp_cmd_copy
                        <1>
                        <1>       ; 22/11/2017
                        <1>       ; AL = 0
00008C1E 803E20         <1>       cmp    byte [esi], 20h
00008C21 7215           <1>       jb     short cmd_echo_nextline
                        <1>       ; 14/04/2016
00008C23 56             <1>       push   esi
                        <1> cmd_echo_asciiz:
                        <1>       ;inc   esi
                        <1>       ;mov   al, [esi]
                        <1>       ; 22/11/2017
00008C24 AC             <1>       lodsb
00008C25 3C20           <1>       cmp    al, 20h
00008C27 73FB           <1>       jnb    short cmd_echo_asciiz
00008C29 4E             <1>       dec    esi
00008C2A C60600         <1>       mov    byte [esi], 0
```

```
1422 00008C2D 5E                <1>       pop    esi
1423 00008C2E 89F7              <1>       mov    edi, esi
1424 00008C30 E820E9FFFF        <1>       call   print_msg
1425 00008C35 C60700            <1>       mov    byte [edi], 0
1426                            <1> cmd_echo_nextline:
1427 00008C38 BE[5D4D0100]      <1>       mov    esi, NextLine
1428                            <1>       ;call  print_msg
1429                            <1>       ;retn
1430 00008C3D E913E9FFFF        <1>       jmp    print_msg
1431                            <1>
1432                            <1> cmp_cmd_copy:
1433 00008C42 B104              <1>       mov    cl, 4
1434 00008C44 BF[13410100]      <1>       mov    edi, Cmd_Copy
1435 00008C49 E836030000        <1>       call   cmp_cmd
1436 00008C4E 0F83CC170000      <1>       jnc    copy_file
1437                            <1>
1438                            <1> cmp_cmd_move:
1439 00008C54 B104              <1>       mov    cl, 4
1440 00008C56 BF[18410100]      <1>       mov    edi, Cmd_Move
1441 00008C5B E824030000        <1>       call   cmp_cmd
1442 00008C60 0F836E160000      <1>       jnc    move_file
1443                            <1>
1444                            <1> cmp_cmd_path:
1445 00008C66 B104              <1>       mov    cl, 4
1446 00008C68 BF[1D410100]      <1>       mov    edi, Cmd_Path
1447 00008C6D E812030000        <1>       call   cmp_cmd
1448 00008C72 0F83F0190000      <1>       jnc    set_get_path
1449                            <1>
1450                            <1> cmp_cmd_beep:
1451 00008C78 B104              <1>       mov    cl, 4
1452 00008C7A BF[4A410100]      <1>       mov    edi, Cmd_Beep
1453 00008C7F E800030000        <1>       call   cmp_cmd
1454 00008C84 720B              <1>       jc     short cmp_cmd_find
1455                            <1>       ; 13/05/2016
1456 00008C86 8A3D[EE890100]    <1>       mov    bh, [ptty] ; [ACTIVE_PAGE]
1457 00008C8C E95897FFFF        <1>       jmp    beeper
1458                            <1>
1459                            <1> cmp_cmd_find:
1460 00008C91 B104              <1>       mov    cl, 4
1461 00008C93 BF[27410100]      <1>       mov    edi, Cmd_Find
1462 00008C98 E8E7020000        <1>       call   cmp_cmd
1463 00008C9D 0F82C4020000      <1>         jc     cmp_cmd_external
1464                            <1>
1465                            <1>       ;call  find_and_list_files
1466 00008CA3 E9AF220000        <1>       jmp    find_and_list_files
1467                            <1>       ;retn
1468                            <1>
1469                            <1> c_1:
1470 00008CA8 AD                <1>       lodsd
1471                            <1> cmp_cmd_help:
1472 00008CA9 3C3F              <1>       cmp    al, '?'
1473 00008CAB 751D              <1>         jne    short cmp_cmd_remark
1474                            <1>
1475 00008CAD BE[B0400100]      <1>       mov    esi, Command_List
1476                            <1> cmd_help_next_w:
1477 00008CB2 E89EE8FFFF        <1>       call   print_msg
1478                            <1>
1479 00008CB7 803E20            <1>       cmp    byte [esi], 20h ; 0
1480 00008CBA 7232              <1>       jb     short cmd_help_retn
1481                            <1>
1482 00008CBC 56                <1>       push   esi
1483 00008CBD BE[EF4C0100]      <1>       mov    esi, nextline
1484 00008CC2 E88EE8FFFF        <1>       call   print_msg
1485 00008CC7 5E                <1>       pop    esi
1486 00008CC8 EBE8              <1>       jmp    short cmd_help_next_w
1487                            <1>
1488                            <1> cmp_cmd_remark:
1489 00008CCA 3C2A              <1>       cmp    al, '*'
1490 00008CCC 0F8595020000      <1>         jne    cmp_cmd_external
1491 00008CD2 46                <1>       inc    esi
1492 00008CD3 BF[E88A0100]      <1>       mov    edi, Remark
1493 00008CD8 8A06              <1>       mov    al, [esi]
1494 00008CDA 3C20              <1>       cmp    al, 20h
1495 00008CDC 7707              <1>       ja     short cmd_remark_write
1496 00008CDE 89FE              <1>       mov    esi, edi ; Remark
1497 00008CE0 E970E8FFFF        <1>       jmp    print_msg
1498                            <1>
1499                            <1> cmd_remark_write:
1500 00008CE5 AA                <1>       stosb
1501 00008CE6 AC                <1>       lodsb
1502 00008CE7 3C20              <1>       cmp    al, 20h
1503 00008CE9 73FA              <1>       jnb    short cmd_remark_write
1504 00008CEB C60700            <1>       mov    byte [edi], 0
1505                            <1>
1506                            <1> cmd_help_retn:
1507                            <1> cmd_remark_retn:
1508                            <1> cd_retn:
1509 00008CEE C3                <1>       retn
1510                            <1>
1511                            <1> c_2:
1512 00008CEF 80F902            <1>       cmp    cl, 2
1513 00008CF2 0F87AF000000      <1>         ja     c_3
1514 00008CF8 BE[368B0100]      <1>       mov    esi, CommandBuffer
1515 00008CFD 72A9              <1>       jb     short c_1
1516                            <1>
1517                            <1> cmp_cmd_cd:
1518 00008CFF 66AD              <1>       lodsw
1519 00008D01 663D4344          <1>       cmp    ax, 'CD'
1520 00008D05 7551              <1>       jne    short cmp_cmd_drive
1521 00008D07 46                <1>         inc  esi
1522                            <1> cd_0:
1523 00008D08 668B06            <1>       mov    ax, [esi]
1524 00008D0B 3C20              <1>       cmp    al, 20h
1525 00008D0D 76DF              <1>       jna    short cd_retn
1526                            <1>       ; 10/02/2016
```

```
1527 00008D0F 80FC3A              <1>       cmp    ah, ':'
1528 00008D12 7504                <1>       jne    short cd_1
1529 00008D14 46                  <1>       inc    esi
1530 00008D15 46                  <1>       inc    esi
1531 00008D16 EB49                <1>       jmp    short cd_2
1532                              <1>
1533                              <1> cd_1: ; change current directory
1534                              <1>       ; 29/11/2009
1535                              <1>       ; AH = CDh   ; to separate 'CD' command from others
1536                              <1>                    ; for restoring current directory
1537                              <1>                    ; 0CDh sign is for saving cdir into
1538                              <1>                    ; DOS drv description table cdir area
1539                              <1>
1540 00008D18 B4CD                <1>       mov    ah, 0CDh ; mov byte [CD_COMMAND], 0CDh
1541                              <1>
1542 00008D1A E81D230000          <1>       call   change_current_directory
1543 00008D1F 0F8337220000        <1>         jnc     change_prompt_dir_string
1544                              <1>
1545                              <1> cd_error_messages:
1546 00008D25 3C03                <1>       cmp    al, 3
1547 00008D27 740C                <1>       je     short cd_path_not_found
1548                              <1>       ; 16/10/2016 (15h -> 15)
1549 00008D29 3C0F                <1>       cmp    al, 15 ; drive not ready error
1550 00008D2B 7459                <1>       je     short cd_drive_not_ready
1551 00008D2D 3C11                <1>       cmp    al, 17 ; read error
1552 00008D2F 7455                <1>       je     short cd_drive_not_ready
1553 00008D31 3C13                <1>       cmp    al, 19 ; ; Bad directory/path name
1554 00008D33 7466                <1>       je     short cd_command_failed
1555                              <1>
1556                              <1> cd_path_not_found:
1557 00008D35 50                  <1>       push   eax ; 29/12/2017
1558                              <1>       ;push  ax
1559 00008D36 BE[53430100]        <1>       mov    esi, Msg_Dir_Not_Found
1560 00008D3B E815E8FFFF          <1>       call   print_msg
1561                              <1>       ;pop   ax
1562 00008D40 58                  <1>       pop    eax ; 29/12/2017
1563 00008D41 3A25[848A0100]      <1>       cmp    ah, [Current_Dir_Level]
1564 00008D47 0F830F220000        <1>         jnb     change_prompt_dir_string
1565 00008D4D 8825[848A0100]      <1>       mov    [Current_Dir_Level], ah
1566 00008D53 E904220000          <1>         jmp     change_prompt_dir_string
1567                              <1>
1568                              <1> cmp_cmd_drive: ; change current drive
1569                              <1>       ; C:, D:, E: etc.
1570 00008D58 80FC3A              <1>       cmp    ah, ':'
1571 00008D5B 0F8506020000        <1>         jne     cmp_cmd_external
1572                              <1>
1573                              <1> cd_2: ; 'CD C:', 'CD D:' ...
1574 00008D61 803E20              <1>       cmp    byte [esi], 20h
1575 00008D64 0F8707020000        <1>         ja      loc_cmd_failed
1576                              <1>
1577 00008D6A 24DF                <1>       and    al, 0DFh
1578 00008D6C 2C41                <1>       sub    al, 'A'
1579 00008D6E 0F82FD010000        <1>         jc      loc_cmd_failed
1580                              <1>
1581 00008D74 3A05[50400100]      <1>         cmp     al, [Last_DOS_DiskNo]
1582 00008D7A 770A                <1>         ja   short cd_drive_not_ready
1583                              <1>
1584 00008D7C 88C2                <1>       mov    dl, al
1585 00008D7E E85BF3FFFF          <1>       call   change_current_drive
1586 00008D83 7201                <1>       jc     short cd_drive_not_ready
1587 00008D85 C3                  <1>       retn
1588                              <1>
1589                              <1> cd_drive_not_ready:
1590 00008D86 BE[10430100]        <1>       mov    esi, Msg_Not_Ready_Read_Err
1591 00008D8B E8C5E7FFFF          <1>       call   print_msg
1592                              <1>
1593                              <1> cd_fail_drive_restart:
1594 00008D90 8A15[868A0100]      <1>       mov    dl, [Current_Drv]
1595                              <1>       ;call  change_current_drive
1596 00008D96 E943F3FFFF          <1>         jmp     change_current_drive
1597                              <1>       ;retn
1598                              <1>
1599                              <1> cd_command_failed:
1600 00008D9B BE[F1420100]        <1>       mov    esi, Msg_Bad_Command
1601 00008DA0 E8B0E7FFFF          <1>       call   print_msg
1602 00008DA5 EBE9                <1>       jmp    short cd_fail_drive_restart
1603                              <1>
1604                              <1> c_3:
1605                              <1> cmp_cmd_dir:
1606 00008DA7 BF[B0400100]        <1>       mov    edi, Cmd_Dir
1607 00008DAC E8D3010000          <1>       call   cmp_cmd
1608 00008DB1 0F8380020000        <1>       jnc    print_directory_list
1609                              <1>
1610                              <1> cmp_cmd_cls:
1611 00008DB7 B103                <1>       mov    cl, 3
1612 00008DB9 BF[EC400100]        <1>       mov    edi, Cmd_Cls
1613 00008DBE E8C1010000          <1>       call   cmp_cmd
1614 00008DC3 0F83A2E7FFFF        <1>         jnc  clear_screen
1615                              <1>
1616                              <1> cmp_cmd_ver:
1617 00008DC9 B103                <1>       mov    cl, 3
1618 00008DCB BF[BA400100]        <1>       mov    edi, Cmd_Ver
1619 00008DD0 E8AF010000          <1>       call   cmp_cmd
1620 00008DD5 720A                <1>       jc     short cmp_cmd_mem
1621                              <1>
1622 00008DD7 BE[58400100]        <1>       mov    esi, mainprog_Version
1623                              <1>       ;call  print_msg
1624 00008DDC E974E7FFFF          <1>       jmp    print_msg
1625                              <1>       ;retn
1626                              <1>
1627                              <1> cmp_cmd_mem:
1628 00008DE1 B103                <1>       mov    cl, 3
1629 00008DE3 BF[22410100]        <1>       mov    edi, Cmd_Mem
1630 00008DE8 E897010000          <1>       call   cmp_cmd
1631 00008DED 0F834CB4FFFF        <1>       jnc    memory_info
```

```
                                <1>
                                <1> cmp_cmd_del:
00008DF3 B103                   <1>         mov    cl, 3
00008DF5 BF[F5400100]           <1>         mov    edi, Cmd_Del
00008DFA E885010000             <1>         call   cmp_cmd
00008DFF 0F83280F0000           <1>           jnc     delete_file
                                <1>
                                <1> cmp_cmd_set:
00008E05 B103                   <1>         mov    cl, 3
00008E07 BF[E8400100]           <1>         mov    edi, Cmd_Set
00008E0C E873010000             <1>         call   cmp_cmd
00008E11 0F83C9170000           <1>           jnc     set_get_env
                                <1>
                                <1> cmp_cmd_run:
00008E17 B103                   <1>         mov    cl, 3
00008E19 BF[E4400100]           <1>         mov    edi, Cmd_Run
00008E1E E861010000             <1>         call   cmp_cmd
                                <1>         ; 07/05/2016
00008E23 0F823E010000           <1>           jc      cmp_cmd_external
00008E29 E90F1E0000             <1>         jmp    load_and_execute_file
                                <1> c_5:
                                <1> cmp_cmd_mkdir:
00008E2E BF[0D410100]           <1>         mov    edi, Cmd_Mkdir
00008E33 E84C010000             <1>         call   cmp_cmd
00008E38 0F83990A0000           <1>           jnc     make_directory
                                <1>
                                <1> cmp_cmd_rmdir:
00008E3E B105                   <1>         mov    cl, 5
00008E40 BF[07410100]           <1>         mov    edi, Cmd_Rmdir
00008E45 E83A010000             <1>         call   cmp_cmd
00008E4A 0F83AA0B0000           <1>           jnc     delete_directory
                                <1>
                                <1> cmp_cmd_chdir:
00008E50 B105                   <1>         mov    cl, 5
00008E52 BF[44410100]           <1>         mov    edi, Cmd_Chdir
00008E57 E828010000             <1>         call   cmp_cmd
00008E5C 0F8205010000           <1>           jc      cmp_cmd_external
                                <1>
00008E62 E9A1FEFFFF             <1>         jmp    cd_0
                                <1>
                                <1> c_6:
00008E67 80F906                 <1>         cmp    cl, 6
00008E6A 0F87E0000000           <1>           ja      c_8
00008E70 72BC                   <1>         jb     short c_5
                                <1> cmp_cmd_prompt:
00008E72 BF[C3400100]           <1>         mov    edi, Cmd_Prompt
00008E77 E808010000             <1>         call   cmp_cmd
00008E7C 722F                   <1>           jc   short cmp_cmd_volume
                                <1> get_prompt_name_fchar:
00008E7E AC                     <1>         lodsb
00008E7F 3C20                   <1>         cmp    al, 20h
00008E81 74FB                   <1>         je     short get_prompt_name_fchar
00008E83 7713                   <1>         ja     short loc_change_prompt_label
                                <1> default_command_prompt: ; 31/12/2017 ('sysprompt')
00008E85 BE[A4400100]           <1>         mov    esi, TRDOSPromptLabel
00008E8A C7065452444F           <1>         mov    dword [esi], "TRDO"
00008E90 66C746045300           <1>                mov    word [esi+4], "S"
                                <1> loc_cmd_prompt_return:
00008E96 C3                     <1>         retn
                                <1>
                                <1> set_command_prompt: ; 31/12/2017 ('sysprompt')
00008E97 AC                     <1>         lodsb
                                <1> loc_change_prompt_label:
00008E98 66B90B00               <1>         mov    cx, 11
00008E9C BF[A4400100]           <1>         mov    edi, TRDOSPromptLabel
                                <1> put_char_new_prompt_label:
00008EA1 AA                     <1>         stosb
00008EA2 AC                     <1>         lodsb
00008EA3 3C20                   <1>         cmp    al, 20h
00008EA5 7202                   <1>         jb     short pass_put_new_prompt_label
00008EA7 E2F8                   <1>         loop   put_char_new_prompt_label
                                <1> pass_put_new_prompt_label:
00008EA9 C60700                 <1>         mov    byte [edi], 0
00008EAC C3                     <1>         retn
                                <1>
                                <1> cmp_cmd_volume:
00008EAD B106                   <1>         mov    cl, 6
00008EAF BF[CA400100]           <1>         mov    edi, Cmd_Volume
00008EB4 E8CB000000             <1>         call   cmp_cmd
00008EB9 7255                   <1>           jc   short cmp_cmd_attrib
                                <1>
                                <1> cmd_vol1:
00008EBB AC                     <1>         lodsb
00008EBC 3C20                   <1>         cmp    al, 20h
00008EBE 7707                   <1>         ja     short cmd_vol2
00008EC0 A0[868A0100]           <1>         mov    al, [Current_Drv]
00008EC5 EB3D                   <1>         jmp    short cmd_vol4
                                <1> cmd_vol2:
00008EC7 3C41                   <1>         cmp    al, 'A'
00008EC9 0F82A2000000           <1>           jb      loc_cmd_failed
00008ECF 3C7A                   <1>         cmp    al, 'z'
00008ED1 0F879A000000           <1>           ja      loc_cmd_failed
00008ED7 3C5A                   <1>         cmp    al, 'Z'
00008ED9 760A                   <1>         jna    short cmd_vol3
00008EDB 3C61                   <1>         cmp    al, 'a'
00008EDD 0F828E000000           <1>           jb      loc_cmd_failed
00008EE3 24DF                   <1>         and    al, 0DFh
                                <1> cmd_vol3:
00008EE5 8A26                   <1>         mov    ah, [esi]
00008EE7 80FC3A                 <1>         cmp    ah, ':'
00008EEA 0F8581000000           <1>           jne     loc_cmd_failed
00008EF0 2C41                   <1>         sub    al, 'A'
00008EF2 3A05[50400100]         <1>           cmp     al, [Last_DOS_DiskNo]
00008EF8 760A                   <1>         jna    short cmd_vol4
                                <1>
```

```
1737 00008EFA BE[10430100]        <1>       mov    esi, Msg_Not_Ready_Read_Err
1738 00008EFF E951E6FFFF          <1>       jmp    print_msg
1739                              <1>
1740                              <1> cmd_vol4:
1741 00008F04 E88EFAFFFF          <1>       call   print_volume_info
1742 00008F09 0F8277FEFFFF        <1>         jc     cd_drive_not_ready
1743 00008F0F C3                  <1>       retn
1744                              <1>
1745                              <1> cmp_cmd_attrib:
1746 00008F10 B106                <1>       mov    cl, 6
1747 00008F12 BF[F9400100]        <1>       mov    edi, Cmd_Attrib
1748 00008F17 E868000000          <1>       call   cmp_cmd
1749 00008F1C 0F831D0F0000        <1>         jnc    set_file_attributes
1750                              <1>
1751                              <1> cmp_cmd_rename:
1752 00008F22 B106                <1>       mov    cl, 6
1753 00008F24 BF[00410100]        <1>       mov    edi, Cmd_Rename
1754 00008F29 E856000000          <1>       call   cmp_cmd
1755 00008F2E 0F8353110000        <1>         jnc    rename_file
1756                              <1>
1757                              <1> cmp_cmd_device:
1758 00008F34 B106                <1>       mov    cl, 6
1759 00008F36 BF[35410100]        <1>       mov    edi, Cmd_Device
1760 00008F3B E844000000          <1>       call   cmp_cmd
1761 00008F40 7225                <1>         jc   short cmp_cmd_external
1762                              <1>
1763 00008F42 C3                  <1>       retn
1764                              <1>
1765                              <1> c_7:
1766                              <1> cmp_cmd_devlist:
1767 00008F43 BF[3C410100]        <1>       mov    edi, Cmd_DevList
1768 00008F48 E837000000          <1>       call   cmp_cmd
1769 00008F4D 7218                <1>         jc   short cmp_cmd_external
1770                              <1>
1771                              <1> loc_cmd_return:
1772 00008F4F C3                  <1>       retn
1773                              <1>
1774                              <1> c_8:
1775 00008F50 80F908              <1>         cmp  cl, 8
1776 00008F53 7712                <1>       ja     short cmp_cmd_external
1777 00008F55 72EC                <1>       jb     short c_7
1778                              <1>
1779                              <1> cmp_cmd_longname:
1780 00008F57 BF[D1400100]        <1>       mov    edi, Cmd_LongName
1781 00008F5C E823000000          <1>       call   cmp_cmd
1782 00008F61 0F8350060000        <1>         jnc    get_and_print_longname
1783                              <1>
1784                              <1> cmp_cmd_external:
1785                              <1>       ; 07/05/2016
1786                              <1>       ; 22/04/2016
1787 00008F67 BE[368B0100]        <1>       mov    esi, CommandBuffer
1788 00008F6C E9CC1C0000          <1>       jmp    loc_run_check_filename
1789                              <1>
1790                              <1> loc_cmd_failed:
1791 00008F71 803D[368B0100]20    <1>       cmp    byte [CommandBuffer], 20h
1792 00008F78 76D5                <1>       jna    short loc_cmd_return
1793 00008F7A BE[F1420100]        <1>       mov    esi, Msg_Bad_Command
1794                              <1> ;     call   print_msg
1795                              <1> ;loc_cmd_return:
1796                              <1> ;     retn
1797 00008F7F E9D1E5FFFF          <1>       jmp    print_msg
1798                              <1>
1799                              <1> cmp_cmd:
1800                              <1>        ; 29/01/2016 (TRDOS 386 = TRDOS v2.0)
1801 00008F84 BE[368B0100]        <1>         mov esi, CommandBuffer
1802                              <1>         ; edi = internal command word (ASCIIZ)
1803                              <1>        ; ecx = command length (<=8)
1804                              <1> cmp_cmd_1:
1805 00008F89 AC                  <1>       lodsb
1806 00008F8A AE                  <1>       scasb
1807 00008F8B 750D                <1>       jne    short cmp_cmd_3
1808 00008F8D E2FA                <1>       loop   cmp_cmd_1
1809 00008F8F AC                  <1>       lodsb
1810 00008F90 3C20                <1>       cmp    al, 20h
1811 00008F92 7703                <1>       ja     short cmp_cmd_2
1812 00008F94 30C0                <1>       xor    al, al
1813                              <1>       ; ZF = 1 -> internal command word matches
1814 00008F96 C3                  <1>       retn
1815                              <1> cmp_cmd_2:
1816                              <1>       ; ZF = 0 (CF = 0) -> external command word
1817 00008F97 58                  <1>       pop    eax ; no return to the caller from here
1818 00008F98 EBCD                <1>       jmp    cmp_cmd_external
1819                              <1> cmp_cmd_3:
1820 00008F9A F9                  <1>       stc
1821                              <1>       ; CF = 1 -> internal command word does not match
1822 00008F9B C3                  <1>       retn
1823                              <1>
1824                              <1> loc_run_cmd_failed:
1825                              <1>       ; 15/03/2016
1826                              <1>       ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
1827                              <1>       ; 07/12/2009 (CMD_INTR.ASM)
1828                              <1>       ; 29/11/2009
1829                              <1>
1830 00008F9C E863000000          <1>       call   restore_cdir_after_cmd_fail
1831                              <1>
1832                              <1> loc_run_cmd_failed_cmp_al:
1833                              <1>       ; End of Restore_CDIR code (29/11/2009)
1834                              <1>
1835 00008FA1 3C01                <1>       cmp    al, 1 ; Bad command or file name
1836 00008FA3 74CC                <1>       je     loc_cmd_failed
1837                              <1> loc_run_dir_not_found:
1838 00008FA5 3C03                <1>       cmp    al, 3
1839 00008FA7 750A                <1>       jne    short loc_run_file_notfound_msg
1840                              <1>       ; Path not found (MS-DOS Error Code = 3)
1841 00008FA9 BE[53430100]        <1>       mov    esi, Msg_Dir_Not_Found
```

```
1842 00008FAE E9A2E5FFFF          <1>       jmp    print_msg
1843                              <1>
1844                              <1> loc_run_file_notfound_msg:
1845 00008FB3 3C02                <1>       cmp    al, 2 ; File not found
1846 00008FB5 750A                <1>       jne    short loc_run_file_drv_read_err
1847                              <1>
1848                              <1> loc_print_file_notfound_msg:
1849 00008FB7 BE[6A430100]        <1>         mov     esi, Msg_File_Not_Found
1850                              <1>       ;call  proc_printmsg
1851                              <1>       ;retn
1852 00008FBC E994E5FFFF          <1>       jmp    print_msg
1853                              <1>
1854                              <1> loc_run_file_drv_read_err:
1855                              <1>       ; Err: 17 (Read fault)
1856 00008FC1 3C11                <1>       cmp    al, 17 ; Drive not ready or read error
1857 00008FC3 7404                <1>       je     short loc_run_file_print_drv_read_err
1858                              <1>       ;
1859 00008FC5 3C0F                <1>       cmp    al, 15 ; Drive not ready (or read error)
1860 00008FC7 750A                <1>       jne    short loc_run_file_toobig
1861                              <1>
1862                              <1> loc_run_file_print_drv_read_err:
1863 00008FC9 BE[10430100]        <1>       mov    esi, Msg_Not_Ready_Read_Err
1864 00008FCE E982E5FFFF          <1>       jmp    print_msg
1865                              <1>
1866                              <1> loc_run_file_toobig:
1867 00008FD3 3C08                <1>       cmp    al, 8 ; Not enough free memory to load&run file
1868 00008FD5 750A                <1>       jne    short loc_run_file_perm_denied
1869 00008FD7 BE[B5430100]        <1>       mov    esi, Msg_Insufficient_Memory
1870 00008FDC E974E5FFFF          <1>       jmp    print_msg
1871                              <1>
1872                              <1> loc_run_file_perm_denied:
1873                              <1>       ; 29/12/2017
1874 00008FE1 3C0B                <1>       cmp    al, ERR_PERM_DENIED ; 11 ; Permission denied
1875 00008FE3 750A                <1>       jne    short loc_run_misc_error
1876 00008FE5 BE[4A450100]        <1>       mov    esi, Msg_Permission_Denied
1877 00008FEA E966E5FFFF          <1>       jmp    print_msg
1878                              <1>
1879                              <1>       ; 15/03/2016
1880                              <1> print_misc_error_msg:
1881                              <1> loc_run_misc_error:
1882                              <1>       ; AL = Error code
1883 00008FEF E8D9B2FFFF          <1>       call   bytetohex
1884 00008FF4 66A3[E9430100]      <1>         mov     [error_code_hex], ax
1885                              <1>
1886 00008FFA BE[CC430100]        <1>       mov    esi, Msg_Error_Code
1887                              <1>       ;call  print_msg
1888                              <1>       ;retn
1889                              <1>
1890 00008FFF E951E5FFFF          <1>       jmp    print_msg
1891                              <1>
1892                              <1> restore_cdir_after_cmd_fail:
1893                              <1>       ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
1894 00009004 50                  <1>       push   eax
1895 00009005 8A3D[E2910100]      <1>       mov    bh, [RUN_CDRV] ; it is set at the beginning
1896                              <1>                               ; of the 'run' command.
1897 0000900B 3A3D[868A0100]      <1>       cmp    bh, [Current_Drv]
1898 00009011 7409                <1>       je     short loc_run_restore_cdir
1899 00009013 88FA                <1>       mov    dl, bh
1900 00009015 E8C4F0FFFF          <1>       call   change_current_drive
1901 0000901A EB19                <1>       jmp    short loc_run_err_pass_restore_cdir
1902                              <1>
1903                              <1> loc_run_restore_cdir:
1904 0000901C 803D[51400100]00    <1>       cmp    byte [Restore_CDIR], 0
1905 00009023 7610                <1>       jna    short loc_run_err_pass_restore_cdir
1906 00009025 30DB                <1>       xor    bl, bl
1907 00009027 0FB7F3              <1>       movzx  esi, bx
1908 0000902A 81C600010900        <1>       add    esi, Logical_DOSDisks
1909 00009030 E860F1FFFF          <1>       call   restore_current_directory
1910                              <1>
1911                              <1> loc_run_err_pass_restore_cdir:
1912 00009035 58                  <1>       pop    eax
1913 00009036 C3                  <1>       retn
1914                              <1>
1915                              <1> print_directory_list:
1916                              <1>       ; 10/02/2016
1917                              <1>       ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)
1918                              <1>       ; 06/12/2009 ('cmp_cmd_dir')
1919                              <1>       ;
1920 00009037 66C705[24930100]00- <1>       mov    word [AttributesMask], 0800h ; ..except volume names..
1920 0000903F 08                  <1>
1921 00009040 A0[868A0100]        <1>       mov    al, [Current_Drv]
1922 00009045 A2[E2910100]        <1>       mov    [RUN_CDRV], al
1923                              <1> get_dfname_fchar:
1924 0000904A AC                  <1>       lodsb
1925 0000904B 3C20                <1>       cmp    al, 20h
1926 0000904D 74FB                <1>       je     short get_dfname_fchar
1927 0000904F 0F82A4000000        <1>         jb      loc_print_dir_call_all
1928 00009055 3C2D                <1>       cmp    al, '-'
1929 00009057 7542                <1>       jne    short loc_print_dir_call_flt
1930                              <1> get_next_attr_char:
1931 00009059 AC                  <1>       lodsb
1932 0000905A 3C20                <1>       cmp    al, 20h
1933 0000905C 74FB                <1>       je     short get_next_attr_char
1934 0000905E 0F820DFFFFFF        <1>         jb      loc_cmd_failed
1935 00009064 24DF                <1>       and    al, 0DFh
1936 00009066 3C44                <1>       cmp    al, 'D' ; directories only ?
1937 00009068 7512                <1>       jne    short pass_only_directories
1938 0000906A AC                  <1>       lodsb
1939 0000906B 3C20                <1>       cmp    al, 20h
1940 0000906D 0F87FEFEFFFF        <1>         ja      loc_cmd_failed
1941 00009073 800D[24930100]10    <1>       or     byte [AttributesMask], 10h ; ..directory..
1942 0000907A EB18                <1>       jmp    short get_dfname_fchar_attr
1943                              <1> pass_only_directories:
1944 0000907C 3C46                <1>       cmp    al, 'F'      ; files only ?
1945 0000907E 0F85B0000000        <1>         jne     check_attr_s
```

```
1946 00009084 AC                <1>       lodsb
1947 00009085 3C20              <1>       cmp    al, 20h
1948 00009087 0F87E4FEFFFF      <1>         ja      loc_cmd_failed
1949 0000908D 800D[25930100]10  <1>       or     byte [AttributesMask+1], 10h ; ..except directories..
1950                            <1> get_dfname_fchar_attr:
1951 00009094 AC                <1>       lodsb
1952 00009095 3C20              <1>       cmp    al, 20h
1953 00009097 74FB              <1>       je     short get_dfname_fchar_attr
1954 00009099 725E              <1>       jb     short loc_print_dir_call_all
1955                            <1>
1956                            <1> loc_print_dir_call_flt:
1957 0000909B 4E                <1>       dec    esi
1958 0000909C BF[26930100]      <1>       mov    edi, FindFile_Drv
1959 000090A1 E8AC250000        <1>       call   parse_path_name
1960 000090A6 7308              <1>       jnc    short loc_print_dir_change_drv_1
1961 000090A8 3C01              <1>       cmp    al, 1
1962 000090AA 0F87ECFEFFFF      <1>         ja      loc_run_cmd_failed
1963                            <1>
1964                            <1> loc_print_dir_change_drv_1:
1965 000090B0 8A15[26930100]    <1>       mov    dl, [FindFile_Drv]
1966                            <1> loc_print_dir_change_drv_2:
1967 000090B6 3A15[E2910100]    <1>       cmp    dl, [RUN_CDRV]
1968 000090BC 740B              <1>       je     short loc_print_dir_change_directory
1969 000090BE E81BF0FFFF        <1>       call   change_current_drive
1970 000090C3 0F82D3FEFFFF      <1>         jc      loc_run_cmd_failed
1971                            <1> loc_print_dir_change_directory:
1972 000090C9 803D[27930100]20  <1>       cmp    byte [FindFile_Directory], 20h ; 0 or 20h ?
1973 000090D0 761D              <1>       jna    short pass_print_dir_change_directory
1974                            <1>
1975 000090D2 FE05[51400100]    <1>       inc    byte [Restore_CDIR]
1976 000090D8 BE[27930100]      <1>       mov    esi, FindFile_Directory
1977 000090DD 30E4              <1>       xor    ah, ah ; CD_COMMAND sign -> 0
1978 000090DF E8581F0000        <1>       call   change_current_directory
1979 000090E4 0F82B2FEFFFF      <1>         jc      loc_run_cmd_failed
1980                            <1>
1981                            <1> loc_print_dir_change_prompt_dir_string:
1982 000090EA E86D1E0000        <1>       call   change_prompt_dir_string
1983                            <1>
1984                            <1> pass_print_dir_change_directory:
1985 000090EF BE[68930100]      <1>       mov    esi, FindFile_Name
1986 000090F4 803E20            <1>       cmp    byte [esi], 20h ; ; 0 or 20h ?
1987 000090F7 7706              <1>       ja     short loc_print_dir_call
1988                            <1>
1989                            <1> loc_print_dir_call_all:
1990 000090F9 C7062A2E2A00      <1>       mov    dword [esi], '*.*'
1991                            <1> loc_print_dir_call:
1992 000090FF E87E000000        <1>       call   print_directory
1993                            <1>
1994 00009104 8A15[E2910100]    <1>       mov    dl, [RUN_CDRV]  ; it is set at the beginning
1995 0000910A 3A15[868A0100]    <1>       cmp    dl, [Current_Drv]
1996 00009110 7406              <1>       je     short loc_print_dir_call_restore_cdir_retn
1997 00009112 E8C7EFFFFF        <1>       call   change_current_drive
1998 00009117 C3                <1>       retn
1999                            <1>
2000                            <1> loc_print_dir_call_restore_cdir_retn:
2001 00009118 803D[51400100]00  <1>       cmp    byte [Restore_CDIR], 0
2002 0000911F 7610              <1>       jna    short pass_print_dir_call_restore_cdir_retn
2003                            <1>
2004 00009121 BE00010900        <1>       mov    esi, Logical_DOSDisks
2005 00009126 31C0              <1>       xor    eax, eax
2006 00009128 88D4              <1>       mov    ah, dl
2007 0000912A 01C6              <1>       add    esi, eax
2008                            <1>
2009 0000912C E864F0FFFF        <1>       call   restore_current_directory
2010                            <1>
2011                            <1> pass_print_dir_call_restore_cdir_retn:
2012 00009131 C3                <1>       retn
2013                            <1>
2014                            <1> check_attr_s_cap:
2015 00009132 24DF              <1>       and    al, 0DFh
2016                            <1> check_attr_s:
2017 00009134 3C53              <1>       cmp    al, 'S'
2018 00009136 7514              <1>       jne    short pass_attr_s
2019 00009138 800D[24930100]04  <1>       or     byte [AttributesMask], 4 ; system
2020 0000913F AC                <1>       lodsb
2021 00009140 3C20              <1>       cmp    al, 20h
2022 00009142 0F844CFFFFFF      <1>         je      get_dfname_fchar_attr
2023 00009148 72AF              <1>       jb     short loc_print_dir_call_all
2024 0000914A 24DF              <1>       and    al, 0DFh
2025                            <1> pass_attr_s:
2026 0000914C 3C48              <1>       cmp    al, 'H'
2027 0000914E 7514              <1>       jne    short pass_attr_h
2028 00009150 800D[24930100]02  <1>       or     byte [AttributesMask], 2 ; hidden
2029                            <1> pass_attr_shr:
2030 00009157 AC                <1>       lodsb
2031 00009158 3C20              <1>       cmp    al, 20h
2032 0000915A 0F8434FFFFFF      <1>         je      get_dfname_fchar_attr
2033 00009160 7297              <1>       jb     short loc_print_dir_call_all
2034 00009162 EBCE              <1>       jmp    short check_attr_s_cap
2035                            <1>
2036                            <1> pass_attr_h:
2037 00009164 3C52              <1>       cmp    al, 'R'
2038 00009166 7509              <1>       jne    short pass_attr_r
2039 00009168 800D[24930100]01  <1>       or     byte [AttributesMask], 1 ; read only
2040 0000916F EBE6              <1>       jmp    short pass_attr_shr
2041                            <1>
2042                            <1> pass_attr_r:
2043 00009171 3C41              <1>       cmp    al, 'A'
2044 00009173 0F85F8FDFFFF      <1>         jne     loc_cmd_failed
2045 00009179 800D[24930100]20  <1>       or     byte [AttributesMask], 20h ; archive
2046 00009180 EBD5              <1>       jmp    short pass_attr_shr
2047                            <1>
2048                            <1> print_directory:
2049                            <1>       ; 13/05/2016
2050                            <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>
00009182 56                     <1>     push    esi
                                <1>
00009183 29C0                   <1>     sub     eax, eax
                                <1>
00009185 66A3[B0930100]         <1>     mov     word [Dir_Count], ax ; 0
0000918B 66A3[AE930100]         <1>     mov     word [File_Count], ax ; 0
00009191 A3[B2930100]           <1>     mov     dword [Total_FSize], eax ; 0
                                <1>
00009196 E8D0E3FFFF             <1>     call    clear_screen
                                <1>
0000919B 31C9                   <1>     xor     ecx, ecx
0000919D 8A2D[868A0100]         <1>     mov     ch, [Current_Drv] ; DirBuff_Drv - 'A'
000091A3 A0[878A0100]           <1>     mov     al, [Current_Dir_Drv]
000091A8 A2[0E420100]           <1>     mov     [Dir_Drive_Name], al
000091AD BE00010900             <1>     mov     esi, Logical_DOSDisks
000091B2 01CE                   <1>     add     esi, ecx
                                <1>
000091B4 E858F9FFFF             <1>     call    move_volume_name_and_serial_no
000091B9 730C                   <1>     jnc     short print_dir_strlen_check
                                <1>
000091BB 5E                     <1>     pop     esi
000091BC 8A3D[EE890100]         <1>     mov     bh, [ptty] ; [ACTIVE_PAGE]
                                <1>     ;call   beeper
                                <1>     ;retn
000091C2 E92292FFFF             <1>     jmp     beeper  ; beep ! and return
                                <1>
                                <1> print_dir_strlen_check:
000091C7 BE[898A0100]           <1>       mov     esi, Current_Dir_Root
000091CC BF[AB420100]           <1>       mov     edi, Dir_Str_Root
                                <1>
                                <1>       ;xor    ecx, ecx
000091D1 8A0D[E58A0100]         <1>         mov     cl, [Current_Dir_StrLen]
000091D7 FEC1                   <1>       inc     cl
000091D9 80F940                 <1>       cmp     cl, 64
000091DC 760D                   <1>       jna     short pass_print_dir_strlen_shorting
000091DE 46                     <1>       inc     esi
000091DF 01CE                   <1>       add     esi, ecx
000091E1 83EE40                 <1>       sub     esi, 64
000091E4 47                     <1>       inc     edi
000091E5 B82E2E2E20             <1>       mov     eax, '... '
000091EA AB                     <1>       stosd
                                <1>
                                <1> pass_print_dir_strlen_shorting:
000091EB F3A4                   <1>       rep     movsb
                                <1>
000091ED BE[01420100]           <1>       mov     esi, Dir_Drive_Str
000091F2 E85EE3FFFF             <1>       call    print_msg
                                <1>
000091F7 BE[60420100]           <1>       mov     esi, Vol_Serial_Header
000091FC E854E3FFFF             <1>       call    print_msg
                                <1>
00009201 BE[A0420100]           <1>       mov     esi, Dir_Str_Header
00009206 E84AE3FFFF             <1>       call    print_msg
                                <1>
0000920B BE[ED4C0100]           <1>       mov     esi, next2line
00009210 E840E3FFFF             <1>       call    print_msg
                                <1>
                                <1> loc_print_dir_first_file:
00009215 C605[C5930100]10       <1>       mov     byte [PrintDir_RowCounter], 16
0000921C 66A1[24930100]         <1>       mov     ax, [AttributesMask]
00009222 5E                     <1>       pop     esi
                                <1>
00009223 E859020000             <1>       call    find_first_file
00009228 0F826F010000           <1>         jc      loc_dir_ok
                                <1>
                                <1> loc_dfname_use_this:
                                <1>       ; bl = File Attributes (bh = Long Name Entry Length)
0000922E F6C310                 <1>       test    bl, 10h  ; Is it a directory?
00009231 741B                   <1>       jz      short loc_not_dir
                                <1>
00009233 66FF05[B0930100]       <1>       inc     word [Dir_Count]
0000923A 89F2                   <1>       mov     edx, esi      ; FindFile_DirEntry address
0000923C BE[F0430100]           <1>       mov     esi, Type_Dir; '<DIR>     '
00009241 BF[07440100]           <1>       mov     edi, Dir_Or_FileSize
                                <1>       ; move 10 bytes
00009246 A5                     <1>       movsd
00009247 A5                     <1>       movsd
00009248 66A5                   <1>       movsw
0000924A 89D6                   <1>       mov     esi, edx
0000924C EB36                   <1>       jmp     short loc_dir_attribute
                                <1>
                                <1> loc_not_dir:
0000924E 66FF05[AE930100]       <1>       inc     word [File_Count]
00009255 0105[B2930100]         <1>       add     [Total_FSize], eax
                                <1>
0000925B B90A000000             <1>       mov     ecx, 10  ; 32 bit divisor
00009260 89CF                   <1>       mov     edi, ecx
00009262 81C7[07440100]         <1>       add     edi, Dir_Or_FileSize
                                <1> loc_dir_rdivide:
00009268 29D2                   <1>       sub     edx, edx
0000926A F7F1                   <1>       div     ecx     ; remainder in dl (< 10)
0000926C 80C230                 <1>       add     dl, '0'      ; to make visible (ascii)
0000926F 4F                     <1>       dec     edi
00009270 8817                   <1>       mov     [edi], dl
00009272 21C0                   <1>       and     eax, eax
00009274 75F2                   <1>       jnz     short loc_dir_rdivide
                                <1>
```

```
                                <1> loc_dir_fill_space:
00009276 81FF[07440100]         <1>         cmp     edi, Dir_Or_FileSize
0000927C 7606                   <1>         jna     short loc_dir_attribute
0000927E 4F                     <1>         dec     edi
0000927F C60720                 <1>         mov     byte [edi], 20h
00009282 EBF2                   <1>         jmp     short loc_dir_fill_space
                                <1>
                                <1> loc_dir_attribute:
00009284 C705[12440100]2020-    <1>         mov    dword [File_Attribute], 20202020h
0000928C 2020                   <1>
                                <1>
0000928E 80FB20                 <1>         cmp    bl, 20h  ; Is it an archive file?
00009291 7207                   <1>         jb     short loc_dir_pass_arch
00009293 C605[15440100]41       <1>         mov    byte [File_Attribute+3], 'A'
                                <1>
                                <1> loc_dir_pass_arch:
0000929A 80E307                 <1>         and    bl, 7
0000929D 7428                   <1>         jz     short loc_dir_file_name
0000929F 88DF                   <1>         mov    bh, bl
000092A1 80E303                 <1>         and    bl, 3
000092A4 38DF                   <1>         cmp    bh, bl
000092A6 7607                   <1>         jna    short loc_dir_pass_s
000092A8 C605[12440100]53       <1>         mov    byte [File_Attribute], 'S'
                                <1>
                                <1> loc_dir_pass_s:
000092AF 80E302                 <1>         and     bl,2
000092B2 7407                   <1>         jz      short loc_dir_pass_h
000092B4 C605[13440100]48       <1>         mov     byte [File_Attribute+1], 'H'
                                <1> loc_dir_pass_h:
000092BB 80E701                 <1>         and     bh,1
000092BE 7407                   <1>         jz      short loc_dir_file_name
000092C0 C605[14440100]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
000092C7 8B5E16                 <1>         mov    ebx, [esi+16h]
000092CA 89F1                   <1>         mov    ecx, esi ; FindFile_DirEntry address
000092CC BF[FA430100]           <1>         mov     edi, File_Name
                                <1>         ; move 8 bytes
000092D1 A5                     <1>         movsd
000092D2 A5                     <1>         movsd
000092D3 C60720                 <1>         mov    byte [edi], 20h
000092D6 47                     <1>         inc    edi
                                <1>         ; move 3 bytes
000092D7 66A5                   <1>         movsw
000092D9 A4                     <1>         movsb
000092DA 89CE                   <1>         mov    esi, ecx
                                <1>
                                <1> Dir_Time_start:
                                <1>         ;mov   ax, dx       ; Time
000092DC 6689D8                 <1>         mov    ax, bx
000092DF 66C1E805               <1>         shr    ax, 5        ; shift right 5 times
000092E3 6683E03F               <1>         and    ax, 0000111111b     ; Minute Mask
000092E7 D40A                   <1>         aam                 ; Q([AL]/10)->AH
                                <1>                             ; R([AL]/10)->AL
                                <1>                             ; [AL]+[AH]= Minute as BCD
000092E9 660D3030               <1>         or     ax, '00'     ; Convert to ASCII
000092ED 86E0                   <1>         xchg   ah, al
000092EF 66A3[25440100]         <1>         mov    [File_Minute], ax
                                <1>
                                <1>         ;mov   al, dh
000092F5 88F8                   <1>         mov    al, bh
000092F7 C0E803                 <1>         shr    al, 3        ; shift right 3 times
000092FA D40A                   <1>         aam                 ; [AL]+[AH]= Hours as BCD
000092FC 660D3030               <1>         or     ax, '00'
00009300 86E0                   <1>         xchg   ah, al
00009302 66A3[22440100]         <1>         mov     [File_Hour], ax
                                <1>
00009308 C1EB10                 <1>         shr    ebx, 16             ; BX = Date
                                <1>
                                <1> Dir_Date_start:
0000930B 6689D8                 <1>         mov    ax, bx       ; Date
0000930E 6683E01F               <1>         and    ax, 00011111b; Day Mask
00009312 D40A                   <1>         aam                 ; Q([AL]/10)->AH
                                <1>                             ; R([AL]/10)->AL
                                <1>                             ; [AL]+[AH]= Day as BCD
00009314 660D3030               <1>         or     ax, '00'     ; Convert to ASCII
00009318 86C4                   <1>         xchg   al, ah
                                <1>
0000931A 66A3[17440100]         <1>         mov    [File_Day], ax
                                <1>
00009320 6689D8                 <1>         mov    ax, bx
00009323 66C1E805               <1>         shr    ax, 5        ; shift right 5 times
00009327 6683E00F               <1>         and    ax, 00001111b; Month Mask
0000932B D40A                   <1>         aam
0000932D 660D3030               <1>         or     ax, '00'
00009331 86E0                   <1>         xchg   ah, al
00009333 66A3[1A440100]         <1>         mov    [File_Month], ax
                                <1>
00009339 6689D8                 <1>         mov    ax, bx
0000933C 66C1E809               <1>         shr     ax, 9
00009340 6683E07F               <1>         and    ax, 01111111b; Result = Year - 1980
00009344 6605BC07               <1>         add    ax, 1980
                                <1>
00009348 B10A                   <1>         mov    cl, 10
0000934A F6F1                   <1>         div    cl           ; Q -> AL, R -> AH
0000934C 80CC30                 <1>         or     ah, '0'
0000934F 8825[20440100]         <1>         mov    [File_Year+3], ah
00009355 D40A                   <1>         aam
00009357 86E0                   <1>         xchg   ah, al
00009359 80CC30                 <1>         or     ah, '0'        ; Convert to ASCII
0000935C 8825[1F440100]         <1>         mov    [File_Year+2], ah
00009362 D40A                   <1>         aam
00009364 86C4                   <1>         xchg   al, ah
00009366 660D3030               <1>         or     ax, '00'
```

```
2260 0000936A 66A3[1D440100]      <1>       mov    [File_Year], ax
2261                              <1>
2262                              <1> loc_show_line:
2263 00009370 56                  <1>       push   esi
2264 00009371 BE[FA430100]        <1>       mov     esi, File_Name
2265 00009376 E8DAE1FFFF          <1>       call   print_msg
2266 0000937B BE[EF4C0100]        <1>       mov    esi, nextline
2267 00009380 E8D0E1FFFF          <1>       call   print_msg
2268 00009385 5E                  <1>       pop    esi
2269                              <1>
2270 00009386 FE0D[C5930100]      <1>       dec    byte [PrintDir_RowCounter]
2271 0000938C 0F84D4000000        <1>         jz      pause_dir_scroll
2272                              <1>
2273                              <1> loc_next_entry:
2274 00009392 E899010000          <1>       call   find_next_file
2275 00009397 0F8391FEFFFF        <1>         jnc     loc_dfname_use_this
2276                              <1>
2277                              <1> loc_dir_ok:
2278 0000939D B90A000000          <1>       mov     ecx, 10
2279 000093A2 66A1[B0930100]      <1>       mov    ax, [Dir_Count]
2280 000093A8 BF[3B440100]        <1>       mov    edi, Decimal_Dir_Count
2281 000093AD 6639C8              <1>       cmp    ax, cx ; 10
2282 000093B0 7216                <1>       jb     short pass_ddc
2283 000093B2 47                  <1>       inc    edi
2284 000093B3 6683F864            <1>       cmp    ax, 100
2285 000093B7 720F                <1>       jb     short pass_ddc
2286 000093B9 47                  <1>       inc    edi
2287 000093BA 663DE803            <1>       cmp    ax, 1000
2288 000093BE 7208                <1>       jb     short pass_ddc
2289 000093C0 47                  <1>       inc    edi
2290 000093C1 663D1027            <1>       cmp    ax, 10000
2291 000093C5 7201                <1>       jb     short pass_ddc
2292 000093C7 47                  <1>       inc    edi
2293                              <1> pass_ddc:
2294 000093C8 886F01              <1>       mov     [edi+1], ch ; 0
2295                              <1> loc_ddc_rediv:
2296 000093CB 31D2                <1>       xor     edx, edx
2297 000093CD 66F7F1              <1>       div     cx   ; 10
2298 000093D0 80C230              <1>       add     dl, '0'
2299 000093D3 8817                <1>       mov     [edi], dl
2300 000093D5 4F                  <1>       dec     edi
2301 000093D6 6609C0              <1>       or     ax, ax
2302 000093D9 75F0                <1>       jnz    short loc_ddc_rediv
2303                              <1>
2304 000093DB 66A1[AE930100]      <1>       mov     ax, [File_Count]
2305 000093E1 BF[2A440100]        <1>       mov     edi, Decimal_File_Count
2306 000093E6 6639C8              <1>       cmp     ax, cx ; 10
2307 000093E9 7216                <1>       jb      short pass_dfc
2308 000093EB 47                  <1>       inc     edi
2309 000093EC 6683F864            <1>       cmp     ax, 100
2310 000093F0 720F                <1>       jb      short pass_dfc
2311 000093F2 47                  <1>       inc     edi
2312 000093F3 663DE803            <1>       cmp     ax, 1000
2313 000093F7 7208                <1>       jb      short pass_dfc
2314 000093F9 47                  <1>       inc     edi
2315 000093FA 663D1027            <1>       cmp     ax, 10000
2316 000093FE 7201                <1>       jb      short pass_dfc
2317 00009400 47                  <1>       inc     edi
2318                              <1> pass_dfc:
2319                              <1>       ;mov    cx, 10
2320 00009401 886F01              <1>       mov     [edi+1], ch ; 00
2321                              <1> loc_dfc_rediv:
2322                              <1>       ;xor   dx, dx
2323 00009404 30D2                <1>       xor    dl, dl
2324 00009406 66F7F1              <1>       div    cx
2325 00009409 80C230              <1>       add    dl, '0'
2326 0000940C 8817                <1>       mov    [edi], dl
2327 0000940E 4F                  <1>       dec    edi
2328 0000940F 6609C0              <1>       or     ax, ax
2329 00009412 75F0                <1>       jnz    short loc_dfc_rediv
2330                              <1>
2331 00009414 BF[C4930100]        <1>       mov     edi, TFS_Dec_End
2332                              <1>         ;mov    byte [edi], 0
2333 00009419 A1[B2930100]        <1>       mov     eax, [Total_FSize]
2334                              <1>       ;mov    ecx, 10
2335                              <1> rediv_tfs_hex:
2336                              <1>       ;sub   edx, edx
2337 0000941E 28D2                <1>       sub    dl, dl
2338 00009420 F7F1                <1>       div    ecx
2339 00009422 80C230              <1>       add    dl, '0'
2340 00009425 4F                  <1>       dec     edi
2341 00009426 8817                <1>       mov     [edi], dl
2342 00009428 21C0                <1>       and    eax, eax
2343 0000942A 75F2                <1>       jnz    short rediv_tfs_hex
2344                              <1>
2345 0000942C 893D[B6930100]      <1>       mov    [TFS_Dec_Begin], edi
2346 00009432 BE[28440100]        <1>       mov    esi, Decimal_File_Count_Header
2347 00009437 E819E1FFFF          <1>       call   print_msg
2348 0000943C BE[30440100]        <1>       mov    esi, str_files
2349 00009441 E80FE1FFFF          <1>       call   print_msg
2350 00009446 BE[41440100]        <1>       mov    esi, str_dirs
2351 0000944B E805E1FFFF          <1>       call   print_msg
2352 00009450 8B35[B6930100]      <1>       mov    esi, [TFS_Dec_Begin]
2353 00009456 E8FAE0FFFF          <1>       call   print_msg
2354 0000945B BE[52440100]        <1>       mov    esi, str_bytes
2355 00009460 E8F0E0FFFF          <1>       call   print_msg
2356                              <1>
2357 00009465 C3                  <1>       retn
2358                              <1>
2359                              <1> pause_dir_scroll:
2360 00009466 28E4                <1>       sub    ah, ah
2361 00009468 E8787AFFFF          <1>       call   int16h
2362 0000946D 3C1B                <1>       cmp    al, 1Bh
2363 0000946F 0F8428FFFFFF        <1>         je      loc_dir_ok
2364 00009475 C605[C5930100]10    <1>       mov    byte [PrintDir_RowCounter], 16 ; Reset counter
```

```
2365 0000947C E911FFFFFF          <1>         jmp     loc_next_entry
2366                              <1>
2367                              <1> find_first_file:
2368                              <1>         ; 11/02/2016
2369                              <1>         ; 10/02/2016
2370                              <1>         ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)
2371                              <1>         ; 09/10/2011
2372                              <1>         ; 17/09/2009
2373                              <1>         ; 2005
2374                              <1>         ; INPUT ->
2375                              <1>         ;     ESI = ASCIIZ File/Dir Name Address (in Current Directory)
2376                              <1>         ;     AL = Attributes AND mask (The AND result must be equal to AL)
2377                              <1>         ;           bit 0 = Read Only
2378                              <1>         ;           bir 1 = Hidden
2379                              <1>         ;           bit 2 = System
2380                              <1>         ;           bit 3 = Volume Label
2381                              <1>         ;           bit 4 = Directory
2382                              <1>         ;           bit 5 = Archive
2383                              <1>         ;           bit 6 = Reserved, must be 0
2384                              <1>         ;           bit 7 = Reserved, must be 0
2385                              <1>         ;       AH = Attributes Negative AND mask (The AND result must be ZERO)
2386                              <1>         ;
2387                              <1>         ; OUTPUT ->
2388                              <1>         ;     CF = 1 -> Error, Error Code in EAX (AL)
2389                              <1>         ;     CF = 0 ->
2390                              <1>         ;          ESI = Directory Entry (FindFile_DirEntry) Location
2391                              <1>         ;          EDI = Directory Buffer Directory Entry Location
2392                              <1>         ;          EAX = File Size
2393                              <1>         ;           BL = Attributes of The File/Directory
2394                              <1>         ;           BH = Long Name Yes/No Status (>0 is YES)
2395                              <1>         ;             DX > 0 : Ambiguous filename chars are used
2396                              <1>         ;
2397                              <1>         ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)
2398                              <1>
2399 00009481 66A3[76930100]      <1>         mov    [FindFile_AttributesMask], ax
2400 00009487 BF[78930100]        <1>         mov    edi, FindFile_DirEntry ; TR-DOS Fullfilename formatted buffer
2401 0000948C 31C0                <1>         xor    eax, eax
2402 0000948E B90B000000          <1>         mov    ecx, 11
2403 00009493 F3AB                <1>         rep    stosd  ; 44 bytes
2404                              <1>         ;stosw       ; +2 bytes
2405                              <1>
2406 00009495 BF[68930100]        <1>         mov    edi, FindFile_Name ; FFF structure, offset 66
2407 0000949A 39FE                <1>         cmp    esi, edi
2408 0000949C 7408                <1>         je     short loc_fff_mfn_ok
2409 0000949E 89FA                <1>         mov    edx, edi
2410                              <1>         ; move 13 bytes
2411 000094A0 A5                  <1>         movsd
2412 000094A1 A5                  <1>         movsd
2413 000094A2 A5                  <1>         movsd
2414 000094A3 AA                  <1>         stosb
2415 000094A4 89D6                <1>         mov    esi, edx
2416                              <1> loc_fff_mfn_ok:
2417 000094A6 BF[17930100]        <1>         mov    edi, Dir_Entry_Name ; Dir Entry Format File Name
2418 000094AB E8D7200000          <1>         call   convert_file_name
2419 000094B0 89FE                <1>         mov    esi, edi ; offset Dir_Entry_Name
2420                              <1>
2421 000094B2 66A1[76930100]      <1>         mov    ax, [FindFile_AttributesMask]
2422                              <1>         ;xor   ecx, ecx
2423 000094B8 30C9                <1>         xor    cl, cl
2424 000094BA E8D11D0000          <1>         call   locate_current_dir_file
2425 000094BF 726E                <1>         jc     short loc_fff_retn
2426                              <1>         ; EDI = Directory Entry
2427                              <1>         ; EBX = Directory Buffer Entry Index/Number
2428                              <1>
2429                              <1> loc_fff_fnf_ln_check:
2430 000094C1 30ED                <1>         xor    ch, ch
2431 000094C3 80F60F              <1>         xor    dh, 0Fh
2432 000094C6 7408                <1>         jz     short loc_fff_longname_yes
2433 000094C8 882D[75930100]      <1>         mov    [FindFile_LongNameYes], ch ; 0
2434 000094CE EB0C                <1>         jmp    short loc_fff_longname_no
2435                              <1>
2436                              <1> loc_fff_longname_yes:
2437                              <1>         ;inc   byte [FindFile_LongNameYes]
2438 000094D0 8A0D[82920100]      <1>         mov    cl, [LFN_EntryLength]
2439 000094D6 880D[75930100]      <1>         mov    [FindFile_LongNameEntryLength], cl ; FindFile_LongNameYes
2440                              <1>
2441                              <1> loc_fff_longname_no:
2442                              <1>         ;mov   bx, [DirBuff_CurrentEntry]
2443 000094DC 66891D[A0930100]    <1>         mov    [FindFile_DirEntryNumber], bx
2444 000094E3 6689C2              <1>         mov    dx, ax ; Ambiguous Filename chars used sign > 0
2445                              <1>
2446 000094E6 A0[868A0100]        <1>         mov    al, [Current_Drv]
2447 000094EB A2[26930100]        <1>         mov    [FindFile_Drv], al
2448                              <1>
2449 000094F0 A1[808A0100]        <1>         mov    eax, [Current_Dir_FCluster]
2450 000094F5 A3[98930100]        <1>         mov    [FindFile_DirFirstCluster], eax
2451                              <1>
2452 000094FA A1[B1910100]        <1>         mov    eax, [DirBuff_Cluster]
2453 000094FF A3[9C930100]        <1>         mov    [FindFile_DirCluster], eax
2454                              <1>
2455 00009504 66FF05[A2930100]    <1>         inc    word [FindFile_MatchCounter]
2456                              <1>
2457 0000950B 89FB                <1>         mov    ebx, edi
2458 0000950D 89FE                <1>         mov    esi, edi
2459 0000950F BF[78930100]        <1>         mov    edi, FindFile_DirEntry
2460 00009514 89F8                <1>         mov    eax, edi
2461 00009516 B108                <1>         mov    cl, 8
2462 00009518 F3A5                <1>         rep    movsd
2463 0000951A 89C6                <1>         mov    esi, eax
2464 0000951C 89DF                <1>         mov    edi, ebx
2465                              <1>
2466 0000951E A1[94930100]        <1>         mov    eax, [FindFile_DirEntry+28] ; File Size
2467                              <1>
2468 00009523 8A1D[83930100]      <1>         mov    bl, [FindFile_DirEntry+11] ; File Attributes
2469 00009529 8A3D[75930100]      <1>         mov    bh, [FindFile_LongNameYes]
```

```
2470                                    <1>
2471                                    <1>       ;mov   cx, [DirBuff_EntryCounter]
2472                                    <1>       ;mov   [FindFile_DirEntryNumber], cx
2473                                    <1>       ;mov   cx, [FindFile_DirEntryNumber]
2474                                    <1>       ; ecx = 0
2475                                    <1>
2476                                    <1> loc_fff_retn:
2477 0000952F C3                        <1>       retn
2478                                    <1>
2479                                    <1> find_next_file:
2480                                    <1>       ; 15/10/2016
2481                                    <1>       ; 10/02/2016
2482                                    <1>       ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)
2483                                    <1>       ; 06/02/2011
2484                                    <1>       ; 17/09/2009
2485                                    <1>       ; 2005
2486                                    <1>       ; INPUT ->
2487                                    <1>       ;     NONE, Find First File Parameters
2488                                    <1>       ; OUTPUT ->
2489                                    <1>       ;     CF = 1 -> Error, Error Code in EAX (AL)
2490                                    <1>       ;     CF = 0 ->
2491                                    <1>       ;         ESI = Directory Entry (FindFile_DirEntry) Location
2492                                    <1>       ;         EDI = Directory Buffer Directory Entry Location
2493                                    <1>       ;         EAX = File Size
2494                                    <1>       ;           BL = Attributes of The File/Directory
2495                                    <1>       ;           BH = Long Name Yes/No Status (>0 is YES)
2496                                    <1>       ;             DX > 0 : Ambiguous filename chars are used
2497                                    <1>       ;
2498                                    <1>       ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)
2499                                    <1>
2500 00009530 66833D[A2930100]00        <1>       cmp    word [FindFile_MatchCounter], 0
2501 00009538 7707                      <1>       ja     short loc_start_search_next_file
2502                                    <1>
2503                                    <1> loc_fnf_stc_retn:
2504 0000953A F9                        <1>       stc
2505                                    <1> loc_fnf_ax12h_retn:
2506 0000953B B80C000000                <1>       mov    eax, 12 ; No More files
2507                                    <1> ;loc_fnf_retn:
2508 00009540 C3                        <1>       retn
2509                                    <1>
2510                                    <1> loc_start_search_next_file:
2511 00009541 668B1D[A0930100]          <1>       mov    bx, [FindFile_DirEntryNumber]
2512 00009548 6643                      <1>       inc    bx
2513 0000954A 663B1D[AF910100]          <1>       cmp    bx, [DirBuff_LastEntry]
2514 00009551 7719                      <1>       ja     short loc_cont_search_next_file
2515                                    <1>
2516                                    <1> loc_fnf_search:
2517 00009553 BE[17930100]              <1>       mov    esi, Dir_Entry_Name
2518 00009558 66A1[76930100]            <1>       mov    ax, [FindFile_AttributesMask]
2519 0000955E 6631C9                    <1>       xor    cx, cx
2520 00009561 E82E1E0000                <1>       call   find_directory_entry
2521 00009566 0F8355FFFFFF              <1>         jnc     loc_fff_fnf_ln_check
2522                                    <1>
2523                                    <1> loc_cont_search_next_file:
2524 0000956C 31DB                      <1>       xor    ebx, ebx
2525 0000956E 8A3D[868A0100]            <1>       mov    bh, [Current_Drv]
2526 00009574 BE00010900                <1>       mov    esi, Logical_DOSDisks
2527 00009579 01DE                      <1>       add    esi, ebx
2528                                    <1>
2529 0000957B 803D[848A0100]00          <1>       cmp    byte [Current_Dir_Level], 0
2530 00009582 7608                      <1>       jna    short loc_fnf_check_FAT_type
2531 00009584 807E0301                  <1>       cmp    byte [esi+LD_FATType], 1
2532 00009588 72B1                      <1>       jb     short loc_fnf_ax12h_retn
2533 0000958A EB06                      <1>       jmp    short loc_fnf_check_next_cluster
2534                                    <1>
2535                                    <1> loc_fnf_check_FAT_type:
2536 0000958C 807E0303                  <1>       cmp    byte [esi+LD_FATType], 3
2537 00009590 72A9                      <1>       jb     short loc_fnf_ax12h_retn
2538                                    <1>
2539                                    <1> loc_fnf_check_next_cluster:
2540 00009592 A1[B1910100]              <1>       mov    eax, [DirBuff_Cluster]
2541 00009597 E8CA370000                <1>       call   get_next_cluster
2542 0000959C 7306                      <1>       jnc    short loc_fnf_load_next_dir_cluster
2543 0000959E 09C0                      <1>       or     eax, eax
2544 000095A0 7498                      <1>       jz     short loc_fnf_stc_retn
2545                                    <1>       ;mov   eax, 17 ;Drive not ready or read error
2546 000095A2 F5                        <1>       cmc    ;stc
2547                                    <1> loc_fnf_retn:
2548 000095A3 C3                        <1>       retn
2549                                    <1>
2550                                    <1> loc_fnf_load_next_dir_cluster:
2551 000095A4 E8A3390000                <1>       call   load_FAT_sub_directory
2552 000095A9 72F8                      <1>       jc     short loc_fnf_retn
2553 000095AB 6631DB                    <1>       xor    bx, bx
2554 000095AE 66891D[A0930100]          <1>       mov    [FindFile_DirEntryNumber], bx
2555 000095B5 EB9C                      <1>       jmp    short loc_fnf_search
2556                                    <1>
2557                                    <1> get_and_print_longname:
2558                                    <1>       ; 16/10/2016
2559                                    <1>       ; 13/02/2016 (TRDOS 386 = TRDOS v2.0)
2560                                    <1>       ; 24/01/2010
2561                                    <1>       ; 17/10/2009 (CMD_INTR.ASM, 'cmp_cmd_longname')
2562                                    <1> get_longname_fchar:
2563 000095B7 803E20                    <1>       cmp    byte [esi], 20h
2564 000095BA 7701                      <1>       ja     short loc_find_longname
2565                                    <1>       ;jb    short loc_longname_retn
2566                                    <1>       ;inc   esi
2567                                    <1>       ;je    short get_longname_fchar
2568                                    <1> ;loc_longname_retn:
2569 000095BC C3                        <1>       retn
2570                                    <1> loc_find_longname:
2571 000095BD E839210000                <1>       call   find_longname
2572 000095C2 7328                      <1>       jnc    short loc_print_longname
2573                                    <1>
2574 000095C4 08C0                      <1>       or     al, al
```

```
2575 000095C6 741A              <1>       jz      short loc_longname_not_found
2576                            <1>
2577                            <1>       ; 16/10/2016 (15h -> 15, 17)
2578 000095C8 3C0F              <1>       cmp     al, 15
2579 000095CA 0F84B6F7FFFF      <1>       je      cd_drive_not_ready ; drive not ready
2580                            <1>                                  ; or
2581 000095D0 3C11              <1>       cmp     al, 17             ; read error
2582 000095D2 0F84AEF7FFFF      <1>       je      cd_drive_not_ready
2583                            <1>
2584                            <1> loc_ln_file_dir_not_found:
2585 000095D8 BE[7C430100]      <1>       mov     esi, Msg_File_Directory_Not_Found
2586                            <1>       ;call  print_msg
2587                            <1>         ;retn
2588 000095DD E973DFFFFF        <1>       jmp     print_msg
2589                            <1>
2590                            <1> loc_longname_not_found:
2591 000095E2 BE[9B430100]      <1>         mov     esi, Msg_LongName_Not_Found
2592                            <1>       ;call  print_msg
2593                            <1>         ;retn
2594 000095E7 E969DFFFFF        <1>       jmp     print_msg
2595                            <1>
2596                            <1> loc_print_longname:
2597                            <1>       ;mov   esi, LongFileName
2598 000095EC BF[868B0100]      <1>       mov     edi, TextBuffer
2599 000095F1 57                <1>       push    edi
2600 000095F2 3C00              <1>       cmp     al, 0
2601 000095F4 7708              <1>       ja      short loc_print_longname_1
2602                            <1> loc_print_FS_longname: ; Singlix FS (64 byte ASCIIZ file name)
2603 000095F6 AC                <1>       lodsb
2604 000095F7 AA                <1>       stosb
2605 000095F8 08C0              <1>       or      al, al
2606 000095FA 75FA              <1>       jnz     short loc_print_FS_longname
2607 000095FC EB07              <1>       jmp     short loc_print_longname_2
2608                            <1>       ;
2609                            <1> loc_print_longname_1: ; MS Windows long name (UNICODE chars)
2610 000095FE 66AD              <1>       lodsw
2611 00009600 AA                <1>       stosb
2612 00009601 08C0              <1>       or      al, al
2613 00009603 75F9              <1>       jnz     short loc_print_longname_1
2614                            <1>       ;
2615                            <1> loc_print_longname_2:
2616 00009605 5E                <1>       pop     esi
2617 00009606 E84ADFFFFF        <1>       call    print_msg
2618 0000960B BE[EF4C0100]      <1>       mov     esi, nextline
2619                            <1>       ;call  print_msg
2620                            <1>       ;retn
2621 00009610 E940DFFFFF        <1>       jmp     print_msg
2622                            <1>
2623                            <1> show_file:
2624                            <1>       ; 18/02/2016
2625                            <1>       ; 17/02/2016
2626                            <1>       ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
2627                            <1>       ; 13/09/2011 (CMD_INTR.ASM, 'cmp_cmd_show')
2628                            <1>       ; 08/11/2009
2629                            <1>
2630                            <1> loc_show_parse_path_name:
2631 00009615 BF[26930100]      <1>       mov     edi, FindFile_Drv
2632 0000961A E833200000        <1>       call    parse_path_name
2633 0000961F 0F824CF9FFFF      <1>       jc      loc_cmd_failed
2634                            <1>
2635                            <1> loc_show_check_filename_exists:
2636 00009625 BE[68930100]      <1>       mov     esi, FindFile_Name
2637 0000962A 803E20            <1>       cmp     byte [esi], 20h
2638 0000962D 0F863EF9FFFF      <1>       jna     loc_cmd_failed
2639                            <1>
2640                            <1>       ; 15/02/2016 (invalid file name check)
2641 00009633 E807020000        <1>       call    check_filename
2642 00009638 730A              <1>       jnc     short loc_show_change_drv
2643                            <1>
2644 0000963A BE[68440100]      <1>       mov     esi, Msg_invalid_name_chars
2645 0000963F E911DFFFFF        <1>       jmp     print_msg
2646                            <1>
2647                            <1> loc_show_change_drv:
2648 00009644 8A35[868A0100]    <1>       mov     dh, [Current_Drv]
2649 0000964A 8835[E2910100]    <1>       mov     [RUN_CDRV], dh
2650 00009650 8A15[26930100]    <1>       mov     dl, [FindFile_Drv]
2651 00009656 38F2              <1>       cmp     dl, dh
2652 00009658 740B              <1>       je      short loc_show_change_directory
2653 0000965A E87FEAFFFF        <1>       call    change_current_drive
2654                            <1>       ;jc    loc_file_rw_cmd_failed
2655 0000965F 0F8237F9FFFF      <1>       jc      loc_run_cmd_failed
2656                            <1>
2657                            <1> loc_show_change_directory:
2658 00009665 803D[27930100]20  <1>       cmp     byte [FindFile_Directory], 20h
2659 0000966C 7618              <1>       jna     short loc_findload_showfile
2660                            <1>
2661 0000966E FE05[51400100]    <1>       inc     byte [Restore_CDIR]
2662 00009674 BE[27930100]      <1>       mov     esi, FindFile_Directory
2663 00009679 30E4              <1>       xor     ah, ah ; CD_COMMAND sign -> 0
2664 0000967B E8BC190000        <1>       call    change_current_directory
2665                            <1>       ;jc    loc_file_rw_cmd_failed
2666 00009680 0F8216F9FFFF      <1>       jc      loc_run_cmd_failed
2667                            <1>
2668                            <1> ;loc_show_change_prompt_dir_string:
2669                            <1>       ;call  change_prompt_dir_string
2670                            <1>
2671                            <1> loc_findload_showfile:
2672                            <1>       ; 15/02/2016
2673 00009686 BE[68930100]      <1>       mov     esi, FindFile_Name
2674 0000968B BF[17930100]      <1>       mov     edi, Dir_Entry_Name ; Dir Entry Format File Name
2675 00009690 E8F21E0000        <1>       call    convert_file_name
2676 00009695 89FE              <1>       mov     esi, edi ; offset Dir_Entry_Name
2677                            <1>
2678 00009697 28C0              <1>       sub     al, al ; Attrib AND mask = 0
2679                            <1>       ; Directory attribute : 10h
```

```
2680                            <1>       ; Volume name attribute: 8h
2681 00009699 B418              <1>       mov    ah, 00011000b ; 18h (Attrib NAND, AND --> zero mask)
2682                            <1>       ;
2683 0000969B 6631C9            <1>       xor    cx, cx
2684 0000969E E8ED1B0000        <1>       call   locate_current_dir_file
2685                            <1>       ;jc    loc_file_rw_cmd_failed
2686 000096A3 0F82F3F8FFFF      <1>       jc     loc_run_cmd_failed
2687                            <1>
2688                            <1> loc_show_load_file:
2689                            <1>       ; EDI = Directory Entry
2690 000096A9 668B4714          <1>       mov    ax, [edi+DirEntry_FstClusHI] ; First Cluster High Word
2691 000096AD C1E010            <1>       shl    eax, 16
2692 000096B0 668B471A          <1>       mov    ax, [edi+DirEntry_FstClusLO] ; First Cluster Low Word
2693 000096B4 A3[D0930100]      <1>       mov    [Show_Cluster], eax
2694 000096B9 8B471C            <1>       mov    eax, [edi+DirEntry_FileSize] ; File Size
2695 000096BC 21C0              <1>       and    eax, eax ; Empty file !
2696 000096BE 0F8491000000      <1>         jz     end_of_show_file
2697 000096C4 A3[D4930100]      <1>       mov    [Show_FileSize], eax
2698 000096C9 31C0              <1>       xor    eax, eax
2699 000096CB A3[D8930100]      <1>       mov    [Show_FilePointer], eax ; 0
2700 000096D0 66A3[DC930100]    <1>       mov    [Show_ClusterPointer], ax ; 0
2701 000096D6 29DB              <1>       sub    ebx, ebx
2702 000096D8 8A3D[868A0100]    <1>       mov    bh, [Current_Drv]
2703 000096DE BE00010900        <1>       mov    esi, Logical_DOSDisks
2704 000096E3 01DE              <1>       add    esi, ebx
2705 000096E5 8935[CC930100]    <1>       mov    [Show_LDDDT], esi ; Logical DOS Drv Description Table addr
2706                            <1>
2707 000096EB 807E0300          <1>       cmp    byte [esi+LD_FATType], 0
2708 000096EF 7713              <1>       ja     short loc_show_calculate_cluster_size
2709                            <1>       ; Singlix FS
2710                            <1>       ; First Cluster Number is FDT number (in compatibility buffer)
2711 000096F1 8B15[D0930100]    <1>       mov    edx, [Show_Cluster] ; Compatibility dir. buffer value (FDT)
2712 000096F7 8915[C8930100]    <1>       mov    [Show_FDT], edx
2713 000096FD 31C0              <1>       xor    eax, eax
2714 000096FF A3[D0930100]      <1>       mov    [Show_Cluster], eax ; Sector index  = 0
2715                            <1>                                  ; (next time it will be 1)
2716                            <1> loc_show_calculate_cluster_size:
2717 00009704 668B5E11          <1>       mov    bx, [esi+LD_BPB+BPB_BytsPerSec] ; FAT 12-16-32 (512)
2718                            <1>       ; BX = 512 = [esi+LD_FS_BytesPerSec] ; Singlix FS
2719 00009708 8A4613            <1>       mov    al, [esi+LD_BPB+BPB_SecPerClust] ; FAT 12-16-32 (<= 128)
2720                            <1>       ; AL = 1 = [esi+LD_FS_Reserved2] ; SectPerClust for Singlix FS
2721 0000970B F7E3              <1>       mul    ebx
2722                            <1>
2723                            <1>       ;cmp   eax, 65536 ; non-compatible (very big) cluster size
2724                            <1>       ;ja    short end_of_show_file
2725 0000970D 66A3[DE930100]    <1>       mov    [Show_ClusterSize], ax
2726                            <1>
2727                            <1> loc_start_show_file:
2728 00009713 BE[EF4C0100]      <1>       mov    esi, nextline
2729 00009718 E838DEFFFF        <1>       call   print_msg
2730                            <1>
2731 0000971D A1[D0930100]      <1>       mov    eax, [Show_Cluster]
2732 00009722 C605[E0930100]17  <1>       mov    byte [Show_RowCount], 23
2733                            <1>
2734                            <1>       ; 17/02/2016
2735 00009729 8B35[CC930100]    <1>       mov    esi, [Show_LDDDT]
2736                            <1>
2737                            <1> loc_show_next_cluster:
2738                            <1>       ; 15/02/2016
2739 0000972F BB00000700        <1>       mov    ebx, Cluster_Buffer ; 70000h (for current TRDOS 386 version)
2740                            <1>       ; ESI = Logical DOS drv description table address
2741 00009734 E851380000        <1>       call   read_cluster
2742                            <1>       ;jc    loc_file_rw_cmd_failed
2743 00009739 0F825DF8FFFF      <1>       jc     loc_run_cmd_failed
2744                            <1>
2745 0000973F 31DB              <1>       xor    ebx, ebx
2746                            <1> loc_show_next_byte:
2747 00009741 803D[E0930100]00  <1>       cmp    byte [Show_RowCount], 0
2748 00009748 7521              <1>       jne    short pass_show_wait_for_key
2749 0000974A 30E4              <1>       xor    ah, ah
2750 0000974C E89477FFFF        <1>       call   int16h
2751 00009751 3C1B              <1>       cmp    al, 1Bh
2752 00009753 750F              <1>       jne    short pass_exit_show
2753                            <1> end_of_show_file:
2754                            <1> pass_show_file:
2755 00009755 BE[EF4C0100]      <1>       mov    esi, nextline
2756 0000975A E8F6DDFFFF        <1>       call   print_msg
2757 0000975F E94B010000        <1>       jmp    loc_file_rw_restore_retn
2758                            <1>
2759                            <1> pass_exit_show:
2760 00009764 C605[E0930100]14  <1>       mov    byte [Show_RowCount], 20
2761                            <1> pass_show_wait_for_key:
2762 0000976B 81C300000700      <1>       add    ebx, Cluster_Buffer
2763 00009771 8A03              <1>       mov    al, [ebx]
2764 00009773 3C0D              <1>       cmp    al, 0Dh
2765 00009775 0F8590000000      <1>         jne    loc_show_check_tab_space
2766 0000977B FE0D[E0930100]    <1>       dec    byte [Show_RowCount]
2767                            <1> pass_show_dec_rowcount:
2768 00009781 B307              <1>       mov    bl, 7 ; (light gray character color, black background)
2769 00009783 8A3D[EE890100]    <1>       mov    bh, [ACTIVE_PAGE] ; [ptty]
2770 00009789 E86F8BFFFF        <1>       call   _write_tty
2771                            <1> loc_show_check_eof:
2772 0000978E FF05[D8930100]    <1>       inc    dword [Show_FilePointer]
2773 00009794 A1[D8930100]      <1>       mov    eax, [Show_FilePointer]
2774 00009799 3B05[D4930100]    <1>       cmp    eax, [Show_FileSize]
2775 0000979F 73B4              <1>       jnb    short end_of_show_file
2776 000097A1 66FF05[DC930100]  <1>       inc    word [Show_ClusterPointer]
2777 000097A8 0FB71D[DC930100]  <1>       movzx  ebx, word [Show_ClusterPointer]
2778                            <1>
2779                            <1>       ; 17/02/2016
2780                            <1>       ; (sector boundary -9 bits- check, 512 = 0)
2781 000097AF 66F7C3FF01        <1>         test    bx, 1FFh ;  1 to 511
2782 000097B4 758B              <1>       jnz    short loc_show_next_byte
2783                            <1>
2784                            <1>       ; 16/02/2016
```

```
2785 000097B6 8B35[CC930100]      <1>       mov    esi, [Show_LDDDT]
2786                              <1>       ;
2787 000097BC 807E0300            <1>       cmp    byte [esi+LD_FATType], 0
2788 000097C0 7719                <1>       ja     short loc_show_check_fat_cluster_size
2789                              <1>
2790                              <1>       ; Singlix FS
2791                              <1>       ; 1 sector, more... (cluster size = 1 sector)
2792 000097C2 A1[D0930100]        <1>       mov    eax, [Show_Cluster]
2793 000097C7 40                  <1>       inc    eax
2794 000097C8 A3[D0930100]        <1>       mov    [Show_Cluster], eax
2795                              <1>
2796 000097CD 6621DB              <1>       and    bx, bx ; 65536 -> 0
2797 000097D0 0F856BFFFFFF        <1>         jnz  loc_show_next_byte
2798 000097D6 E954FFFFFF          <1>       jmp     loc_show_next_cluster
2799                              <1>
2800                              <1> loc_show_check_fat_cluster_size:
2801                              <1>       ; 17/02/2016
2802 000097DB 663B1D[DE930100]    <1>       cmp    bx, [Show_ClusterSize] ; cluster size in bytes
2803 000097E2 0F8259FFFFFF        <1>         jb   loc_show_next_byte
2804 000097E8 66C705[DC930100]00- <1>       mov    word [Show_ClusterPointer], 0
2804 000097F0 00                  <1>
2805                              <1>
2806 000097F1 A1[D0930100]        <1>       mov    eax, [Show_Cluster]
2807                              <1>       ;mov   esi, [Show_LDDDT]
2808                              <1> loc_show_get_next_cluster:
2809 000097F6 E86B350000          <1>       call   get_next_cluster
2810                              <1>       ;jc    loc_file_rw_cmd_failed
2811 000097FB 0F829BF7FFFF        <1>       jc     loc_run_cmd_failed
2812                              <1> loc_show_update_ccluster:
2813 00009801 A3[D0930100]        <1>       mov    [Show_Cluster], eax
2814 00009806 E924FFFFFF          <1>         jmp     loc_show_next_cluster
2815                              <1>
2816                              <1> loc_show_check_tab_space:
2817 0000980B 3C09                <1>       cmp    al, 09h
2818 0000980D 0F856EFFFFFF        <1>         jne     pass_show_dec_rowcount
2819                              <1> loc_show_put_tab_space:
2820 00009813 8A3D[EE890100]      <1>       mov    bh, [ACTIVE_PAGE] ; [ptty]
2821 00009819 E85B87FFFF          <1>       call   get_cpos
2822                              <1>       ; dl = cursor column
2823 0000981E 80E207              <1>       and    dl, 7 ; 18/02/2016
2824                              <1>       ;shr   bh, 1 ; [ACTIVE_PAGE]
2825 00009821 8A3D[EE890100]      <1>       mov    bh, [ACTIVE_PAGE]
2826 00009827 B307                <1>       mov    bl, 7 ; color attribute
2827                              <1> loc_show_put_space_chars:
2828 00009829 B020                <1>       mov    al, 20h ; space
2829                              <1>       ;mov   bh, [ACTIVE_PAGE] ; [ptty]
2830                              <1>       ;mov   bl, 7 ; color attribute
2831                              <1>       ;push  dx
2832 0000982B 52                  <1>       push   edx ; 29/12/2017
2833 0000982C E8CC8AFFFF          <1>       call   _write_tty
2834 00009831 5A                  <1>       pop    edx ; 29/12/2017
2835                              <1>       ;pop   dx
2836                              <1>       ; 18/02/2016
2837 00009832 80FA07              <1>       cmp    dl, 7
2838 00009835 0F8353FFFFFF        <1>       jnb    loc_show_check_eof
2839 0000983B FEC2                <1>       inc    dl
2840 0000983D EBEA                <1>       jmp    short loc_show_put_space_chars
2841                              <1>
2842                              <1> check_filename:
2843                              <1>       ; 10/10/2016
2844                              <1>       ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
2845                              <1>       ; 07/08/2010 (FILE.ASM, 'proc_check_filename')
2846                              <1>       ; 10/07/2010
2847                              <1>       ; Derived from 'proc_check_filename'
2848                              <1>       ; in the old TRDOS.ASM (09/02/2005).
2849                              <1>       ;
2850                              <1>       ; INPUT ->
2851                              <1>       ;      ESI = Dot File Name Location
2852                              <1>       ; OUTPUT ->
2853                              <1>       ;      cf = 1 -> error code in AL
2854                              <1>       ;           AL = ERR_INV_FILE_NAME (=26)
2855                              <1>       ;              Invalid file name chars
2856                              <1>       ;      cf = 0 -> valid file name
2857                              <1>       ;
2858                              <1>       ;(EAX, ECX, EDI will be changed)
2859                              <1>
2860                              <1> check_invalid_filename_chars:
2861                              <1>       ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
2862                              <1>       ; 10/07/2010 (FILE.ASM, 'proc_check_invalid_filename_chars')
2863                              <1>       ; 10/02/2010
2864                              <1>       ; Derived from 'proc_check_invalid_filename_chars'
2865                              <1>       ; in the old TRDOS.ASM (09/02/2005).
2866                              <1>       ;
2867                              <1>       ; INPUT ->
2868                              <1>       ;      ESI = ASCIIZ FileName
2869                              <1>       ; OUTPUT ->
2870                              <1>       ;      cf = 1 -> invalid
2871                              <1>       ;      cf = 0 -> valid
2872                              <1>       ;
2873                              <1>       ;(EAX, ECX, EDI will be changed)
2874                              <1>
2875 0000983F 56                  <1>       push   esi
2876                              <1>
2877 00009840 BF[50410100]        <1>         mov     edi, invalid_fname_chars
2878 00009845 AC                  <1>       lodsb
2879                              <1> check_filename_next_char:
2880 00009846 B914000000          <1>       mov    ecx, sizeInvFnChars
2881 0000984B BF[50410100]        <1>       mov    edi, invalid_fname_chars
2882                              <1> loc_scan_invalid_filename_char:
2883 00009850 AE                  <1>       scasb
2884 00009851 741F                <1>       je     short loc_invalid_filename_stc
2885 00009853 E2FB                <1>       loop   loc_scan_invalid_filename_char
2886 00009855 AC                  <1>       lodsb
2887 00009856 3C1F                <1>       cmp    al, 1Fh  ; 20h and above
2888 00009858 77EC                <1>       ja     short check_filename_next_char
```

```
2889                                    <1>
2890                                    <1> check_filename_dot:
2891 0000985A 8B3424                   <1>         mov     esi, [esp]
2892                                    <1>
2893 0000985D B421                     <1>         mov     ah, 21h
2894 0000985F B908000000               <1>         mov     ecx, 8
2895                                    <1> loc_check_filename_next_char:
2896 00009864 AC                       <1>         lodsb
2897 00009865 3C2E                     <1>         cmp     al, 2Eh
2898 00009867 7511                     <1>         jne     short pass_check_fn_dot_check
2899                                    <1> loc_check_filename_ext_0:
2900 00009869 AC                       <1>         lodsb
2901 0000986A 38E0                     <1>         cmp     al, ah ; 21h
2902 0000986C 7205                     <1>         jb      short loc_invalid_filename
2903 0000986E 3C2E                     <1>         cmp     al, 2Eh
2904 00009870 7519                     <1>         jne     short loc_check_filename_ext_1
2905                                    <1>
2906                                    <1> loc_invalid_filename_stc:
2907                                    <1> loc_check_fn_stc_rtn:
2908 00009872 F9                       <1>         stc
2909                                    <1> loc_invalid_filename:
2910                                    <1>         ; 10/10/2016 (0Bh -> 26)
2911 00009873 B81A000000               <1>         mov     eax, ERR_INV_FILE_NAME ; (=26)
2912                                    <1>         ; Invalid file name chars
2913                                    <1> loc_check_fn_rtn:
2914 00009878 5E                       <1>         pop     esi
2915 00009879 C3                       <1>         retn
2916                                    <1>
2917                                    <1> pass_check_fn_dot_check:
2918 0000987A 38E0                     <1>         cmp     al, ah ; 21h
2919 0000987C 7224                     <1>         jb      short loc_check_fn_clc_rtn
2920 0000987E E2E4                     <1>         loop    loc_check_filename_next_char
2921 00009880 AC                       <1>         lodsb
2922 00009881 38E0                     <1>         cmp     al, ah ; 21h
2923 00009883 721D                     <1>         jb      short loc_check_fn_clc_rtn
2924 00009885 3C2E                     <1>         cmp     al, 2Eh
2925 00009887 75E9                     <1>         jne     short loc_check_fn_stc_rtn
2926 00009889 EBDE                     <1>         jmp     short loc_check_filename_ext_0
2927                                    <1>
2928                                    <1> loc_check_filename_ext_1:
2929 0000988B AC                       <1>         lodsb
2930 0000988C 38E0                     <1>         cmp     al, ah ; 21h
2931 0000988E 7212                     <1>         jb      short loc_check_fn_clc_rtn
2932 00009890 3C2E                     <1>         cmp     al, 2Eh
2933 00009892 74DE                     <1>         je      short loc_check_fn_stc_rtn
2934 00009894 AC                       <1>         lodsb
2935 00009895 38E0                     <1>         cmp     al, ah ; 21h
2936 00009897 7209                     <1>         jb      short loc_check_fn_clc_rtn
2937 00009899 3C2E                     <1>         cmp     al, 2Eh
2938 0000989B 74D5                     <1>         je      short loc_check_fn_stc_rtn
2939 0000989D AC                       <1>         lodsb
2940 0000989E 38E0                     <1>         cmp     al, ah ; 21h
2941 000098A0 73D0                     <1>         jnb     short loc_check_fn_stc_rtn
2942                                    <1>
2943                                    <1> loc_check_fn_clc_rtn:
2944 000098A2 5E                       <1>         pop     esi
2945 000098A3 F8                       <1>         clc
2946 000098A4 C3                       <1>         retn
2947                                    <1>
2948                                    <1> loc_print_deleted_message:
2949 000098A5 BE[3D450100]             <1>         mov     esi, Msg_Deleted
2950 000098AA E8A6DCFFFF               <1>         call    print_msg
2951                                    <1>
2952                                    <1>         ;clc
2953                                    <1>
2954                                    <1> loc_file_rw_restore_retn:
2955                                    <1>         ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
2956                                    <1>         ; 28/02/2010 (CMD_INTR.ASM)
2957                                    <1> loc_file_rw_cmd_failed:
2958 000098AF 9C                       <1>         pushf
2959 000098B0 E84FF7FFFF               <1>         call    restore_cdir_after_cmd_fail
2960 000098B5 9D                       <1>         popf
2961 000098B6 720D                     <1>         jc      short loc_file_rw_check_write_fault
2962 000098B8 C3                       <1>         retn
2963                                    <1>
2964                                    <1> loc_permission_denied:
2965                                    <1>         ; 27/02/2016
2966 000098B9 BE[4A450100]             <1>         mov     esi, Msg_Permission_Denied
2967 000098BE E892DCFFFF               <1>         call    print_msg
2968 000098C3 EBEA                     <1>         jmp     short loc_file_rw_restore_retn
2969                                    <1>
2970                                    <1> loc_file_rw_check_write_fault:
2971                                    <1>         ;cmp    al, 1Dh ; Write Fault
2972 000098C5 3C12                     <1>           cmp   al, 18 ; 05/11/2016
2973 000098C7 0F85D4F6FFFF             <1>         jne     loc_run_cmd_failed_cmp_al
2974 000098CD BE[31430100]             <1>         mov     esi, Msg_Not_Ready_Write_Err
2975                                    <1>         ;call   print_msg
2976                                    <1>         ;retn
2977 000098D2 E97EDCFFFF               <1>         jmp     print_msg
2978                                    <1>
2979                                    <1> make_directory:
2980                                    <1>         ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)
2981                                    <1>         ; 12/03/2011 (CMD_INTR.ASM, 'cmp_cmd_mkdir')
2982                                    <1>         ; 14/08/2010
2983                                    <1>         ; 10/07/2010
2984                                    <1>         ; 29/11/2009
2985                                    <1>         ;
2986                                    <1> get_mkdir_fchar:
2987                                    <1>         ; esi = directory name
2988 000098D7 803E20                   <1>         cmp     byte [esi], 20h
2989 000098DA 7701                     <1>           ja    short loc_mkdir_parse_path_name
2990                                    <1>
2991                                    <1> loc_mkdir_nodirname_retn:
2992 000098DC C3                       <1>         retn
2993                                    <1>
```

```
2994                                    <1> loc_mkdir_parse_path_name:
2995 000098DD BF[26930100]             <1>       mov   edi, FindFile_Drv
2996 000098E2 E86B1D0000               <1>         call    parse_path_name
2997 000098E7 0F8284F6FFFF             <1>       jc    loc_cmd_failed
2998                                    <1>
2999                                    <1> loc_mkdir_check_dirname_exists:
3000 000098ED BE[68930100]             <1>       mov   esi, FindFile_Name
3001 000098F2 803E20                   <1>       cmp   byte [esi], 20h
3002 000098F5 0F8676F6FFFF             <1>       jna   loc_cmd_failed
3003 000098FB 8935[E4930100]           <1>       mov   [DelFile_FNPointer], esi
3004 00009901 E839FFFFFF               <1>       call  check_filename
3005 00009906 7259                     <1>       jc    short loc_mkdir_invalid_dir_name_chars
3006                                    <1>
3007                                    <1> loc_mkdir_drv:
3008 00009908 8A35[868A0100]           <1>       mov   dh, [Current_Drv]
3009 0000990E 8835[E2910100]           <1>       mov   [RUN_CDRV], dh
3010                                    <1>
3011 00009914 8A15[26930100]           <1>       mov   dl, [FindFile_Drv]
3012 0000991A 38F2                     <1>       cmp   dl, dh
3013 0000991C 7407                     <1>       je    short loc_mkdir_change_directory
3014                                    <1>
3015 0000991E E8BBE7FFFF               <1>       call  change_current_drive
3016 00009923 728A                     <1>       jc    loc_file_rw_cmd_failed
3017                                    <1>
3018                                    <1> loc_mkdir_change_directory:
3019 00009925 803D[27930100]20         <1>       cmp   byte [FindFile_Directory], 20h
3020 0000992C 7614                     <1>       jna   short loc_mkdir_find_directory
3021                                    <1>
3022 0000992E FE05[51400100]           <1>       inc   byte [Restore_CDIR]
3023 00009934 BE[27930100]             <1>       mov   esi, FindFile_Directory
3024 00009939 30E4                     <1>       xor   ah, ah ; CD_COMMAND sign -> 0
3025 0000993B E8FC160000               <1>       call  change_current_directory
3026 00009940 722E                     <1>       jc    short loc_mkdir_check_error_code
3027                                    <1>
3028                                    <1> ;loc_mkdir_change_prompt_dir_string:
3029                                    <1>       ;call  change_prompt_dir_string
3030                                    <1>
3031                                    <1> loc_mkdir_find_directory:
3032                                    <1>       ;mov   esi, FindFile_Name
3033 00009942 8B35[E4930100]           <1>       mov   esi, [DelFile_FNPointer]
3034                                    <1>       ;xor   eax, eax
3035 00009948 6631C0                   <1>       xor   ax, ax ; any name (dir, file, volume)
3036 0000994B E831FBFFFF               <1>       call  find_first_file
3037 00009950 721E                     <1>       jc    short loc_mkdir_check_error_code
3038                                    <1>
3039                                    <1> loc_mkdir_directory_found:
3040 00009952 BE[95440100]             <1>       mov   esi, Msg_Name_Exists
3041 00009957 E8F9DBFFFF               <1>       call  print_msg
3042                                    <1>
3043 0000995C E94EFFFFFF               <1>         jmp     loc_file_rw_restore_retn
3044                                    <1>
3045                                    <1> loc_mkdir_invalid_dir_name_chars:
3046 00009961 BE[68440100]             <1>       mov   esi, Msg_invalid_name_chars
3047 00009966 E8EADBFFFF               <1>       call  print_msg
3048                                    <1>
3049 0000996B E93FFFFFFF               <1>         jmp     loc_file_rw_restore_retn
3050                                    <1>
3051                                    <1> loc_mkdir_check_error_code:
3052 00009970 3C02                     <1>       cmp   al, 2
3053                                    <1>       ;je    short loc_mkdir_directory_not_found
3054 00009972 7406                     <1>       je    short loc_mkdir_ask_for_yes_no
3055 00009974 F9                       <1>       stc
3056 00009975 E935FFFFFF               <1>         jmp     loc_file_rw_cmd_failed
3057                                    <1>
3058                                    <1> loc_mkdir_directory_not_found:
3059                                    <1> loc_mkdir_ask_for_yes_no:
3060 0000997A BE[B6440100]             <1>       mov   esi, Msg_DoYouWantMkdir
3061 0000997F E8D1DBFFFF               <1>       call  print_msg
3062 00009984 8B35[E4930100]           <1>       mov   esi, [DelFile_FNPointer]
3063 0000998A E8C6DBFFFF               <1>       call  print_msg
3064 0000998F BE[D5440100]             <1>       mov   esi, Msg_YesNo
3065 00009994 E8BCDBFFFF               <1>       call  print_msg
3066                                    <1>
3067 00009999 C605[DF440100]20         <1>       mov   byte [Y_N_nextline], 20h
3068                                    <1>
3069                                    <1> loc_mkdir_ask_again:
3070 000099A0 30E4                     <1>       xor   ah, ah
3071 000099A2 E83E75FFFF               <1>       call  int16h
3072 000099A7 3C1B                     <1>       cmp   al, 1Bh
3073                                    <1>       ;je    short loc_do_not_make_directory
3074 000099A9 7439                     <1>       je    short loc_mkdir_y_n_escape
3075 000099AB 24DF                     <1>       and   al, 0DFh ; y -> Y, n -> N
3076 000099AD 3C59                     <1>       cmp   al, 'Y' ; 'yes'
3077 000099AF 7404                     <1>       je    short loc_mkdir_yes_make_directory
3078 000099B1 3C4E                     <1>       cmp   al, 'N' ; 'no'
3079 000099B3 75EB                     <1>       jne   short loc_mkdir_ask_again
3080                                    <1>
3081                                    <1> loc_do_not_make_directory:
3082                                    <1> loc_mkdir_yes_make_directory:
3083 000099B5 E82E000000               <1>       call  y_n_answer ; 29/12/2017
3084                                    <1>       ;cmp   al, 'Y' ; 'yes'
3085                                    <1>       ;cmc
3086                                    <1>         ;jnc loc_file_rw_restore_retn
3087 000099BA 3C4E                     <1>       cmp   al, 'N' ; 'no'
3088 000099BC 0F84EDFEFFFF             <1>         je  loc_file_rw_restore_retn
3089                                    <1>
3090                                    <1> loc_mkdir_call_make_sub_directory:
3091 000099C2 8B35[E4930100]           <1>       mov   esi, [DelFile_FNPointer]
3092 000099C8 B110                     <1>       mov   cl, 10h ; Directory attributes
3093 000099CA E8821D0000               <1>       call  make_sub_directory
3094                                    <1> loc_rename_file_ok: ; 06/03/2016
3095 000099CF 0F82DAFEFFFF             <1>         jc  loc_file_rw_cmd_failed
3096                                    <1> move_source_file_to_destination_OK:
3097 000099D5 BE[E3440100]             <1>       mov   esi, Msg_OK
3098 000099DA E876DBFFFF               <1>       call  print_msg
```

```
3099 000099DF E9CBFEFFFF          <1>       jmp    loc_file_rw_restore_retn
3100                              <1>
3101                              <1> loc_mkdir_y_n_escape:
3102 000099E4 B04E                <1>       mov    al, 'N' ; 'no'
3103 000099E6 EBCD                <1>       jmp    short loc_do_not_make_directory
3104                              <1>
3105                              <1> y_n_answer:
3106                              <1>       ; 29/12/2017
3107 000099E8 A2[DF440100]        <1>       mov    [Y_N_nextline], al
3108                              <1>       ;push  ax
3109 000099ED 50                  <1>       push   eax
3110 000099EE BE[DF440100]        <1>       mov    esi, Y_N_nextline
3111 000099F3 E85DDBFFFF          <1>       call   print_msg
3112 000099F8 58                  <1>       pop    eax
3113                              <1>       ;pop   ax
3114 000099F9 C3                  <1>       retn
3115                              <1>
3116                              <1> delete_directory:
3117                              <1>       ; 29/12/2017
3118                              <1>       ; 15/10/2016
3119                              <1>       ; 01/03/2016, 06/03/2016
3120                              <1>       ; 27/02/2016, 28/02/2016, 29/02/2016
3121                              <1>       ; 26/02/2016 (TRDOS 386 = TRDOS v2.0)
3122                              <1>       ; 16/10/2010 (CMD_INTR.ASM, 'cmp_cmd_rmdir')
3123                              <1>       ; 05/06/2010
3124                              <1>       ;
3125                              <1> get_fchar:
3126                              <1>       ; esi = directory name
3127 000099FA 803E20              <1>       cmp    byte [esi], 20h
3128 000099FD 7701                <1>         ja   short loc_rmdir_parse_path_name
3129                              <1>
3130                              <1> loc_rmdir_nodirname_retn:
3131 000099FF C3                  <1>       retn
3132                              <1>
3133                              <1> loc_rmdir_parse_path_name:
3134 00009A00 BF[26930100]        <1>       mov    edi, FindFile_Drv
3135 00009A05 E8481C0000          <1>       call   parse_path_name
3136 00009A0A 0F8261F5FFFF        <1>       jc     loc_cmd_failed
3137                              <1>
3138                              <1> loc_rmdir_check_dirname_exists:
3139 00009A10 BE[68930100]        <1>       mov    esi, FindFile_Name
3140 00009A15 803E20              <1>       cmp    byte [esi], 20h
3141 00009A18 0F8653F5FFFF        <1>       jna    loc_cmd_failed
3142 00009A1E 8935[E4930100]      <1>       mov    [DelFile_FNPointer], esi
3143                              <1>
3144                              <1> loc_rmdir_drv:
3145 00009A24 8A35[868A0100]      <1>       mov    dh, [Current_Drv]
3146 00009A2A 8835[E2910100]      <1>       mov    [RUN_CDRV], dh
3147                              <1>
3148 00009A30 8A15[26930100]      <1>       mov    dl, [FindFile_Drv]
3149 00009A36 38F2                <1>       cmp    dl, dh
3150 00009A38 740B                <1>       je     short loc_rmdir_change_directory
3151                              <1>
3152 00009A3A E89FE6FFFF          <1>       call   change_current_drive
3153 00009A3F 0F826AFEFFFF        <1>       jc     loc_file_rw_cmd_failed
3154                              <1>
3155                              <1> loc_rmdir_change_directory:
3156 00009A45 803D[27930100]20    <1>       cmp    byte [FindFile_Directory], 20h
3157 00009A4C 7614                <1>       jna    short loc_rmdir_find_directory
3158                              <1>
3159 00009A4E FE05[51400100]      <1>       inc    byte [Restore_CDIR]
3160 00009A54 BE[27930100]        <1>       mov    esi, FindFile_Directory
3161 00009A59 30E4                <1>       xor    ah, ah ; CD_COMMAND sign -> 0
3162 00009A5B E8DC150000          <1>       call   change_current_directory
3163 00009A60 7211                <1>       jc     short loc_rmdir_check_error_code
3164                              <1>
3165                              <1> ;loc_rmdir_change_prompt_dir_string:
3166                              <1>       ;call  change_prompt_dir_string
3167                              <1>
3168                              <1> loc_rmdir_find_directory:
3169                              <1>       ;mov   esi, FindFile_Name
3170 00009A62 8B35[E4930100]      <1>       mov    esi, [DelFile_FNPointer]
3171 00009A68 66B81008            <1>       mov    ax, 0810h ; Only directories
3172 00009A6C E810FAFFFF          <1>       call   find_first_file
3173 00009A71 730A                <1>       jnc    short loc_rmdir_ambgfn_check
3174                              <1>
3175                              <1> loc_rmdir_check_error_code:
3176 00009A73 3C02                <1>       cmp    al, 2
3177 00009A75 740B                <1>       je     short loc_rmdir_directory_not_found
3178 00009A77 F9                  <1>       stc
3179 00009A78 E932FEFFFF          <1>       jmp    loc_file_rw_cmd_failed
3180                              <1>
3181                              <1> loc_rmdir_ambgfn_check:
3182 00009A7D 6621D2              <1>       and    dx, dx ; Ambiguous filename chars used sign (DX>0)
3183 00009A80 740F                <1>       jz     short loc_rmdir_directory_found
3184                              <1>
3185                              <1> loc_rmdir_directory_not_found:
3186 00009A82 BE[53430100]        <1>       mov    esi, Msg_Dir_Not_Found
3187 00009A87 E8C9DAFFFF          <1>       call   print_msg
3188                              <1>
3189 00009A8C E91EFEFFFF          <1>       jmp    loc_file_rw_restore_retn
3190                              <1>
3191                              <1> loc_rmdir_directory_found:
3192 00009A91 80E307              <1>       and    bl, 07h ; Attributes
3193 00009A94 0F851FFEFFFF        <1>       jnz    loc_permission_denied
3194                              <1>
3195                              <1> loc_rmdir_save_lnel: ; 28/02/2016
3196                              <1>        ;mov  bh, [LongName_EntryLength]
3197 00009A9A 883D[EE930100]      <1>       mov    [DelFile_LNEL], bh ; Long name entry length (if > 0)
3198                              <1>       ; edi = Directory Entry Offset (DirBuff)
3199                              <1>       ; esi = Directory Entry (FFF Structure)
3200                              <1>       ;mov   [DelFile_DirEntryAddr], edi ; not required
3201                              <1>       ;mov   ax, [edi+20] ; First Cluster High Word
3202                              <1>         ;shl eax, 16
3203                              <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:
00009AA0 57             <1>       push  edi ; * (29/02/2016)
                        <1>
00009AA1 BE[E9440100]   <1>       mov   esi, Msg_DoYouWantRmDir
00009AA6 E8AADAFFFF     <1>       call  print_msg
00009AAB 8B35[E4930100] <1>       mov   esi, [DelFile_FNPointer]
00009AB1 E89FDAFFFF     <1>       call  print_msg
00009AB6 BE[D5440100]   <1>       mov   esi, Msg_YesNo
00009ABB E895DAFFFF     <1>       call  print_msg
                        <1>
                        <1> loc_rmdir_ask_again:
00009AC0 30E4           <1>       xor   ah, ah
00009AC2 E81E74FFFF     <1>       call  int16h
00009AC7 3C1B           <1>       cmp   al, 1Bh
                        <1>       ;je   short loc_do_not_delete_directory
00009AC9 7433           <1>         je    loc_rmdir_y_n_escape ; 06/03/2016
00009ACB 24DF           <1>       and   al, 0DFh
00009ACD A2[DF440100]   <1>       mov   [Y_N_nextline], al
00009AD2 3C59           <1>       cmp   al, 'Y'
00009AD4 7404           <1>       je    short loc_rmdir_yes_delete_directory
00009AD6 3C4E           <1>       cmp   al, 'N'
00009AD8 75E6           <1>       jne   short loc_rmdir_ask_again
                        <1>
                        <1> loc_do_not_delete_directory:
                        <1> loc_rmdir_yes_delete_directory:
00009ADA E809FFFFFF     <1>       call  y_n_answer ; 29/12/2017
00009ADF 5F             <1>       pop   edi ; * (29/02/2016)
                        <1>       ;cmp  al, 'Y' ; 'yes'
                        <1>       ;cmc
                        <1>         ;jnc loc_file_rw_restore_retn
00009AE0 3C4E           <1>       cmp   al, 'N' ; 'no'
00009AE2 0F84C7FDFFFF   <1>         je  loc_file_rw_restore_retn
                        <1>
                        <1>       ; 29/12/2017
00009AE8 E869000000     <1>       call  delete_sub_directory
00009AED 7213           <1>       jc    short loc_rmdir_cmd_failed
                        <1>
                        <1> loc_rmdir_ok:
00009AEF BE[E3440100]   <1>       mov   esi, Msg_OK
00009AF4 E85CDAFFFF     <1>       call  print_msg
00009AF9 E9B1FDFFFF     <1>       jmp   loc_file_rw_restore_retn
                        <1>
                        <1> loc_rmdir_y_n_escape:
00009AFE B04E           <1>       mov   al, 'N' ; 'no'
00009B00 EBD8           <1>         jmp   loc_do_not_delete_directory
                        <1>
                        <1> loc_rmdir_cmd_failed:
                        <1>       ; 29/12/2017
00009B02 09C0           <1>       or    eax, eax ; EAX = 0 -> Directory not empty!
00009B04 7426           <1>       jz    short loc_rmdir_directory_not_empty
                        <1>
                        <1>       ; EAX > 0 -> Error code in AL (or AX or EAX)
                        <1>
00009B06 833D[A2910100]01 <1>     cmp   dword [FAT_ClusterCounter], 1
00009B0D 0F829CFDFFFF   <1>       jb    loc_file_rw_cmd_failed
00009B13 F9             <1>       stc
                        <1> loc_rmdir_cmd_return:
                        <1>       ; 01/03/2016
00009B14 9C             <1>       pushf
                        <1>       ; ESI = Logical DOS Drive Description Table address
00009B15 66BB00FF       <1>       mov   bx, 0FF00h ; BH = FFh -> use ESI for Drive parameters
                        <1>                  ; BL = 0 -> Recalculate free cluster count
00009B19 50             <1>       push  eax
00009B1A E8C3380000     <1>       call  calculate_fat_freespace
00009B1F 58             <1>       pop   eax
00009B20 9D             <1>       popf
00009B21 0F8288FDFFFF   <1>       jc    loc_file_rw_cmd_failed
00009B27 E983FDFFFF     <1>       jmp   loc_file_rw_restore_retn
                        <1>
                        <1> loc_rmdir_directory_not_empty:
00009B2C BE[0A450100]   <1>       mov   esi, Msg_Dir_Not_Empty
00009B31 E81FDAFFFF     <1>       call  print_msg
                        <1>       ; 01/03/2016
00009B36 A1[A2910100]   <1>       mov   eax, [FAT_ClusterCounter]
00009B3B 09C0           <1>       or    eax, eax ; 0 ?
00009B3D 0F846CFDFFFF   <1>       jz    loc_file_rw_restore_retn
                        <1>       ; ESI = Logical DOS Drive Description Table address
00009B43 66BB01FF       <1>       mov   bx, 0FF01h ; BH = FFh -> use ESI for Drive parameters
                        <1>                  ; BL = 1 -> add free clusters
00009B47 E896380000     <1>       call  calculate_fat_freespace
00009B4C 09C9           <1>       or    ecx, ecx
00009B4E 0F845BFDFFFF   <1>         jz    loc_file_rw_restore_retn ; ecx = 0 -> OK
                        <1>       ; ecx > 0 -> Error (Recalculation is needed)
00009B54 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:
00009B56 668B4714       <1>       mov   ax, [edi+20] ; First Cluster High Word
00009B5A C1E010         <1>         shl  eax, 16
00009B5D 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
```

```
3309                                    <1>       ;mov   [DelFile_EntryCounter], bx
3310                                    <1>
3311 00009B61 29DB                     <1>       sub    ebx, ebx
3312                                    <1>       ; 29/12/2017
3313 00009B63 891D[A2910100]            <1>       mov    [FAT_ClusterCounter], ebx ; 0 ; Reset
3314                                    <1>
3315 00009B69 8A3D[26930100]            <1>       mov    bh, [FindFile_Drv]
3316 00009B6F BE00010900                <1>       mov    esi, Logical_DOSDisks
3317 00009B74 01DE                      <1>       add    esi, ebx
3318                                    <1>
3319 00009B76 66817F0CA101              <1>       cmp    word [edi+DirEntry_NTRes], 01A1h
3320 00009B7C 745A                      <1>       je     short loc_rmdir_check_fs_directory
3321                                    <1>
3322                                    <1>       ;cmp   byte [esi+LD_FATType], 1
3323                                    <1>       ;jb    short loc_rmdir_get__last_cluster_0
3324                                    <1>
3325                                    <1>       ; 29/12/2017
3326 00009B7E 83F802                    <1>       cmp    eax, 2
3327 00009B81 7306                      <1>       jnb    short loc_rmdir_get_last_cluster_1
3328                                    <1>       ; eax < 2
3329                                    <1> loc_rmdir_get_last_cluster_0:
3330                                    <1>       ;mov   eax, ERR_INV_FORMAT ; invalid format!
3331 00009B83 B813000000                <1>       mov    eax, ERR_NOT_DIR ; not a valid directory!
3332                                    <1>       ;stc
3333 00009B88 C3                        <1>       retn
3334                                    <1>
3335                                    <1> loc_rmdir_get_last_cluster_1:
3336 00009B89 807E0303                  <1>       cmp    byte [esi+LD_FATType], 3 ; FAT32
3337 00009B8D 750C                      <1>       jne    short loc_rmdir_get_last_cluster_2
3338                                    <1>
3339                                    <1>       ; is it root directory ?
3340 00009B8F 3B4632                    <1>       cmp    eax, [esi+LD_BPB+BPB_RootClus]
3341 00009B92 7507                      <1>       jne    short loc_rmdir_get_last_cluster_2
3342                                    <1>
3343                                    <1>       ; root directory can not be deleted !!
3344                                    <1> loc_rmdir_permission_denied:
3345 00009B94 B80B000000                <1>       mov    eax, ERR_PERM_DENIED ; permission denied!
3346 00009B99 F9                        <1>       stc
3347 00009B9A C3                        <1>       retn
3348                                    <1>
3349                                    <1> loc_rmdir_get_last_cluster_2:
3350                                    <1>       ; 29/12/2017
3351 00009B9B A3[E8930100]              <1>       mov    [DelFile_FCluster], eax
3352                                    <1>
3353                                    <1>       ;mov   dx, [DirBuff_EntryCounter]
3354 00009BA0 668B15[A0930100]          <1>       mov    dx, [FindFile_DirEntryNumber] ; 27/02/2016
3355 00009BA7 668915[EC930100]          <1>       mov    [DelFile_EntryCounter], dx
3356                                    <1>
3357 00009BAE 8B15[B1910100]            <1>       mov    edx, [DirBuff_Cluster]
3358 00009BB4 8915[18940100]            <1>       mov    [RmDir_ParentDirCluster], edx
3359                                    <1>
3360 00009BBA 893D[14940100]            <1>       mov    [RmDir_DirEntryOffset], edi
3361                                    <1>
3362                                    <1>       ; 01/03/2016
3363                                    <1>       ;mov   dword [FAT_ClusterCounter], 0 ; Reset
3364                                    <1>
3365                                    <1> loc_rmdir_get_last_cluster_3:
3366 00009BC0 E89C390000                <1>       call   get_last_cluster
3367                                    <1>         ;jc  loc_rmdir_cmd_failed
3368 00009BC5 721E                      <1>       jc     short loc_delete_sub_dir_retn ; 29/12/2017
3369                                    <1>
3370 00009BC7 3B05[E8930100]            <1>       cmp    eax, [DelFile_FCluster]
3371 00009BCD 7517                      <1>       jne    short loc_rmdir_multi_dir_clusters
3372                                    <1>
3373 00009BCF C605[13940100]00          <1>       mov    byte [RmDir_MultiClusters], 0
3374 00009BD6 EB15                      <1>       jmp    short pass_rmdir_multi_dir_clusters
3375                                    <1>
3376                                    <1> loc_rmdir_check_fs_directory:
3377                                    <1>       ; 29/12/2017
3378 00009BD8 807E04A1                  <1>       cmp    byte [esi+LD_FSType], 0A1h
3379 00009BDC 75B6                      <1>       jne    short loc_rmdir_permission_denied
3380                                    <1>
3381                                    <1> loc_rmdir_delete_fs_directory:
3382 00009BDE E876130000                <1>       call   delete_fs_directory
3383                                    <1>       ;jnc   loc_print_deleted_message
3384 00009BE3 7300                      <1>       jnc    short loc_delete_sub_dir_retn ; 29/12/2017
3385                                    <1>
3386                                    <1>       ; EAX=0 -> Directory not empty !
3387                                    <1>       ; EAX>0 -> Disk r/w error or another (misc) error
3388                                    <1>
3389                                    <1>       ;or    eax, eax
3390                                    <1>       ;jz    loc_rmdir_directory_not_empty_2
3391                                    <1>       ;;stc
3392                                    <1>       ;;jmp  loc_file_rw_cmd_failed
3393                                    <1>
3394                                    <1> loc_delete_sub_dir_retn:
3395 00009BE5 C3                        <1>       retn
3396                                    <1>
3397                                    <1> loc_rmdir_multi_dir_clusters:
3398 00009BE6 C605[13940100]01          <1>       mov    byte [RmDir_MultiClusters], 1
3399                                    <1>
3400                                    <1> pass_rmdir_multi_dir_clusters:
3401 00009BED A3[1C940100]              <1>       mov    [RmDir_DirLastCluster], eax
3402 00009BF2 890D[20940100]            <1>       mov    [RmDir_PreviousCluster], ecx
3403                                    <1>
3404                                    <1> loc_rmdir_load_fat_sub_directory:
3405 00009BF8 E84F330000                <1>       call   load_FAT_sub_directory
3406                                    <1>       ;jc    loc_rmdir_cmd_failed
3407 00009BFD 72E6                      <1>       jc     short loc_delete_sub_dir_retn
3408                                    <1>
3409                                    <1> loc_rmdir_find_last_dir_entry:
3410 00009BFF 56                        <1>       push   esi
3411 00009C00 BE[0A930100]              <1>       mov    esi, Dir_File_Name
3412 00009C05 C6062A                    <1>       mov    byte [esi], '*'
3413 00009C08 C646082A                  <1>       mov    byte [esi+8], '*'
```

```
3414 00009C0C 31DB              <1>       xor    ebx, ebx ; Entry offset  = 0
3415                            <1> loc_rmdir_find_last_dir_entry_next:
3416 00009C0E 66B80008          <1>       mov    ax, 0800h ; Except volume/long names
3417 00009C12 6631C9            <1>       xor    cx, cx ; 0 = Find a valid file or dir name
3418 00009C15 E87A170000        <1>       call   find_directory_entry
3419 00009C1A 7225              <1>       jc     short loc_rmdir_empty_dir_cluster
3420 00009C1C 83FB01            <1>       cmp    ebx, 1
3421 00009C1F 771B              <1>       ja     short loc_rmdir_directory_not_empty_1
3422                            <1> loc_rmdir_dot_entry_check:
3423 00009C21 80FD2E            <1>       cmp    ch, '.' ; The first char of the dir entry
3424 00009C24 7516              <1>       jne    short loc_rmdir_directory_not_empty_1
3425 00009C26 08DB              <1>       or     bl, bl
3426 00009C28 7506              <1>       jnz    short loc_rmdir_dotdot_entry_check
3427 00009C2A 807F0120          <1>       cmp    byte [edi+1], 20h
3428 00009C2E EB06              <1>       jmp    short pass_rmdir_dot_entry_check
3429                            <1>
3430                            <1> loc_rmdir_dotdot_entry_check:
3431 00009C30 66817F012E20      <1>       cmp    word [edi+1], '. '
3432                            <1> pass_rmdir_dot_entry_check:
3433 00009C36 7504              <1>       jne    short loc_rmdir_directory_not_empty_1
3434 00009C38 FEC3              <1>       inc    bl
3435 00009C3A EBD2              <1>       jmp    short loc_rmdir_find_last_dir_entry_next
3436                            <1>
3437                            <1> loc_rmdir_directory_not_empty_1:
3438 00009C3C 58                <1>       pop    eax ; pushed esi
3439 00009C3D 31C0              <1>       xor    eax, eax ; 0
3440                            <1> loc_rmdir_directory_not_empty_2:
3441                            <1> loc_delete_sub_dir_stc_retn:
3442 00009C3F F9                <1>       stc
3443 00009C40 C3                <1>       retn
3444                            <1>
3445                            <1> loc_rmdir_empty_dir_cluster:
3446 00009C41 5E                <1>       pop    esi
3447                            <1>
3448                            <1> loc_rmdir_set_prev_cluster_dir_last_cluster:
3449 00009C42 803D[13940100]00  <1>       cmp    byte [RmDir_MultiClusters], 0
3450 00009C49 7613              <1>       jna    short loc_rmdir_unlink_dir_last_cluster
3451                            <1>
3452 00009C4B A1[20940100]      <1>       mov    eax, [RmDir_PreviousCluster]
3453                            <1>       ;xor   ecx, ecx
3454 00009C50 49                <1>       dec    ecx ; FFFFFFFFh
3455 00009C51 E83A340000        <1>       call   update_cluster
3456 00009C56 7306              <1>       jnc    short loc_rmdir_unlink_dir_last_cluster
3457                            <1>
3458                            <1>       ; 01/03/2016
3459                            <1>       ;cmp   eax, 1  ; eax = 0 -> end of cluster chain
3460                            <1>       ;cmc
3461                            <1>       ;jc    short loc_rmdir_cmd_failed
3462                            <1>       ;jmp   short loc_rmdir_save_fat_buffer
3463                            <1>       ; 29/12/2017
3464 00009C58 21C0              <1>       and    eax, eax
3465 00009C5A 75E3              <1>       jnz    short loc_delete_sub_dir_stc_retn
3466 00009C5C EB12              <1>       jmp    short loc_rmdir_save_fat_buffer
3467                            <1>
3468                            <1> loc_rmdir_unlink_dir_last_cluster:
3469 00009C5E A1[1C940100]      <1>       mov    eax, [RmDir_DirLastCluster]
3470 00009C63 31C9              <1>       xor    ecx, ecx ; 0
3471 00009C65 E826340000        <1>       call   update_cluster
3472 00009C6A 7327              <1>       jnc    short loc_rmdir_unlink_stc_retn_0Bh
3473                            <1>       ; Because of it is the last cluster
3474                            <1>       ; 'update_cluster' must return with eocc error
3475 00009C6C 09C0              <1>       or     eax, eax
3476                            <1>       ;jz    short loc_rmdir_save_fat_buffer ; eocc
3477                            <1>       ;stc
3478                            <1>         ;jmp     short loc_rmdir_cmd_failed
3479                            <1>       ; 29/12/2017
3480 00009C6E 75CF              <1>       jnz    short loc_delete_sub_dir_stc_retn
3481                            <1>
3482                            <1> loc_rmdir_save_fat_buffer:
3483 00009C70 803D[9A910100]02  <1>       cmp    byte [FAT_BuffValidData], 2
3484 00009C77 7528              <1>       jne    short loc_rmdir_calculate_FAT_freespace
3485 00009C79 E8CF360000        <1>       call   save_fat_buffer
3486                            <1>       ;jc    short loc_rmdir_cmd_failed
3487                            <1>       ; 29/12/2017
3488 00009C7E 7219              <1>       jc     short loc_rmdir_unlink_error_retn
3489                            <1>
3490                            <1>       ; 01/03/2016
3491 00009C80 803D[13940100]00  <1>       cmp    byte [RmDir_MultiClusters], 0
3492 00009C87 7618              <1>       jna    short loc_rmdir_calculate_FAT_freespace
3493                            <1>
3494 00009C89 A1[E8930100]      <1>       mov    eax, [DelFile_FCluster]
3495 00009C8E E92DFFFFFF        <1>         jmp     loc_rmdir_get_last_cluster_3
3496                            <1>
3497                            <1> loc_rmdir_unlink_stc_retn_0Bh:
3498                            <1>       ; 15/10/2016 (0Bh -> 28)
3499 00009C93 B81C000000        <1>       mov    eax, ERR_INV_FORMAT ; 28 = Invalid format
3500                            <1> loc_rmdir_unlink_stc_retn:
3501 00009C98 F9                <1>       stc
3502                            <1> loc_rmdir_unlink_error_retn:
3503 00009C99 C3                <1>       retn
3504                            <1>
3505                            <1> loc_rmdir_delete_short_name_invalid_data:
3506 00009C9A B81D000000        <1>       mov    eax, 29 ; Invalid data (15/10/2016)
3507                            <1>       ;stc
3508                            <1>         ;jmp loc_rmdir_cmd_failed
3509                            <1>       ; 29/12/2017
3510 00009C9F EBF7              <1>       jmp    short loc_rmdir_unlink_stc_retn
3511                            <1>
3512                            <1> loc_rmdir_calculate_FAT_freespace:
3513                            <1>       ;mov   eax, [FAT_ClusterCounter]
3514                            <1>       ; 29/12/2017
3515 00009CA1 29C0              <1>       sub    eax, eax ; 0
3516 00009CA3 8705[A2910100]    <1>       xchg   eax, [FAT_ClusterCounter]
3517                            <1>       ;
3518 00009CA9 66BB01FF          <1>       mov    bx, 0FF01h
```

```
3519                                    <1>       ; BL = 1 -> Add EAX to free space count
3520                                    <1>       ; BH = FFh ->
3521                                    <1>       ; ESI = Logical DOS Drive Description Table address
3522 00009CAD E830370000                <1>       call   calculate_fat_freespace
3523                                    <1>
3524 00009CB2 21C9                      <1>       and    ecx, ecx ; ecx = 0 -> valid free sector count
3525 00009CB4 7409                      <1>       jz     short loc_rmdir_delete_short_name_continue
3526                                    <1>
3527                                    <1> loc_rmdir_recalculate_FAT_freespace:
3528 00009CB6 66BB00FF                  <1>         mov     bx, 0FF00h ; BL = 0 -> Recalculate free space
3529 00009CBA E823370000                <1>       call   calculate_fat_freespace
3530                                    <1>
3531                                    <1> loc_rmdir_delete_short_name_continue:
3532 00009CBF A1[18940100]              <1>       mov    eax, [RmDir_ParentDirCluster]
3533 00009CC4 83F802                    <1>       cmp    eax, 2
3534 00009CC7 7309                      <1>       jnb    short loc_rmdir_del_short_name_load_sub_dir
3535 00009CC9 E8F3310000                <1>       call   load_FAT_root_directory
3536                                    <1>       ;jc    loc_file_rw_cmd_failed
3537                                    <1>       ; 29/12/2017
3538 00009CCE 72C9                      <1>       jc     short loc_rmdir_unlink_error_retn
3539 00009CD0 EB07                      <1>       jmp    short loc_rmdir_del_short_name_ld_chk_fclust
3540                                    <1>
3541                                    <1> loc_rmdir_del_short_name_load_sub_dir:
3542 00009CD2 E875320000                <1>       call   load_FAT_sub_directory
3543                                    <1>       ;jc    loc_file_rw_cmd_failed
3544                                    <1>       ; 29/12/2017
3545 00009CD7 72C0                      <1>       jc     short loc_rmdir_unlink_error_retn
3546                                    <1>
3547                                    <1> loc_rmdir_del_short_name_ld_chk_fclust:
3548 00009CD9 0FB73D[14940100]          <1>       movzx  edi, word [RmDir_DirEntryOffset]
3549 00009CE0 81C700000800              <1>       add    edi, Directory_Buffer
3550                                    <1>
3551 00009CE6 668B4714                  <1>       mov    ax, [edi+20] ; First Cluster High Word
3552 00009CEA C1E010                    <1>       shl    eax, 16
3553 00009CED 668B471A                  <1>       mov    ax, [edi+26] ; First Cluster Low Word
3554                                    <1>         ; Not necessary...
3555 00009CF1 3B05[E8930100]            <1>       cmp    eax, [DelFile_FCluster]
3556 00009CF7 75A1                      <1>       jne    short loc_rmdir_delete_short_name_invalid_data
3557                                    <1>       ;
3558 00009CF9 C607E5                    <1>       mov    byte [edi], 0E5h ; 'Deleted' sign
3559                                    <1>       ; 27/02/2016
3560                                    <1>       ; TRDOS v1 has a bug here! it does not set
3561                                    <1>       ; 'DirBuff_ValidData' to 2; as result of this bug,
3562                                    <1>       ; 'save_directory_buffer' would not save the change !
3563 00009CFC C605[AC910100]02            <1>       mov    byte [DirBuff_ValidData], 2 ; change sign
3564                                    <1>       ;
3565 00009D03 E8AE1D0000                <1>       call   save_directory_buffer
3566                                    <1>       ;jc    loc_file_rw_cmd_failed
3567                                    <1>       ; 29/12/2017
3568 00009D08 728F                      <1>       jc     short loc_rmdir_unlink_error_retn
3569                                    <1>
3570                                    <1> loc_rmdir_del_long_name:
3571 00009D0A 0FB615[EE930100]          <1>       movzx  edx, byte [DelFile_LNEL]
3572 00009D11 08D2                      <1>       or     dl, dl
3573 00009D13 7410                      <1>       jz     short loc_rmdir_update_parent_dir_lmdt
3574                                    <1>
3575 00009D15 0FB705[EC930100]          <1>       movzx  eax, word [DelFile_EntryCounter]
3576 00009D1C 29D0                      <1>       sub    eax, edx
3577                                    <1>       ; 29/12/2017
3578 00009D1E 7205                      <1>       jc     short loc_rmdir_update_parent_dir_lmdt
3579                                    <1>
3580                                    <1>       ; EAX = Directory Entry Number of the long name last entry
3581 00009D20 E8EF1E0000                <1>       call   delete_longname
3582                                    <1>
3583                                    <1> loc_rmdir_update_parent_dir_lmdt:
3584 00009D25 E8271E0000                <1>       call   update_parent_dir_lmdt
3585                                    <1>       ;jc    short loc_file_rw_cmd_failed
3586                                    <1>       ; 29/12/2017
3587                                    <1>       ;jc    short loc_rmdir_unlink_error_retn
3588                                    <1>
3589                                    <1> loc_delete_sub_directory_ok:
3590                                    <1>       ; 29/12/2017
3591 00009D2A 31C0                      <1>       xor    eax, eax ;  0 ;  cf = 0
3592 00009D2C C3                        <1>       retn
3593                                    <1>
3594                                    <1>
3595                                    <1> delete_file:
3596                                    <1>       ; 29/02/2016
3597                                    <1>       ; 28/02/2016 (TRDOS 386 = TRDOS v2.0)
3598                                    <1>       ; 09/08/2010 (CMD_INTR.ASM, 'cmp_cmd_del')
3599                                    <1>       ; 28/02/2010
3600                                    <1>
3601                                    <1> get_delfile_fchar:
3602                                    <1>       ; esi = file name
3603 00009D2D 803E20                    <1>       cmp    byte [esi], 20h
3604 00009D30 7701                      <1>         ja   short loc_delfile_parse_path_name
3605                                    <1>
3606                                    <1> loc_delfile_nofilename_retn:
3607 00009D32 C3                        <1>       retn
3608                                    <1>
3609                                    <1> loc_delfile_parse_path_name:
3610 00009D33 BF[26930100]              <1>       mov    edi, FindFile_Drv
3611 00009D38 E815190000                <1>       call   parse_path_name
3612 00009D3D 0F822EF2FFFF              <1>       jc     loc_cmd_failed
3613                                    <1>
3614                                    <1> loc_delfile_check_filename_exists:
3615 00009D43 BE[68930100]              <1>       mov    esi, FindFile_Name
3616 00009D48 803E20                    <1>       cmp    byte [esi], 20h
3617 00009D4B 0F8620F2FFFF              <1>       jna    loc_cmd_failed
3618 00009D51 8935[E4930100]            <1>       mov    [DelFile_FNPointer], esi
3619                                    <1>
3620                                    <1> loc_delfile_drv:
3621 00009D57 8A15[26930100]            <1>       mov    dl, [FindFile_Drv]
3622 00009D5D 8A35[868A0100]            <1>       mov    dh, [Current_Drv]
3623 00009D63 8835[E2910100]            <1>       mov    [RUN_CDRV], dh
```

```
3624 00009D69 38F2                <1>       cmp    dl, dh
3625 00009D6B 740B                <1>       je     short loc_delfile_change_directory
3626                              <1>
3627 00009D6D E86CE3FFFF          <1>       call   change_current_drive
3628 00009D72 0F8237FBFFFF        <1>       jc     loc_file_rw_cmd_failed
3629                              <1>
3630                              <1> loc_delfile_change_directory:
3631 00009D78 803D[27930100]20    <1>       cmp    byte [FindFile_Directory], 20h
3632 00009D7F 7618                <1>       jna    short loc_delfile_find
3633                              <1>
3634 00009D81 FE05[51400100]      <1>       inc    byte [Restore_CDIR]
3635 00009D87 BE[27930100]        <1>       mov    esi, FindFile_Directory
3636 00009D8C 30E4                <1>       xor    ah, ah ; CD_COMMAND sign -> 0
3637 00009D8E E8A9120000          <1>       call   change_current_directory
3638 00009D93 0F8216FBFFFF        <1>       jc     loc_file_rw_cmd_failed
3639                              <1>
3640                              <1> ;loc_delfile_change_prompt_dir_string:
3641                              <1>       ;call  change_prompt_dir_string
3642                              <1>
3643                              <1> loc_delfile_find:
3644                              <1>       ;mov   esi, FindFile_Name
3645 00009D99 8B35[E4930100]      <1>       mov    esi, [DelFile_FNPointer]
3646 00009D9F 66B80018            <1>       mov    ax, 1800h ; Except volume label and dirs
3647 00009DA3 E8D9F6FFFF          <1>       call   find_first_file
3648 00009DA8 0F8201FBFFFF        <1>       jc     loc_file_rw_cmd_failed
3649                              <1>
3650                              <1> loc_delfile_ambgfn_check:
3651 00009DAE 6621D2              <1>       and    dx, dx ; Ambiguous filename chars used sign (DX>0)
3652 00009DB1 740B                <1>       jz     short loc_delfile_found
3653                              <1>
3654                              <1> loc_file_not_found:
3655 00009DB3 B802000000          <1>       mov    eax, 2 ; File not found sign
3656 00009DB8 F9                  <1>       stc
3657 00009DB9 E9F1FAFFFF          <1>       jmp    loc_file_rw_cmd_failed
3658                              <1>
3659                              <1> loc_delfile_found:
3660 00009DBE 80E307              <1>       and    bl, 07h ; Attributes
3661 00009DC1 0F85F2FAFFFF        <1>         jnz     loc_permission_denied
3662                              <1>
3663                              <1> ;loc_delfile_found_save_lnel:
3664                              <1> ;     mov    [DelFile_LNEL], bh ; Long name entry length (if > 0)
3665                              <1>
3666                              <1> loc_delfile_ask_for_delete:
3667 00009DC7 57                  <1>       push   edi ; * (29/02/2016)
3668                              <1>
3669 00009DC8 BE[21450100]        <1>       mov    esi, Msg_DoYouWantDelete
3670 00009DCD E883D7FFFF          <1>       call   print_msg
3671 00009DD2 8B35[E4930100]      <1>       mov    esi, [DelFile_FNPointer]
3672 00009DD8 E878D7FFFF          <1>       call   print_msg
3673 00009DDD BE[D5440100]        <1>       mov    esi, Msg_YesNo
3674 00009DE2 E86ED7FFFF          <1>       call   print_msg
3675                              <1>
3676                              <1> loc_delfile_ask_again:
3677 00009DE7 30E4                <1>       xor    ah, ah
3678 00009DE9 E8F770FFFF          <1>       call   int16h
3679 00009DEE 3C1B                <1>       cmp    al, 1Bh
3680                              <1>       ;je    short loc_do_not_delete_file
3681 00009DF0 7449                <1>       je     short loc_delfile_y_n_escape ; 06/03/2016
3682 00009DF2 24DF                <1>       and    al, 0DFh
3683 00009DF4 A2[DF440100]        <1>       mov    [Y_N_nextline], al
3684 00009DF9 3C59                <1>       cmp    al, 'Y'
3685 00009DFB 7404                <1>       je     short loc_yes_delete_file
3686 00009DFD 3C4E                <1>       cmp    al, 'N'
3687 00009DFF 75E6                <1>       jne    short loc_delfile_ask_again
3688                              <1>
3689                              <1> loc_do_not_delete_file:
3690                              <1> loc_yes_delete_file:
3691 00009E01 E8E2FBFFFF          <1>       call   y_n_answer ; 29/12/2017
3692 00009E06 5F                  <1>       pop    edi ; * (29/02/2016)
3693                              <1>       ;cmp   al, 'Y' ; 'yes'
3694                              <1>       ;cmc
3695                              <1>         ;jnc loc_file_rw_restore_retn
3696 00009E07 3C4E                <1>       cmp    al, 'N' ; 'no'
3697 00009E09 0F84A0FAFFFF        <1>         je   loc_file_rw_restore_retn
3698                              <1>
3699                              <1> loc_delete_file:
3700 00009E0F 8A3D[26930100]      <1>       mov    bh, [FindFile_Drv]
3701                              <1>       ;mov   bl, [DelFile_LNEL]
3702 00009E15 8A1D[75930100]      <1>       mov    bl, [FindFile_LongNameEntryLength]
3703                              <1>       ;mov   cx, [DirBuff_EntryCounter]
3704 00009E1B 668B0D[A0930100]    <1>       mov    cx, [FindFile_DirEntryNumber]
3705                              <1>       ; (*) EDI = Directory buffer entry offset/address
3706 00009E22 E8D71F0000          <1>       call   remove_file ; (FILE.ASM, 'proc_delete_file')
3707 00009E27 0F8378FAFFFF        <1>       jnc    loc_print_deleted_message
3708                              <1>
3709                              <1>       ;cmp   al, 05h
3710 00009E2D 3C0B                <1>       cmp    al, ERR_PERM_DENIED  ; 29/12/2017 (5 -> 11)
3711 00009E2F 0F8484FAFFFF        <1>       je     loc_permission_denied
3712 00009E35 F9                  <1>       stc
3713 00009E36 E974FAFFFF          <1>       jmp    loc_file_rw_cmd_failed
3714                              <1>
3715                              <1> loc_delfile_y_n_escape:
3716 00009E3B B04E                <1>       mov    al, 'N' ; 'no'
3717 00009E3D EBC2                <1>       jmp    short loc_do_not_delete_file
3718                              <1>
3719                              <1> set_file_attributes:
3720                              <1>       ; 06/03/2016
3721                              <1>       ; 04/03/2016 (TRDOS 386 = TRDOS v2.0)
3722                              <1>       ; 10/07/2010 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_attrib')
3723                              <1>       ; 23/05/2010
3724                              <1>       ; 17/12/2000 (P2000.ASM)
3725                              <1>
3726                              <1>       ; esi = file or directory name
3727 00009E3F 6631C0              <1>       xor    ax, ax
3728 00009E42 66A3[72450100]      <1>       mov    [Attr_Chars], ax
```

```
3729 00009E48 A2[3C940100]        <1>       mov    [Attributes], al
3730                              <1>
3731                              <1> get_attrib_fchar:
3732                              <1>       ; esi = file name
3733 00009E4D 8A06                <1>       mov    al, [esi]
3734 00009E4F 3C20                <1>       cmp    al, 20h
3735 00009E51 7623                <1>       jna    short loc_attr_file_nofilename_retn
3736                              <1>
3737                              <1> loc_scan_attrib_params:
3738 00009E53 3C2D                <1>       cmp    al, '-'
3739 00009E55 0F871C010000        <1>       ja     loc_attr_file_parse_path_name
3740 00009E5B 7408                <1>       je     short loc_attr_space
3741                              <1>
3742 00009E5D 3C2B                <1>       cmp    al, '+'
3743 00009E5F 0F850CF1FFFF        <1>       jne    loc_cmd_failed
3744                              <1>
3745                              <1> loc_attr_space:
3746 00009E65 8A6601              <1>       mov    ah, [esi+1]
3747 00009E68 80FC20              <1>       cmp    ah, 20h
3748 00009E6B 770A                <1>       ja     short pass_attr_space
3749 00009E6D 0F82FEF0FFFF        <1>       jb     loc_cmd_failed
3750 00009E73 46                  <1>       inc    esi
3751 00009E74 EBEF                <1>       jmp    short loc_attr_space
3752                              <1>
3753                              <1> loc_attr_file_nofilename_retn:
3754 00009E76 C3                  <1>       retn
3755                              <1>
3756                              <1> pass_attr_space:
3757 00009E77 80E4DF              <1>       and    ah, 0DFh
3758 00009E7A 80FC53              <1>       cmp    ah, 'S'
3759 00009E7D 0F87EEF0FFFF        <1>       ja     loc_cmd_failed
3760 00009E83 7204                <1>       jb     short pass_attr_system
3761 00009E85 B404                <1>       mov    ah, 04h    ; System
3762 00009E87 EB21                <1>       jmp    short pass_attr_archive
3763                              <1>
3764                              <1> pass_attr_system:
3765 00009E89 80FC48              <1>       cmp    ah, 'H'
3766 00009E8C 7706                <1>       ja     short pass_attr_hidden
3767 00009E8E 7213                <1>       jb     short pass_attr_read_only
3768 00009E90 B402                <1>       mov    ah, 02h    ; Hidden
3769 00009E92 EB16                <1>       jmp    short pass_attr_archive
3770                              <1>
3771                              <1> pass_attr_hidden:
3772 00009E94 80FC52              <1>       cmp    ah, 'R'
3773 00009E97 0F87D4F0FFFF        <1>       ja     loc_cmd_failed
3774 00009E9D 7204                <1>       jb     short pass_attr_read_only ; Read only
3775 00009E9F B401                <1>       mov    ah, 01h
3776 00009EA1 EB07                <1>       jmp    short pass_attr_archive
3777                              <1>
3778                              <1> pass_attr_read_only:
3779 00009EA3 80FC41              <1>       cmp    ah, 'A'
3780 00009EA6 753B                <1>       jne    short loc_chk_attr_enter
3781 00009EA8 B420                <1>       mov    ah, 20h    ; Archive
3782                              <1>
3783                              <1> pass_attr_archive:
3784 00009EAA 3C2D                <1>       cmp    al, '-'
3785 00009EAC 7508                <1>       jne    short pass_reducing_attributes
3786 00009EAE 0825[72450100]      <1>       or     [Attr_Chars], ah
3787 00009EB4 EB06                <1>       jmp    short loc_change_attributes_inc
3788                              <1>
3789                              <1> pass_reducing_attributes:
3790 00009EB6 0825[73450100]      <1>       or     [Attr_Chars+1], ah
3791                              <1>
3792                              <1> loc_change_attributes_inc:
3793 00009EBC 46                  <1>       inc    esi
3794 00009EBD 8A6601              <1>       mov    ah, [esi+1]
3795 00009EC0 80FC20              <1>       cmp    ah, 20h
3796 00009EC3 7227                <1>       jb     short pass_change_attr
3797 00009EC5 74F5                <1>       je     short loc_change_attributes_inc
3798 00009EC7 80FC2D              <1>       cmp    ah, '-'
3799 00009ECA 770D                <1>       ja     short loc_chk_next_attr_char1
3800 00009ECC 7405                <1>       je     short loc_chk_next_attr_char0
3801 00009ECE 80FC2B              <1>       cmp    ah, '+'
3802 00009ED1 7506                <1>       jne    short loc_chk_next_attr_char1
3803                              <1>
3804                              <1> loc_chk_next_attr_char0:
3805 00009ED3 46                  <1>       inc    esi
3806 00009ED4 668B06              <1>       mov    ax, [esi]
3807 00009ED7 EB9E                <1>       jmp    short pass_attr_space
3808                              <1>
3809                              <1> loc_chk_next_attr_char1:
3810 00009ED9 803E2D              <1>       cmp    byte [esi], '-'
3811 00009EDC 7799                <1>       ja     short pass_attr_space
3812 00009EDE E988000000          <1>         jmp     loc_attr_file_check_fname_fchar
3813                              <1>
3814                              <1> loc_chk_attr_enter:
3815 00009EE3 80FC0D              <1>       cmp    ah, 0Dh
3816 00009EE6 0F8585F0FFFF        <1>       jne    loc_cmd_failed
3817                              <1>
3818                              <1> pass_change_attr:
3819 00009EEC A0[72450100]        <1>       mov    al, [Attr_Chars]
3820 00009EF1 F6D0                <1>       not    al
3821 00009EF3 2005[3C940100]      <1>       and    [Attributes], al
3822 00009EF9 A0[73450100]        <1>       mov    al, [Attr_Chars+1]
3823 00009EFE 0805[3C940100]      <1>       or     [Attributes], al
3824                              <1>
3825                              <1> loc_show_attributes:
3826 00009F04 BE[EF4C0100]        <1>       mov    esi, nextline
3827 00009F09 E847D6FFFF          <1>       call   print_msg
3828                              <1>
3829                              <1> loc_show_attributes_no_nextline:
3830 00009F0E C705[72450100]4E4F- <1>       mov    dword [Attr_Chars], 'NORM'
3830 00009F16 524D                <1>
3831 00009F18 66C705[76450100]41- <1>       mov    word [Attr_Chars+4], 'AL'
3831 00009F20 4C                  <1>
```

```
3832 00009F21 BE[72450100]        <1>       mov   esi, Attr_Chars
3833 00009F26 A0[3C940100]        <1>       mov   al, [Attributes]
3834 00009F2B A804                <1>       test  al, 04h
3835 00009F2D 7406                <1>       jz    short pass_put_attr_s
3836 00009F2F 66C7065300          <1>       mov   word [esi], 0053h     ; S
3837 00009F34 46                  <1>       inc   esi
3838                              <1>
3839                              <1> pass_put_attr_s:
3840 00009F35 A802                <1>       test  al, 02h
3841 00009F37 7406                <1>       jz    short pass_put_attr_h
3842 00009F39 66C7064800          <1>       mov   word [esi], 0048h     ; H
3843 00009F3E 46                  <1>       inc   esi
3844                              <1>
3845                              <1> pass_put_attr_h:
3846 00009F3F A801                <1>       test  al, 01h
3847 00009F41 7406                <1>       jz    short pass_put_attr_r
3848 00009F43 66C7065200          <1>       mov   word [esi], 0052h     ; R
3849 00009F48 46                  <1>       inc   esi
3850                              <1>
3851                              <1> pass_put_attr_r:
3852 00009F49 3C20                <1>       cmp   al, 20h
3853 00009F4B 7205                <1>       jb    short pass_put_attr_a
3854 00009F4D 66C7064100          <1>       mov   word [esi], 0041h     ; A
3855                              <1>
3856                              <1> pass_put_attr_a:
3857 00009F52 BE[65450100]        <1>       mov   esi, Str_Attributes
3858 00009F57 E8F9D5FFFF          <1>       call  print_msg
3859 00009F5C BE[EF4C0100]        <1>       mov   esi, nextline
3860 00009F61 E8EFD5FFFF          <1>       call  print_msg
3861 00009F66 E944F9FFFF          <1>       jmp   loc_file_rw_restore_retn
3862                              <1>
3863                              <1> loc_attr_file_check_fname_fchar:
3864 00009F6B 46                  <1>       inc   esi
3865 00009F6C 803E20              <1>       cmp   byte [esi], 20h
3866 00009F6F 74FA                <1>       je    short loc_attr_file_check_fname_fchar
3867 00009F71 0F8275FFFFFF        <1>         jb      pass_change_attr
3868                              <1>
3869                              <1> loc_attr_file_parse_path_name:
3870 00009F77 BF[26930100]        <1>       mov   edi, FindFile_Drv
3871 00009F7C E8D1160000          <1>       call  parse_path_name
3872 00009F81 0F82EAEFFFFF        <1>       jc    loc_cmd_failed
3873                              <1>
3874                              <1> loc_attr_file_check_filename_exists:
3875 00009F87 BE[68930100]        <1>       mov   esi, FindFile_Name
3876 00009F8C 803E20              <1>       cmp   byte [esi], 20h
3877 00009F8F 0F86DCEFFFFF        <1>       jna   loc_cmd_failed
3878 00009F95 8935[E4930100]      <1>       mov   [DelFile_FNPointer], esi
3879                              <1>
3880                              <1> loc_attr_file_drv:
3881 00009F9B 8A35[868A0100]      <1>       mov   dh, [Current_Drv]
3882 00009FA1 8835[E2910100]      <1>       mov   [RUN_CDRV], dh
3883                              <1>
3884 00009FA7 8A15[26930100]      <1>       mov   dl, [FindFile_Drv]
3885 00009FAD 38F2                <1>       cmp   dl, dh
3886 00009FAF 740B                <1>       je    short loc_attr_file_change_directory
3887                              <1>
3888 00009FB1 E828E1FFFF          <1>       call  change_current_drive
3889 00009FB6 0F82F3F8FFFF        <1>       jc    loc_file_rw_cmd_failed
3890                              <1>
3891                              <1> loc_attr_file_change_directory:
3892 00009FBC 803D[27930100]20    <1>         cmp     byte [FindFile_Directory], 20h
3893 00009FC3 7618                <1>       jna   short loc_attr_file_find
3894                              <1>
3895 00009FC5 FE05[51400100]      <1>       inc   byte [Restore_CDIR]
3896                              <1>
3897 00009FCB BE[27930100]        <1>       mov   esi, FindFile_Directory
3898 00009FD0 30E4                <1>       xor   ah, ah ; CD_COMMAND sign -> 0
3899 00009FD2 E865100000          <1>       call  change_current_directory
3900 00009FD7 0F82D2F8FFFF        <1>       jc    loc_file_rw_cmd_failed
3901                              <1>
3902                              <1> ;loc_attr_file_change_prompt_dir_string:
3903                              <1>       ;call  change_prompt_dir_string
3904                              <1>
3905                              <1> loc_attr_file_find:
3906                              <1>       ;mov   esi, FindFile_Name
3907 00009FDD 8B35[E4930100]      <1>       mov   esi, [DelFile_FNPointer]
3908 00009FE3 66B80008            <1>       mov   ax, 0800h ; Except volume labels
3909 00009FE7 E895F4FFFF          <1>       call  find_first_file
3910 00009FEC 0F82BDF8FFFF        <1>       jc    loc_file_rw_cmd_failed
3911                              <1>
3912                              <1> loc_attr_file_ambgfn_check:
3913 00009FF2 6609D2              <1>       or    dx, dx ; Ambiguous filename chars used sign (DX>0)
3914                              <1>       ;     (Note: It was BX in TRDOS v1)
3915                              <1>       ;jz   short loc_attr_file_found
3916 00009FF5 0F85B8FDFFFF        <1>         jnz     loc_file_not_found ; 06/03/2016
3917                              <1>
3918                              <1>       ;mov  eax, 2 ; File not found sign
3919                              <1>       ;stc
3920                              <1>       ;jmp  loc_file_rw_cmd_failed
3921                              <1>
3922                              <1> loc_attr_file_found:
3923                              <1>       ; EDI = Directory buffer entry offset/address
3924                              <1>       ; BL = File (or Directory) Attributes
3925                              <1>       ;     (Note: It was 'CL' in TRDOS v1)
3926                              <1>       ; mov  bl, [EDI+0Bh]
3927                              <1>
3928 00009FFB 66833D[72450100]00  <1>       cmp   word [Attr_Chars], 0
3929 0000A003 770B                <1>       ja    short loc_attr_file_change_attributes
3930 0000A005 881D[3C940100]      <1>       mov   [Attributes], bl
3931 0000A00B E9F4FEFFFF          <1>       jmp   loc_show_attributes
3932                              <1>
3933                              <1> loc_attr_file_change_attributes:
3934 0000A010 A0[72450100]        <1>       mov   al, [Attr_Chars]
3935 0000A015 F6D0                <1>       not   al
3936 0000A017 20C3                <1>       and   bl, al
```

```
3937 0000A019 A0[73450100]        <1>       mov    al, [Attr_Chars+1]
3938 0000A01E 08C3                <1>       or     bl, al
3939                              <1>
3940 0000A020 66817F0CA101        <1>       cmp    word [edi+DirEntry_NTRes], 01A1h ; Singlix FS
3941 0000A026 741D                <1>       je     short loc_attr_file_fs_check
3942                              <1>
3943 0000A028 881D[3C940100]      <1>       mov    [Attributes], bl
3944 0000A02E 885F0B              <1>       mov    [edi+0Bh], bl    ; Attributes (New!)
3945                              <1>
3946                              <1>       ; 04/03/2016
3947                              <1>       ; TRDOS v1 has a bug here! it does not set
3948                              <1>       ; 'DirBuff_ValidData' to 2; as result of this bug,
3949                              <1>       ; 'save_directory_buffer' would not save the new attributes !
3950                              <1>
3951 0000A031 C605[AC910100]02    <1>       mov    byte [DirBuff_ValidData], 2
3952                              <1>
3953 0000A038 E8791A0000          <1>       call   save_directory_buffer
3954 0000A03D 0F826CF8FFFF        <1>       jc     loc_file_rw_cmd_failed
3955                              <1>
3956 0000A043 EB33                <1>       jmp    short loc_print_attr_changed_message
3957                              <1>
3958                              <1> loc_attr_file_fs_check:
3959 0000A045 29C0                <1>       sub    eax, eax
3960 0000A047 8A25[AA910100]      <1>         mov     ah, [DirBuff_DRV]
3961 0000A04D BE00010900          <1>       mov    esi, Logical_DOSDisks
3962 0000A052 01C6                <1>         add     esi, eax
3963 0000A054 807E04A1            <1>         cmp     byte [esi+LD_FSType], 0A1h
3964 0000A058 7309                <1>       jnc    short loc_attr_file_change_fs_file_attributes
3965                              <1>       ; 29/12/2017 (0Dh -> 29)
3966 0000A05A 66B81D00            <1>       mov    ax, 29 ; Invalid Data
3967 0000A05E E94CF8FFFF          <1>       jmp    loc_file_rw_cmd_failed
3968                              <1>
3969                              <1> loc_attr_file_change_fs_file_attributes:
3970                              <1>       ; BL = New MS-DOS File Attributes
3971 0000A063 88D8                <1>       mov    al, bl ; File/Directory Attributes
3972 0000A065 30E4                <1>       xor    ah, ah ; Attributes in MS-DOS format sign
3973 0000A067 E873050000          <1>       call   change_fs_file_attributes
3974 0000A06C 0F823DF8FFFF        <1>       jc     loc_file_rw_cmd_failed
3975                              <1>
3976 0000A072 881D[3C940100]      <1>       mov    [Attributes], bl
3977                              <1>
3978                              <1> loc_print_attr_changed_message:
3979 0000A078 BE[60450100]        <1>       mov    esi, Msg_New
3980 0000A07D E8D3D4FFFF          <1>       call   print_msg
3981 0000A082 E987FEFFFF          <1>       jmp    loc_show_attributes_no_nextline
3982                              <1>
3983                              <1> rename_file:
3984                              <1>       ; 13/11/2017
3985                              <1>       ; 06/11/2016
3986                              <1>       ; 05/11/2016
3987                              <1>       ; 16/10/2016
3988                              <1>       ; 08/03/2016
3989                              <1>       ; 06/03/2016 (TRDOS 386 = TRDOS v2.0)
3990                              <1>       ; 20/11/2010 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_rename')
3991                              <1>       ; 16/11/2010
3992                              <1>
3993                              <1> get_rename_source_fchar:
3994                              <1>       ; esi = file name
3995 0000A087 803E20              <1>       cmp    byte [esi], 20h
3996 0000A08A 7614                <1>         jna  short loc_rename_nofilename_retn
3997                              <1>
3998 0000A08C 8935[64940100]      <1>       mov    [SourceFilePath], esi
3999                              <1>
4000                              <1> rename_scan_source_file:
4001 0000A092 46                  <1>       inc    esi
4002 0000A093 803E20              <1>       cmp    byte [esi], 20h
4003 0000A096 7409                <1>       je     short rename_scan_destination_file_1
4004                              <1>       ;jb    short loc_rename_nofilename_retn
4005 0000A098 0F82D3EEFFFF        <1>       jb     loc_cmd_failed
4006 0000A09E EBF2                <1>       jmp    short rename_scan_source_file
4007                              <1>
4008                              <1> loc_rename_nofilename_retn: ; 08/03/2016
4009 0000A0A0 C3                  <1>       retn
4010                              <1>
4011                              <1> rename_scan_destination_file_1:
4012 0000A0A1 C60600              <1>       mov    byte [esi], 0
4013                              <1>
4014                              <1> rename_scan_destination_file_2:
4015 0000A0A4 46                  <1>       inc    esi
4016 0000A0A5 803E20              <1>       cmp    byte [esi], 20h
4017 0000A0A8 74FA                <1>       je     short rename_scan_destination_file_2
4018                              <1>       ;jb    short loc_rename_nofilename_retn
4019 0000A0AA 0F82C1EEFFFF        <1>       jb     loc_cmd_failed
4020                              <1>
4021 0000A0B0 8935[68940100]      <1>       mov    [DestinationFilePath], esi
4022                              <1>
4023                              <1> rename_scan_destination_file_3:
4024 0000A0B6 46                  <1>       inc    esi
4025 0000A0B7 803E20              <1>       cmp    byte [esi], 20h
4026 0000A0BA 77FA                <1>       ja     short rename_scan_destination_file_3
4027                              <1>
4028 0000A0BC C60600              <1>       mov    byte [esi], 0
4029                              <1>
4030                              <1> loc_rename_save_current_drive:
4031 0000A0BF 8A35[868A0100]      <1>       mov    dh, [Current_Drv]
4032 0000A0C5 8835[E2910100]      <1>       mov    byte [RUN_CDRV], dh
4033                              <1>
4034                              <1> loc_rename_sf_parse_path_name:
4035 0000A0CB 8B35[64940100]      <1>       mov    esi, [SourceFilePath]
4036 0000A0D1 BF[26930100]        <1>       mov    edi, FindFile_Drv
4037 0000A0D6 E877150000          <1>       call   parse_path_name
4038 0000A0DB 0F8290EEFFFF        <1>       jc     loc_cmd_failed
4039                              <1>
4040                              <1> loc_rename_sf_check_filename_exists:
4041 0000A0E1 BE[68930100]        <1>       mov    esi, FindFile_Name
```

```
4042 0000A0E6 803E20              <1>       cmp   byte [esi], 20h
4043 0000A0E9 0F8682EEFFFF        <1>       jna   loc_cmd_failed
4044                              <1>
4045                              <1>       ;mov  [DelFile_FNPointer], esi
4046                              <1>
4047                              <1> loc_rename_sf_drv:
4048                              <1>       ;mov  dh, [Current_Drv]
4049                              <1>       ;mov  [RUN_CDRV], dh
4050                              <1>
4051 0000A0EF 8A15[26930100]      <1>       mov   dl, [FindFile_Drv]
4052 0000A0F5 38F2                <1>       cmp   dl, dh ; dh = [Current_Drv]
4053 0000A0F7 740B                <1>       je    short rename_sf_change_directory
4054                              <1>
4055 0000A0F9 E8E0DFFFFF          <1>       call  change_current_drive
4056 0000A0FE 0F82ABF7FFFF        <1>       jc    loc_file_rw_cmd_failed
4057                              <1>
4058                              <1> rename_sf_change_directory:
4059 0000A104 803D[27930100]20    <1>       cmp   byte [FindFile_Directory], 20h
4060 0000A10B 7618                <1>       jna   short rename_sf_find
4061                              <1>
4062 0000A10D FE05[51400100]      <1>       inc   byte [Restore_CDIR]
4063 0000A113 BE[27930100]        <1>       mov   esi, FindFile_Directory
4064 0000A118 30E4                <1>       xor   ah, ah ; CD_COMMAND sign -> 0
4065 0000A11A E81D0F0000          <1>       call  change_current_directory
4066 0000A11F 0F828AF7FFFF        <1>       jc    loc_file_rw_cmd_failed
4067                              <1>
4068                              <1> ;rename_sf_change_prompt_dir_string:
4069                              <1>       ;call  change_prompt_dir_string
4070                              <1>
4071                              <1> rename_sf_find:
4072                              <1>       ;mov  esi, [DelFile_FNPointer]
4073 0000A125 BE[68930100]        <1>       mov   esi, FindFile_Name
4074                              <1>
4075 0000A12A 66B80008            <1>       mov   ax, 0800h ; Except volume labels
4076 0000A12E E84EF3FFFF          <1>       call  find_first_file
4077 0000A133 0F8276F7FFFF        <1>       jc    loc_file_rw_cmd_failed
4078                              <1>
4079                              <1> loc_rename_sf_ambgfn_check:
4080 0000A139 6621D2              <1>       and   dx, dx ; Ambiguous filename chars used sign (DX>0)
4081                              <1>       ;     (Note: It was BX in TRDOS v1)
4082                              <1>       ;jz   short loc_rename_sf_found
4083 0000A13C 0F8571FCFFFF        <1>       jnz   loc_file_not_found
4084                              <1>
4085                              <1>       ;mov  eax, 2 ; File not found sign
4086                              <1>       ;stc
4087                              <1>       ;jmp  loc_file_rw_cmd_failed
4088                              <1>
4089                              <1> loc_rename_sf_found:
4090                              <1>       ; EDI = Directory buffer entry offset/address
4091                              <1>       ; BL = File (or Directory) Attributes
4092                              <1>       ;     (Note: It was 'CL' in TRDOS v1)
4093                              <1>       ; mov  bl, [EDI+0Bh]
4094                              <1>
4095 0000A142 F6C307              <1>       test  bl, 07h ; Attributes, S-H-R
4096 0000A145 0F856EF7FFFF        <1>       jnz   loc_permission_denied
4097                              <1>
4098 0000A14B BE[26930100]        <1>         mov     esi, FindFile_Drv
4099 0000A150 BF[6C940100]        <1>         mov     edi, SourceFile_Drv
4100 0000A155 B920000000          <1>       mov   ecx, 32
4101 0000A15A F3A5                <1>       rep   movsd
4102                              <1>
4103                              <1> loc_rename_df_parse_path_name:
4104 0000A15C 8B35[68940100]      <1>       mov   esi, [DestinationFilePath]
4105 0000A162 BF[26930100]        <1>       mov   edi, FindFile_Drv
4106 0000A167 E8E6140000          <1>       call  parse_path_name
4107 0000A16C 7219                <1>       jc    short loc_rename_df_cmd_failed
4108                              <1>
4109                              <1>       ;mov  dh, [RUN_CDRV]
4110 0000A16E 8A35[868A0100]      <1>       mov   dh, [Current_Drv]
4111                              <1>
4112                              <1>       ; 'rename' command is valid only for same dos drive and same dir!
4113                              <1>       ; ('move' command must be used if source file and destination file
4114                              <1>       ; directories are not same!)
4115 0000A174 8A15[26930100]      <1>       mov   dl, [FindFile_Drv]
4116 0000A17A 38F2                <1>       cmp   dl, dh ; are source and destination drives different ?!
4117 0000A17C 7509                <1>       jne   short loc_rename_df_cmd_failed ; yes!
4118                              <1>
4119                              <1> rename_df_check_dirname_exists:
4120 0000A17E 803D[27930100]00    <1>       cmp   byte [FindFile_Directory], 0
4121 0000A185 760B                <1>       jna   short rename_df_check_filename_exists
4122                              <1>
4123                              <1>       ; different source file and destination file directories !
4124                              <1> loc_rename_df_cmd_failed:
4125 0000A187 B801000000          <1>       mov   eax, 1 ; TRDOS 'Bad command or file name' error
4126 0000A18C F9                  <1>       stc
4127 0000A18D E91DF7FFFF          <1>       jmp   loc_file_rw_cmd_failed
4128                              <1>
4129                              <1> rename_df_check_filename_exists:
4130 0000A192 BE[68930100]        <1>       mov   esi, FindFile_Name
4131 0000A197 E8A3F6FFFF          <1>       call  check_filename
4132 0000A19C 0F82BFF7FFFF        <1>       jc    loc_mkdir_invalid_dir_name_chars
4133                              <1>
4134                              <1>       ;mov  [DelFile_FNPointer], esi
4135                              <1>       ;cmp  byte [esi], 20h
4136                              <1>       ;ja   short loc_rename_df_find
4137                              <1>
4138                              <1>       ;mov  dh, [Current_Drv] ; dh has not been changed
4139                              <1>
4140                              <1> rename_df_drv_check_writable:
4141 0000A1A2 0FB6F6              <1>       movzx  esi, dh
4142                              <1>       ;movzx esi, byte [Current_Drv]
4143 0000A1A5 81C600010900        <1>       add   esi, Logical_DOSDisks
4144                              <1>
4145 0000A1AB 88F2                <1>       mov   dl, dh ; dl = [Current_Drv]
4146 0000A1AD 8A7601              <1>       mov   dh, [esi+LD_DiskType]
```

```
4147                                    <1>
4148 0000A1B0 80FE01                    <1>       cmp    dh, 1 ; 0 = Invalid
4149 0000A1B3 7310                      <1>       jnb    short rename_df_compare_sf_df_name
4150                                    <1>
4151                                    <1>       ; 16/10/2016 (13h -> 30)
4152 0000A1B5 B81E000000                <1>       mov    eax, 30 ; 'Disk write-protected' error
4153 0000A1BA 8B1D[68940100]            <1>       mov    ebx, [DestinationFilePath]
4154 0000A1C0 E9EAF6FFFF                <1>       jmp    loc_file_rw_cmd_failed
4155                                    <1>
4156                                    <1> rename_df_compare_sf_df_name:
4157 0000A1C5 BE[68930100]              <1>       mov    esi, FindFile_Name
4158 0000A1CA BF[AE940100]              <1>       mov    edi, SourceFile_Name
4159 0000A1CF B90C000000                <1>       mov    ecx, 12
4160                                    <1> rename_df_compare_sf_df_name_next:
4161 0000A1D4 AC                        <1>       lodsb
4162 0000A1D5 AE                        <1>       scasb
4163 0000A1D6 7506                      <1>       jne    short loc_rename_df_find
4164 0000A1D8 08C0                      <1>       or     al, al
4165 0000A1DA 74AB                      <1>       jz     short loc_rename_df_cmd_failed
4166 0000A1DC E2F6                      <1>       loop   rename_df_compare_sf_df_name_next
4167                                    <1>
4168                                    <1> loc_rename_df_find:
4169                                    <1>       ;mov   esi, [DelFile_FNPointer]
4170 0000A1DE BE[68930100]              <1>       mov    esi, FindFile_Name
4171                                    <1>
4172 0000A1E3 6631C0                    <1>       xor    ax, ax ; Any
4173 0000A1E6 E896F2FFFF                <1>       call   find_first_file
4174                                    <1>       ;jnc   short loc_rename_df_found
4175                                    <1>       ; 29/12/2017
4176 0000A1EB 0F83C8F6FFFF              <1>       jnc    loc_permission_denied
4177                                    <1>
4178                                    <1> loc_rename_df_check_error_code:
4179                                    <1>       ;cmp   eax, 2
4180 0000A1F1 3C02                      <1>       cmp    al, 2 ; Not found error
4181 0000A1F3 7406                      <1>       je     short rename_df_move_find_struct_to_dest
4182 0000A1F5 F9                        <1>       stc
4183 0000A1F6 E9B4F6FFFF                <1>       jmp    loc_file_rw_cmd_failed
4184                                    <1>
4185                                    <1> ;loc_rename_df_found:
4186                                    <1>       ; 05/11/2016
4187                                    <1>       ; Permission denied error
4188                                    <1>       ;mov   eax, ERR_PERM_DENIED ; 29/12/2017
4189                                    <1>       ;stc
4190                                    <1>       ;jmp   loc_permission_denied  ; 06/11/2016
4191                                    <1>
4192                                    <1> rename_df_move_find_struct_to_dest:
4193 0000A1FB BE[26930100]              <1>         mov     esi, FindFile_Drv
4194 0000A200 BF[EC940100]              <1>         mov     edi, DestinationFile_Drv
4195 0000A205 B920000000                <1>       mov    ecx, 32
4196 0000A20A F3A5                      <1>       rep    movsd
4197                                    <1>
4198                                    <1> loc_rename_df_process_q_sf:
4199                                    <1>       ;mov   ecx, 12
4200 0000A20C B10C                      <1>       mov    cl, 12
4201 0000A20E BE[AE940100]              <1>       mov    esi, SourceFile_Name
4202 0000A213 BF[A1450100]              <1>       mov    edi, Rename_OldName
4203                                    <1> rename_df_process_q_nml_1_sf:
4204 0000A218 AC                        <1>       lodsb
4205 0000A219 3C20                      <1>         cmp  al, 20h
4206 0000A21B 7603                      <1>         jna  short rename_df_process_q_nml_2_sf
4207 0000A21D AA                        <1>       stosb
4208 0000A21E E2F8                      <1>       loop   rename_df_process_q_nml_1_sf
4209                                    <1>
4210                                    <1> rename_df_process_q_nml_2_sf:
4211 0000A220 C60700                    <1>       mov    byte [edi], 0
4212                                    <1>
4213                                    <1> loc_rename_df_process_q_df:
4214                                    <1>       ;mov   ecx, 12
4215 0000A223 B10C                      <1>       mov    cl, 12
4216 0000A225 BE[2E950100]              <1>       mov    esi, DestinationFile_Name
4217 0000A22A BF[B2450100]              <1>       mov    edi, Rename_NewName
4218                                    <1> rename_df_process_q_nml_1_df:
4219 0000A22F AC                        <1>       lodsb
4220 0000A230 3C20                      <1>       cmp    al, 20h
4221 0000A232 7603                      <1>       jna    short loc_rename_df_process_q_nml_2_df
4222 0000A234 AA                        <1>       stosb
4223 0000A235 E2F8                      <1>       loop   rename_df_process_q_nml_1_df
4224                                    <1>
4225                                    <1> loc_rename_df_process_q_nml_2_df:
4226 0000A237 C60700                    <1>       mov    byte [edi], 0
4227                                    <1>
4228                                    <1> loc_rename_confirmation_question:
4229 0000A23A BE[79450100]              <1>       mov    esi, Msg_DoYouWantRename
4230 0000A23F E811D3FFFF                <1>       call   print_msg
4231                                    <1>
4232 0000A244 A0[C9940100]              <1>       mov    al, [SourceFile_DirEntry+11] ; Attributes
4233 0000A249 2410                      <1>       and    al, 10h
4234 0000A24B 750C                      <1>       jnz    short rename_confirmation_question_dir
4235                                    <1>
4236                                    <1> rename_confirmation_question_file:
4237 0000A24D BE[90450100]              <1>       mov    esi, Rename_File
4238 0000A252 E8FED2FFFF                <1>       call   print_msg
4239 0000A257 EB0A                      <1>       jmp    short rename_confirmation_question_as
4240                                    <1>
4241                                    <1> rename_confirmation_question_dir:
4242 0000A259 BE[96450100]              <1>       mov    esi, Rename_Directory
4243 0000A25E E8F2D2FFFF                <1>       call   print_msg
4244                                    <1>
4245                                    <1> rename_confirmation_question_as:
4246 0000A263 BE[A1450100]              <1>       mov    esi, Rename_OldName
4247 0000A268 E8E8D2FFFF                <1>       call   print_msg
4248 0000A26D BE[AE450100]              <1>       mov    esi, Msg_File_rename_as
4249 0000A272 E8DED2FFFF                <1>       call   print_msg
4250 0000A277 BE[D5440100]              <1>       mov    esi, Msg_YesNo
4251 0000A27C E8D4D2FFFF                <1>       call   print_msg
```

```
4252                                    <1>
4253                                    <1> loc_rename_ask_again:
4254 0000A281 30E4                     <1>         xor     ah, ah
4255 0000A283 E85D6CFFFF               <1>         call    int16h
4256 0000A288 3C1B                     <1>         cmp     al, 1Bh
4257 0000A28A 740F                     <1>         je      short loc_do_not_rename_file
4258 0000A28C 24DF                     <1>         and     al, 0DFh
4259 0000A28E A2[DF440100]             <1>         mov     [Y_N_nextline], al
4260 0000A293 3C59                     <1>         cmp     al, 'Y'
4261 0000A295 7404                     <1>         je      short loc_yes_rename_file
4262 0000A297 3C4E                     <1>         cmp     al, 'N'
4263 0000A299 75E6                     <1>         jne     short loc_rename_ask_again
4264                                    <1>
4265                                    <1> loc_do_not_rename_file:
4266                                    <1> loc_yes_rename_file:
4267 0000A29B E848F7FFFF               <1>         call    y_n_answer ; 29/12/2017
4268                                    <1>         ;cmp    al, 'Y' ; 'yes'
4269                                    <1>         ;cmc
4270                                    <1>           ;jnc  loc_file_rw_restore_retn
4271 0000A2A0 3C4E                     <1>         cmp     al, 'N' ; 'no'
4272 0000A2A2 0F8407F6FFFF             <1>           je    loc_file_rw_restore_retn
4273                                    <1>
4274 0000A2A8 BE[B2450100]             <1>         mov     esi, Rename_NewName
4275 0000A2AD 668B0D[E6940100]         <1>         mov     cx, [SourceFile_DirEntryNumber]
4276 0000A2B4 66A1[D2940100]           <1>         mov     ax, [SourceFile_DirEntry+20] ; First Cluster, HW
4277 0000A2BA C1E010                   <1>         shl     eax, 16 ; 13/11/2017
4278 0000A2BD 66A1[D8940100]           <1>         mov     ax, [SourceFile_DirEntry+26] ; First Cluster, LW
4279                                    <1>
4280 0000A2C3 0FB61D[BB940100]         <1>         movzx   ebx, byte [SourceFile_LongNameEntryLength]
4281 0000A2CA E8CB1B0000               <1>         call    rename_directory_entry
4282 0000A2CF E9FBF6FFFF               <1>         jmp     loc_rename_file_ok
4283                                    <1> ;loc_rename_file_ok:
4284                                    <1> ;       jc      loc_run_cmd_failed
4285                                    <1> ;       mov     esi, Msg_OK
4286                                    <1> ;       call    proc_printmsg
4287                                    <1> ;       jmp     loc_file_rw_restore_retn
4288                                    <1>
4289                                    <1> move_file:
4290                                    <1>         ; 11/03/2016
4291                                    <1>         ; 09/03/2016
4292                                    <1>         ; 08/03/2016 (TRDOS 386 = TRDOS v2.0)
4293                                    <1>         ; 21/05/2011 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_move')
4294                                    <1>         ; 23/04/2011
4295                                    <1>
4296                                    <1> get_move_source_fchar:
4297                                    <1>         ; esi = file name
4298 0000A2D4 803E20                   <1>         cmp     byte [esi], 20h
4299 0000A2D7 7614                     <1>           jna   short loc_move_nofilename_retn
4300                                    <1>
4301 0000A2D9 8935[64940100]           <1>         mov     [SourceFilePath], esi
4302                                    <1>
4303                                    <1> move_scan_source_file:
4304 0000A2DF 46                       <1>         inc     esi
4305 0000A2E0 803E20                   <1>         cmp     byte [esi], 20h
4306 0000A2E3 7409                     <1>           je      short move_scan_destination_1
4307                                    <1>         ;jb     short loc_move_nofilename_retn
4308 0000A2E5 0F8286ECFFFF             <1>         jb      loc_cmd_failed
4309 0000A2EB EBF2                     <1>         jmp     short move_scan_source_file
4310                                    <1>
4311                                    <1> loc_move_nofilename_retn:
4312 0000A2ED C3                       <1>         retn
4313                                    <1>
4314                                    <1> move_scan_destination_1:
4315 0000A2EE C60600                   <1>         mov     byte [esi], 0
4316                                    <1>
4317                                    <1> move_scan_destination_2:
4318 0000A2F1 46                       <1>         inc     esi
4319 0000A2F2 803E20                   <1>         cmp     byte [esi], 20h
4320 0000A2F5 74FA                     <1>         je      short move_scan_destination_2
4321                                    <1>         ;jb     short loc_move_nofilename_retn
4322 0000A2F7 0F8274ECFFFF             <1>         jb      loc_cmd_failed
4323                                    <1>
4324 0000A2FD 8935[68940100]           <1>         mov     [DestinationFilePath], esi
4325                                    <1>
4326                                    <1> move_scan_destination_3:
4327 0000A303 46                       <1>         inc     esi
4328 0000A304 803E20                   <1>         cmp     byte [esi], 20h
4329 0000A307 77FA                     <1>         ja      short move_scan_destination_3
4330 0000A309 C60600                   <1>         mov     byte [esi], 0
4331                                    <1>
4332                                    <1> loc_move_scan_destination_OK:
4333 0000A30C 8B35[64940100]           <1>         mov     esi, [SourceFilePath]
4334 0000A312 8B3D[68940100]           <1>         mov     edi, [DestinationFilePath]
4335                                    <1>
4336 0000A318 B001                     <1>         mov     al, 1  ; move procedure Phase 1
4337 0000A31A E8F71B0000               <1>         call    move_source_file_to_destination_file
4338 0000A31F 7328                     <1>         jnc     short move_source_file_to_destination_question
4339                                    <1>
4340                                    <1> loc_move_cmd_failed_1:
4341 0000A321 08C0                     <1>         or      al, al
4342 0000A323 0F8448ECFFFF             <1>         jz      loc_cmd_failed
4343 0000A329 3C11                     <1>         cmp     al, 11h
4344 0000A32B 740D                     <1>         je      short loc_msg_not_same_device
4345                                    <1>         ;cmp    al, 05h
4346                                    <1>         ;cmp    al, ERR_PERM_DENIED ; 29/12/2017
4347                                    <1>         ;jne    loc_run_cmd_failed
4348                                    <1>         ;jmp    loc_permission_denied
4349 0000A32D 3C0B                     <1>         cmp     al, ERR_PERM_DENIED
4350 0000A32F 0F8484F5FFFF             <1>         je      loc_permission_denied
4351 0000A335 E962ECFFFF               <1>         jmp     loc_run_cmd_failed
4352                                    <1>
4353                                    <1>         ;mov    esi, Msg_Permission_denied
4354                                    <1>         ;call   print_msg
4355                                    <1>         ;jmp    loc_file_rw_restore_retn
4356                                    <1>
```

```
4357                            <1> loc_msg_not_same_device:
4358 0000A33A BE[BF450100]      <1>         mov     esi, msg_not_same_drv
4359 0000A33F E811D2FFFF        <1>         call    print_msg
4360 0000A344 E966F5FFFF        <1>         jmp     loc_file_rw_restore_retn
4361                            <1>
4362                            <1> move_source_file_to_destination_question:
4363 0000A349 A0[6C940100]      <1>           mov     al, [SourceFile_Drv]
4364 0000A34E 0441              <1>         add     al, 'A'
4365 0000A350 A2[21460100]      <1>         mov     [msg_source_file_drv], al
4366 0000A355 A0[EC940100]      <1>           mov     al, [DestinationFile_Drv]
4367 0000A35A 0441              <1>         add     al, 'A'
4368 0000A35C A2[40460100]      <1>         mov     [msg_destination_file_drv], al
4369                            <1>
4370 0000A361 57                <1>         push    edi ; *
4371                            <1>
4372 0000A362 BE[05460100]      <1>         mov     esi, msg_source_file
4373 0000A367 E8E9D1FFFF        <1>         call    print_msg
4374 0000A36C BE[6D940100]      <1>         mov     esi, SourceFile_Directory
4375 0000A371 803E20            <1>         cmp     byte [esi], 20h
4376 0000A374 7605              <1>         jna     short msftdfq_sfn
4377 0000A376 E8DAD1FFFF        <1>         call    print_msg
4378                            <1> msftdfq_sfn:
4379 0000A37B BE[AE940100]      <1>         mov     esi, SourceFile_Name
4380 0000A380 E8D0D1FFFF        <1>         call    print_msg
4381 0000A385 BE[24460100]      <1>         mov     esi, msg_destination_file
4382 0000A38A E8C6D1FFFF        <1>         call    print_msg
4383 0000A38F BE[ED940100]      <1>         mov     esi, DestinationFile_Directory
4384 0000A394 803E20            <1>         cmp     byte [esi], 20h
4385 0000A397 7605              <1>         jna     short msftdfq_dfn
4386 0000A399 E8B7D1FFFF        <1>         call    print_msg
4387                            <1> msftdfq_dfn:
4388 0000A39E BE[2E950100]      <1>         mov     esi, DestinationFile_Name
4389 0000A3A3 E8ADD1FFFF        <1>         call    print_msg
4390 0000A3A8 BE[43460100]      <1>         mov     esi, msg_copy_nextline
4391 0000A3AD E8A3D1FFFF        <1>         call    print_msg
4392 0000A3B2 BE[43460100]      <1>         mov     esi, msg_copy_nextline
4393 0000A3B7 E899D1FFFF        <1>         call    print_msg
4394                            <1>
4395                            <1> loc_move_ask_for_new_file_yes_no:
4396 0000A3BC BE[D1450100]      <1>         mov     esi, Msg_DoYouWantMoveFile
4397 0000A3C1 E88FD1FFFF        <1>         call    print_msg
4398 0000A3C6 BE[D5440100]      <1>         mov     esi, Msg_YesNo
4399 0000A3CB E885D1FFFF        <1>         call    print_msg
4400                            <1> loc_move_ask_for_new_file_again:
4401 0000A3D0 30E4              <1>         xor     ah, ah
4402 0000A3D2 E80E6BFFFF        <1>         call    int16h
4403 0000A3D7 3C1B              <1>         cmp     al, 1Bh
4404                            <1>         ;je     short loc_do_not_move_file
4405 0000A3D9 7441              <1>         je      short loc_move_y_n_escape
4406 0000A3DB 24DF              <1>         and     al, 0DFh
4407 0000A3DD A2[DF440100]      <1>           mov     [Y_N_nextline], al
4408 0000A3E2 3C59              <1>         cmp     al, 'Y'
4409 0000A3E4 7404              <1>         je      short loc_yes_move_file
4410 0000A3E6 3C4E              <1>         cmp     al, 'N'
4411 0000A3E8 75E6              <1>         jne     short loc_move_ask_for_new_file_again
4412                            <1>
4413                            <1> loc_do_not_move_file:
4414                            <1> loc_yes_move_file:
4415 0000A3EA E8F9F5FFFF        <1>         call    y_n_answer ; 29/12/2017
4416 0000A3EF 5F                <1>         pop     edi ; *
4417                            <1>         ;cmp    al, 'Y' ; 'yes'
4418                            <1>         ;cmc
4419                            <1>           ;jnc loc_file_rw_restore_retn
4420 0000A3F0 3C4E              <1>         cmp     al, 'N' ; 'no'
4421 0000A3F2 0F84B7F4FFFF      <1>           je    loc_file_rw_restore_retn
4422                            <1>
4423                            <1> loc_move_yes_move_file:
4424 0000A3F8 B002              <1>         mov     al, 2 ; move procedure Phase 2
4425 0000A3FA E8171B0000        <1>         call    move_source_file_to_destination_file
4426                            <1>         ;jc     short loc_move_cmd_failed_2
4427 0000A3FF 0F83D0F5FFFF      <1>           jnc     move_source_file_to_destination_OK
4428                            <1>
4429                            <1> ;move_source_file_to_destination_OK:
4430                            <1> ;       mov     esi, Msg_OK
4431                            <1> ;       call    print_msg
4432                            <1> ;       jmp     loc_file_rw_restore_retn
4433                            <1>
4434                            <1> loc_move_cmd_failed_2:
4435 0000A405 3C27              <1>         cmp     al, 27h
4436 0000A407 0F858FEBFFFF      <1>         jne     loc_run_cmd_failed
4437                            <1>
4438 0000A40D BE[EA450100]      <1>         mov     esi, msg_insufficient_disk_space
4439 0000A412 E83ED1FFFF        <1>         call    print_msg
4440                            <1>
4441 0000A417 E993F4FFFF        <1>         jmp     loc_file_rw_restore_retn
4442                            <1>
4443                            <1> loc_move_y_n_escape:
4444 0000A41C B04E              <1>         mov     al, 'N' ; 'no'
4445 0000A41E EBCA              <1>         jmp     short loc_do_not_move_file
4446                            <1>
4447                            <1> copy_file:
4448                            <1>         ; 15/10/2016
4449                            <1>         ; 24/03/2016
4450                            <1>         ; 21/03/2016
4451                            <1>         ; 15/03/2016 (TRDOS 386 = TRDOS v2.0)
4452                            <1>         ; 21/05/2011 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_copy')
4453                            <1>         ; 01/08/2010
4454                            <1>
4455                            <1> get_copy_source_fchar:
4456                            <1>         ; esi = file name
4457 0000A420 803E20            <1>         cmp     byte [esi], 20h
4458 0000A423 7614              <1>           jna     short loc_copy_nofilename_retn
4459                            <1>
4460 0000A425 8935[64940100]    <1>         mov     [SourceFilePath], esi
4461                            <1>
```

```
4462                            <1> copy_scan_source_file:
4463 0000A42B 46                <1>         inc     esi
4464 0000A42C 803E20            <1>         cmp     byte [esi], 20h
4465 0000A42F 7409              <1>         je      short copy_scan_destination_1
4466                            <1>         ;jb     short loc_copy_nofilename_retn
4467 0000A431 0F823AEBFFFF      <1>         jb      loc_cmd_failed
4468 0000A437 EBF2              <1>         jmp     short copy_scan_source_file
4469                            <1>
4470                            <1> loc_copy_nofilename_retn:
4471 0000A439 C3                <1>         retn
4472                            <1>
4473                            <1> copy_scan_destination_1:
4474 0000A43A C60600            <1>         mov     byte [esi], 0
4475                            <1>
4476                            <1> copy_scan_destination_2:
4477 0000A43D 46                <1>         inc     esi
4478 0000A43E 803E20            <1>         cmp     byte [esi], 20h
4479 0000A441 74FA              <1>         je      short copy_scan_destination_2
4480                            <1>         ;jb     short loc_copy_nofilename_retn
4481 0000A443 0F8228EBFFFF      <1>         jb      loc_cmd_failed
4482                            <1>
4483 0000A449 8935[68940100]    <1>         mov     [DestinationFilePath], esi
4484                            <1>
4485                            <1> copy_scan_destination_3:
4486 0000A44F 46                <1>         inc     esi
4487 0000A450 803E20            <1>         cmp     byte [esi], 20h
4488 0000A453 77FA              <1>         ja      short copy_scan_destination_3
4489 0000A455 C60600            <1>         mov     byte [esi], 0
4490                            <1>
4491                            <1> loc_copy_save_current_drive:
4492 0000A458 8A35[868A0100]    <1>         mov     dh, [Current_Drv]
4493 0000A45E 8835[E2910100]    <1>         mov     [RUN_CDRV], dh
4494                            <1>
4495                            <1> copy_source_file_to_destination_phase_1:
4496 0000A464 8B35[64940100]    <1>         mov     esi, [SourceFilePath]
4497 0000A46A 8B3D[68940100]    <1>         mov     edi, [DestinationFilePath]
4498                            <1>
4499 0000A470 B001              <1>         mov     al, 1  ; copy procedure Phase 1
4500 0000A472 E83C1D0000        <1>         call    copy_source_file_to_destination_file
4501 0000A477 732B              <1>         jnc     short copy_source_file_to_destination_question
4502                            <1>
4503                            <1> loc_copy_cmd_failed_1:
4504                            <1>         ; 18/03/2016 (restore current drive and directory)
4505 0000A479 08C0              <1>         or      al, al
4506 0000A47B 7507              <1>         jnz     short loc_copy_cmd_failed_2
4507                            <1>
4508 0000A47D FEC0              <1>           inc     al ; mov al, 1 ; Bad command or file name !
4509 0000A47F E918EBFFFF        <1>         jmp     loc_run_cmd_failed
4510                            <1>
4511                            <1> loc_copy_cmd_failed_2:
4512 0000A484 3C27              <1>         cmp     al, 27h ; Insufficient disk space
4513 0000A486 740D              <1>         je      short loc_file_write_insuff_disk_space_msg
4514                            <1>
4515                            <1>         ; 29/12/2017
4516                            <1>         ;cmp    al, 05h
4517 0000A488 3C0B              <1>         cmp     al, ERR_PERM_DENIED
4518 0000A48A 0F850CEBFFFF      <1>         jne     loc_run_cmd_failed
4519                            <1>
4520 0000A490 E924F4FFFF        <1>         jmp     loc_permission_denied
4521                            <1>
4522                            <1> loc_file_write_insuff_disk_space_msg:
4523 0000A495 BE[EA450100]      <1>         mov     esi, msg_insufficient_disk_space
4524 0000A49A E8B6D0FFFF        <1>         call    print_msg
4525 0000A49F E90BF4FFFF        <1>           jmp     loc_file_rw_restore_retn
4526                            <1>
4527                            <1> copy_source_file_to_destination_question:
4528 0000A4A4 57                <1>         push    edi ; *
4529                            <1>
4530                            <1>         ; dh = source file attributes
4531                            <1>         ; dl > 0 -> destination file found
4532 0000A4A5 20D2              <1>         and     dl, dl
4533 0000A4A7 7449              <1>         jz      short copy_source_file_to_destination_pass_owrq
4534                            <1>
4535                            <1> loc_copy_ask_for_owr_yes_no:
4536 0000A4A9 BE[46460100]      <1>         mov     esi, Msg_DoYouWantOverWriteFile
4537 0000A4AE E8A2D0FFFF        <1>         call    print_msg
4538 0000A4B3 BE[2E950100]      <1>         mov     esi, DestinationFile_Name
4539 0000A4B8 E898D0FFFF        <1>         call    print_msg
4540 0000A4BD BE[D5440100]      <1>         mov     esi, Msg_YesNo
4541 0000A4C2 E88ED0FFFF        <1>         call    print_msg
4542                            <1>
4543                            <1> loc_copy_ask_for_owr_again:
4544 0000A4C7 30E4              <1>         xor     ah, ah
4545 0000A4C9 E8176AFFFF        <1>         call    int16h
4546 0000A4CE 3C1B              <1>         cmp     al, 1Bh
4547                            <1>           ;je     loc_do_not_copy_file
4548 0000A4D0 7419              <1>           je      short loc_copy_y_n_escape
4549 0000A4D2 24DF              <1>         and     al, 0DFh
4550 0000A4D4 A2[DF440100]      <1>           mov     [Y_N_nextline], al
4551 0000A4D9 3C59              <1>         cmp     al, 'Y'
4552 0000A4DB 0F84B1000000      <1>           je      loc_yes_copy_file
4553 0000A4E1 3C4E              <1>         cmp     al, 'N'
4554 0000A4E3 0F84A9000000      <1>           je      loc_do_not_copy_file
4555 0000A4E9 EBDC              <1>         jmp     short loc_copy_ask_for_owr_again
4556                            <1>
4557                            <1> loc_copy_y_n_escape:
4558 0000A4EB B04E              <1>         mov     al, 'N' ; 'no'
4559 0000A4ED E9A0000000        <1>           jmp     loc_do_not_copy_file
4560                            <1>
4561                            <1> copy_source_file_to_destination_pass_owrq:
4562 0000A4F2 A0[6C940100]      <1>         mov     al, [SourceFile_Drv]
4563 0000A4F7 0441              <1>         add     al, 'A'
4564 0000A4F9 A2[21460100]      <1>         mov     [msg_source_file_drv], al
4565 0000A4FE A0[EC940100]      <1>           mov     al, [DestinationFile_Drv]
4566 0000A503 0441              <1>         add     al, 'A'
```

```
4567 0000A505 A2[40460100]        <1>       mov    [msg_destination_file_drv], al
4568                              <1>
4569 0000A50A BE[05460100]        <1>       mov    esi, msg_source_file
4570 0000A50F E841D0FFFF          <1>       call   print_msg
4571 0000A514 BE[6D940100]        <1>       mov    esi, SourceFile_Directory
4572 0000A519 803E20              <1>       cmp    byte [esi], 20h
4573 0000A51C 7605                <1>       jna    short csftdfq_sfn
4574 0000A51E E832D0FFFF          <1>       call   print_msg
4575                              <1> csftdfq_sfn:
4576 0000A523 BE[AE940100]        <1>       mov    esi, SourceFile_Name
4577 0000A528 E828D0FFFF          <1>       call   print_msg
4578 0000A52D BE[24460100]        <1>       mov    esi, msg_destination_file
4579 0000A532 E81ED0FFFF          <1>       call   print_msg
4580 0000A537 BE[ED940100]        <1>       mov    esi, DestinationFile_Directory
4581 0000A53C 803E20              <1>       cmp    byte [esi], 20h
4582 0000A53F 7605                <1>       jna    short csftdfq_dfn
4583 0000A541 E80FD0FFFF          <1>       call   print_msg
4584                              <1> csftdfq_dfn:
4585 0000A546 BE[2E950100]        <1>       mov    esi, DestinationFile_Name
4586 0000A54B E805D0FFFF          <1>       call   print_msg
4587 0000A550 BE[43460100]        <1>       mov    esi, msg_copy_nextline
4588 0000A555 E8FBCFFFFF          <1>       call   print_msg
4589 0000A55A BE[43460100]        <1>       mov    esi, msg_copy_nextline
4590 0000A55F E8F1CFFFFF          <1>       call   print_msg
4591                              <1>
4592                              <1> loc_copy_ask_for_new_file_yes_no:
4593 0000A564 BE[65460100]        <1>       mov    esi, Msg_DoYouWantCopyFile
4594 0000A569 E8E7CFFFFF          <1>       call   print_msg
4595 0000A56E BE[D5440100]        <1>       mov    esi, Msg_YesNo
4596 0000A573 E8DDCFFFFF          <1>       call   print_msg
4597                              <1>
4598                              <1> loc_copy_ask_for_new_file_again:
4599 0000A578 30E4                <1>       xor    ah, ah
4600 0000A57A E86669FFFF          <1>       call   int16h
4601 0000A57F 3C1B                <1>       cmp    al, 1Bh
4602 0000A581 740F                <1>       je     short loc_do_not_copy_file
4603 0000A583 24DF                <1>       and    al, 0DFh
4604 0000A585 A2[DF440100]        <1>         mov     [Y_N_nextline], al
4605 0000A58A 3C59                <1>       cmp    al, 'Y'
4606 0000A58C 7404                <1>       je     short loc_yes_copy_file
4607 0000A58E 3C4E                <1>       cmp    al, 'N'
4608 0000A590 75E6                <1>       jne    short loc_copy_ask_for_new_file_again
4609                              <1>
4610                              <1> loc_do_not_copy_file:
4611                              <1> loc_yes_copy_file:
4612 0000A592 E851F4FFFF          <1>       call   y_n_answer ; 29/12/2017
4613 0000A597 5F                  <1>       pop    edi ; *
4614                              <1>       ;cmp   al, 'Y' ; 'yes'
4615                              <1>       ;cmc
4616                              <1>         ;jnc loc_file_rw_restore_retn
4617 0000A598 3C4E                <1>       cmp    al, 'N' ; 'no'
4618 0000A59A 0F840FF3FFFF        <1>         je   loc_file_rw_restore_retn
4619                              <1>
4620                              <1> copy_source_file_to_destination_pass_q:
4621 0000A5A0 B002                <1>       mov    al, 2  ; copy procedure Phase 2
4622 0000A5A2 E80C1C0000          <1>       call   copy_source_file_to_destination_file
4623                              <1>       ;jc    short loc_file_write_check_disk_space_err
4624                              <1>
4625                              <1>       ; 24/03/2016
4626                              <1>       ;push  cx
4627 0000A5A7 51                  <1>       push   ecx ; 29/12/2017
4628 0000A5A8 BE[43460100]        <1>       mov    esi, msg_copy_nextline
4629 0000A5AD E8A3CFFFFF          <1>       call   print_msg
4630 0000A5B2 58                  <1>       pop    eax ; 29/12/2017
4631                              <1>       ;;pop  cx
4632                              <1>       ;pop   ax
4633                              <1>
4634                              <1>       ;or    cl, cl
4635 0000A5B3 08C0                <1>       or     al, al
4636 0000A5B5 7419                <1>       jz     short copy_source_file_to_destination_OK
4637                              <1>
4638                              <1>       ; 15/10/2016 (1Dh -> 18)
4639                              <1>       ; 18/03/2016 (1Dh)
4640                              <1>       ;cmp   cl, 18 ; write error
4641 0000A5B7 3C12                <1>       cmp    al, 18
4642 0000A5B9 7506                <1>       jne    short copy_source_file_to_destination_not_OK
4643                              <1>       ;
4644                              <1>       ;mov   al, cl ; error number (write fault!)
4645 0000A5BB F9                  <1>       stc
4646 0000A5BC E9EEF2FFFF          <1>       jmp    loc_file_rw_cmd_failed
4647                              <1>
4648                              <1> copy_source_file_to_destination_not_OK:
4649 0000A5C1 BE[7E460100]        <1>       mov    esi, Msg_read_file_error_before_EOF
4650 0000A5C6 E88ACFFFFF          <1>       call   print_msg
4651 0000A5CB E9DFF2FFFF          <1>       jmp    loc_file_rw_restore_retn
4652                              <1>
4653                              <1> copy_source_file_to_destination_OK:
4654 0000A5D0 BE[E3440100]        <1>       mov    esi, Msg_OK
4655 0000A5D5 E87BCFFFFF          <1>       call   print_msg
4656                              <1>
4657 0000A5DA E9D0F2FFFF          <1>       jmp    loc_file_rw_restore_retn
4658                              <1>
4659                              <1> ;loc_file_write_check_disk_space_err:
4660                              <1>       ;cmp   al, 27h ; Insufficient disk space
4661                              <1>       ;je    loc_file_write_insuff_disk_space_msg
4662                              <1>         ;jb  loc_file_rw_cmd_failed
4663                              <1>
4664                              <1>       ;call  print_misc_error_msg ; 15/03/2016
4665                              <1>         ;jmp loc_file_rw_restore_retn
4666                              <1>
4667                              <1> change_fs_file_attributes:
4668                              <1>       ; 04/03/2016 ; Temporary
4669                              <1>       ; AL = File or directory attributes
4670                              <1>       ; AH = 0 -> Attributes are in MS-DOS format
4671                              <1>       ; AH > 0 -> Attributes are in SINGLIX format
```

```
4672                            <1>       ;push  ebx
4673                            <1>       ; ... do somethings here ...
4674                            <1>       ;pop   ebx
4675                            <1>       ; BL = File or directory attributes
4676 0000A5DF C3                <1>       retn
4677                            <1>
4678                            <1> set_get_env:
4679                            <1>       ; 11/04/2016 (TRDOS 386 = TRDOS v2.0)
4680                            <1>       ; 02/09/2011 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_set')
4681                            <1>       ; 2005 - 28/08/2011
4682                            <1> get_setenv_fchar:
4683                            <1>       ; esi = environment variable/string
4684 0000A5E0 8A06              <1>       mov    al, [esi]
4685 0000A5E2 3C20              <1>       cmp    al, 20h
4686 0000A5E4 771E              <1>       ja     short loc_find_env
4687                            <1>
4688 0000A5E6 BE00300900        <1>       mov    esi, Env_Page
4689                            <1> loc_print_setline:
4690 0000A5EB 803E00            <1>       cmp    byte [esi], 0
4691 0000A5EE 7613              <1>       jna    short loc_setenv_retn
4692 0000A5F0 E860CFFFFF        <1>       call   print_msg
4693 0000A5F5 56                <1>       push   esi
4694 0000A5F6 BE[EF4C0100]      <1>       mov    esi, nextline
4695 0000A5FB E855CFFFFF        <1>       call   print_msg
4696 0000A600 5E                <1>       pop    esi
4697 0000A601 EBE8              <1>       jmp    short loc_print_setline
4698                            <1>
4699                            <1> loc_setenv_retn:
4700 0000A603 C3                <1>       retn
4701                            <1>
4702                            <1> loc_find_env:
4703 0000A604 3C3D              <1>       cmp    al, '='
4704 0000A606 0F8465E9FFFF      <1>       je     loc_cmd_failed
4705                            <1>
4706 0000A60C 56                <1>       push   esi
4707                            <1> loc_repeat_env_equal_check:
4708 0000A60D 46                <1>       inc    esi
4709 0000A60E 803E3D            <1>       cmp    byte [esi], '='
4710 0000A611 7431              <1>       je     short pass_env_equal_check
4711 0000A613 803E20            <1>       cmp    byte [esi], 20h
4712 0000A616 73F5              <1>       jnb    short loc_repeat_env_equal_check
4713 0000A618 C60600            <1>       mov    byte [esi], 0
4714 0000A61B 5E                <1>       pop    esi
4715 0000A61C BF[868B0100]      <1>       mov    edi, TextBuffer ; out buffer
4716 0000A621 B9FF000000        <1>       mov    ecx, 255 ; maximum size (limit)
4717 0000A626 30C0              <1>       xor    al, al ; 0 -> use [ESI]
4718 0000A628 E89E000000        <1>       call   get_environment_string
4719 0000A62D 72D4              <1>       jc     short loc_setenv_retn
4720                            <1>
4721 0000A62F BE[868B0100]      <1>       mov    esi, TextBuffer
4722 0000A634 E81CCFFFFF        <1>       call   print_msg
4723 0000A639 BE[EF4C0100]      <1>       mov    esi, nextline
4724 0000A63E E812CFFFFF        <1>       call   print_msg
4725                            <1>
4726 0000A643 C3                <1>       retn
4727                            <1>
4728                            <1> pass_env_equal_check:
4729 0000A644 46                <1>       inc    esi
4730 0000A645 803E20            <1>       cmp    byte [esi], 20h
4731 0000A648 73FA              <1>       jnb    short pass_env_equal_check
4732 0000A64A C60600            <1>       mov    byte [esi], 0
4733                            <1>
4734                            <1> loc_call_set_env_string:
4735 0000A64D 5E                <1>       pop    esi
4736 0000A64E E83B010000        <1>       call   set_environment_string
4737 0000A653 73AE              <1>       jnc    short loc_setenv_retn
4738                            <1>
4739                            <1> loc_set_cmd_failed:
4740 0000A655 3C08              <1>       cmp    al, 08h
4741 0000A657 0F8514E9FFFF      <1>       jne    loc_cmd_failed
4742                            <1>
4743 0000A65D BE[BE460100]      <1>       mov    esi, Msg_No_Set_Space
4744 0000A662 E8ECEFFFFF        <1>       call   print_msg
4745                            <1>
4746 0000A667 C3                <1>       retn
4747                            <1>
4748                            <1> set_get_path:
4749                            <1>       ; 11/04/2016 (TRDOS 386 = TRDOS v2.0)
4750                            <1>       ; 03/09/2011 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_path')
4751                            <1>       ; 2005
4752                            <1> get_path_fchar:
4753                            <1>       ; esi = path
4754 0000A668 803E20            <1>       cmp    byte [esi], 20h
4755 0000A66B 7737              <1>       ja     short loc_set_path
4756                            <1>
4757 0000A66D BE00300900        <1>       mov    esi, Env_Page
4758                            <1> loc_print_path:
4759 0000A672 803E00            <1>       cmp    byte [esi], 0
4760 0000A675 762C              <1>       jna    short loc_path_retn
4761                            <1>
4762 0000A677 BE[1D410100]      <1>       mov    esi, Cmd_Path ; 'PATH' address
4763 0000A67C BF[868B0100]      <1>       mov    edi, TextBuffer ; out buffer
4764 0000A681 30C0              <1>       xor    al, al  ; use [ESI]
4765 0000A683 B9FF000000        <1>       mov    ecx, 255 ; maximum size (limit)
4766 0000A688 E83E000000        <1>       call   get_environment_string
4767 0000A68D 7214              <1>       jc     short loc_path_retn
4768                            <1>
4769 0000A68F BE[868B0100]      <1>       mov    esi, TextBuffer
4770 0000A694 E8BCCEFFFF        <1>       call   print_msg
4771 0000A699 BE[EF4C0100]      <1>       mov    esi, nextline
4772 0000A69E E8B2CEFFFF        <1>       call   print_msg
4773                            <1>
4774                            <1> loc_path_retn:
4775 0000A6A3 C3                <1>       retn
4776                            <1>
```

```
                                <1> loc_set_path:
0000A6A4 56                     <1>       push   esi
                                <1> loc_set_path_find_end:
0000A6A5 46                     <1>       inc    esi
0000A6A6 803E20                 <1>       cmp    byte [esi], 20h
0000A6A9 73FA                   <1>       jnb    short loc_set_path_find_end
0000A6AB C60600                 <1>       mov    byte [esi], 0
                                <1> loc_set_path_header:
0000A6AE 5E                     <1>       pop    esi
                                <1> set_path_x: ; 31/12/2017 ('syspath')
0000A6AF 4E                     <1>       dec    esi
0000A6B0 C6063D                 <1>       mov    byte [esi], '='
0000A6B3 4E                     <1>       dec    esi
0000A6B4 C60648                 <1>       mov    byte [esi], 'H'
0000A6B7 4E                     <1>       dec    esi
0000A6B8 C60654                 <1>       mov    byte [esi], 'T'
0000A6BB 4E                     <1>       dec    esi
0000A6BC C60641                 <1>       mov    byte [esi], 'A'
0000A6BF 4E                     <1>       dec    esi
0000A6C0 C60650                 <1>       mov    byte [esi], 'P'
                                <1>
                                <1> loc_path_call_set_env_string:
0000A6C3 E8C6000000             <1>       call   set_environment_string
0000A6C8 728B                   <1>         jc   short loc_set_cmd_failed
                                <1>
0000A6CA 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>
0000A6CB BA00300900             <1>       mov    edx, Env_Page
0000A6D0 803A00                 <1>       cmp    byte [edx], 0
0000A6D3 7474                   <1>       jz     short get_env_string_with_word_stc_retn
                                <1>
0000A6D5 66890D[F0950100]       <1>       mov    [env_var_length], cx
                                <1>
0000A6DC 51                     <1>       push   ecx ; *
0000A6DD 56                     <1>       push   esi ; **
                                <1>
0000A6DE 08C0                   <1>       or     al, al
0000A6E0 7449                   <1>       jz     short get_env_string_with_word
                                <1>
                                <1> get_env_string_with_seq_number:
0000A6E2 B101                   <1>       mov    cl, 1
0000A6E4 88C5                   <1>       mov    ch, al
0000A6E6 31C0                   <1>       xor    eax, eax
0000A6E8 89D6                   <1>       mov    esi, edx ; Env_Page
                                <1>
                                <1> get_env_string_seq_number_check:
0000A6EA 38CD                   <1>       cmp    ch, cl
0000A6EC 7726                   <1>       ja     short get_env_string_seq_number_next
                                <1>
                                <1> get_env_string_move_to_buff:
0000A6EE 57                     <1>       push   edi ; ***
                                <1>
0000A6EF 29D2                   <1>       sub    edx, edx
                                <1>
                                <1> get_env_string_seq_number_repeat1:
0000A6F1 42                     <1>       inc    edx
0000A6F2 AC                     <1>       lodsb
0000A6F3 AA                     <1>       stosb
                                <1>
0000A6F4 66FF0D[F0950100]       <1>       dec    word [env_var_length]
0000A6FB 7508                   <1>       jnz    short get_env_string_seq_number_repeat3
                                <1>
                                <1> get_env_string_seq_number_repeat2:
0000A6FD 20C0                   <1>       and    al, al
0000A6FF 7408                   <1>       jz     short get_env_string_seq_number_ok
0000A701 42                     <1>       inc    edx
0000A702 AC                     <1>       lodsb
0000A703 EBF8                   <1>       jmp    short get_env_string_seq_number_repeat2
                                <1>
                                <1> get_env_string_seq_number_repeat3:
0000A705 08C0                   <1>       or     al, al
0000A707 75E8                   <1>       jnz    short get_env_string_seq_number_repeat1
                                <1>
                                <1> get_env_string_seq_number_ok:
0000A709 5F                     <1>       pop    edi ; ***
0000A70A 89D0                   <1>       mov    eax, edx ; Length of the environment string
                                <1>                       ; (ASCIIZ, includes ZERO tail)
0000A70C BA00300900             <1>       mov    edx, Env_Page
                                <1>
                                <1> get_env_string_stc_retn:
0000A711 5E                     <1>       pop    esi ; **
0000A712 59                     <1>       pop    ecx ; *
0000A713 C3                     <1>       retn
```

```
4882                                    <1>
4883                                    <1> get_env_string_seq_number_next:
4884 0000A714 AC                       <1>         lodsb
4885 0000A715 08C0                     <1>         or      al, al
4886 0000A717 75FB                     <1>         jnz     short get_env_string_seq_number_next
4887                                    <1>
4888 0000A719 81FE00320900             <1>         cmp     esi, Env_Page + Env_Page_Size ; +512 (+4096)
4889 0000A71F F5                       <1>         cmc
4890 0000A720 72EF                     <1>         jc      short get_env_string_stc_retn
4891                                    <1>
4892 0000A722 AC                       <1>         lodsb
4893 0000A723 3C01                     <1>         cmp     al, 1
4894 0000A725 72EA                     <1>         jb      short get_env_string_stc_retn
4895 0000A727 FEC1                     <1>         inc     cl
4896 0000A729 EBBF                     <1>         jmp     short get_env_string_seq_number_check
4897                                    <1>
4898                                    <1> get_env_string_with_word:
4899 0000A72B 31C9                     <1>         xor     ecx, ecx
4900                                    <1>
4901                                    <1> get_env_string_calc_word_length:
4902 0000A72D AC                       <1>         lodsb
4903 0000A72E 3C20                     <1>         cmp     al, 20h
4904 0000A730 7211                     <1>         jb      short get_env_string_calc_word_length_ok
4905                                    <1>         ;inc    cx
4906 0000A732 FEC1                     <1>         inc     cl
4907                                    <1>
4908 0000A734 3C61                     <1>         cmp     al, 'a'
4909 0000A736 72F5                     <1>         jb      short get_env_string_calc_word_length
4910 0000A738 3C7A                     <1>         cmp     al, 'z'
4911 0000A73A 77F1                     <1>         ja      short get_env_string_calc_word_length
4912 0000A73C 24DF                     <1>         and     al, 0DFh
4913 0000A73E 8846FF                   <1>         mov     [esi-1], al
4914 0000A741 EBEA                     <1>         jmp     short get_env_string_calc_word_length
4915                                    <1>
4916                                    <1> get_env_string_calc_word_length_ok:
4917 0000A743 08C9                     <1>         or      cl, cl
4918 0000A745 7506                     <1>         jnz     short get_env_string_calc_word_length_save
4919                                    <1>
4920 0000A747 5E                       <1>         pop     esi ; **
4921                                    <1>
4922                                    <1> get_env_string_stc_retn1:
4923 0000A748 59                       <1>         pop     ecx ; *
4924                                    <1>
4925                                    <1> get_env_string_with_word_stc_retn:
4926 0000A749 31C0                     <1>         xor     eax, eax
4927 0000A74B F9                       <1>         stc
4928 0000A74C C3                       <1>         retn
4929                                    <1>
4930                                    <1> get_env_string_calc_word_length_save:
4931 0000A74D 871C24                   <1>         xchg    ebx, [esp] ; **
4932 0000A750 89DE                     <1>         mov     esi, ebx
4933                                    <1>                 ; Start of the env string (to be searched)
4934                                    <1>
4935 0000A752 57                       <1>         push    edi ; ***
4936 0000A753 89D7                     <1>         mov     edi, edx ; Env_Page
4937                                    <1>
4938                                    <1> get_env_string_compare:
4939 0000A755 57                       <1>         push    edi ; ****
4940 0000A756 51                       <1>         push    ecx ; ***** ; Variable name length
4941                                    <1>
4942                                    <1> get_env_string_compare_rep:
4943 0000A757 AC                       <1>         lodsb
4944 0000A758 AE                       <1>         scasb
4945 0000A759 7511                     <1>         jne     short get_env_string_compare_next1
4946 0000A75B E2FA                     <1>         loop    get_env_string_compare_rep
4947                                    <1>
4948 0000A75D 803F3D                   <1>         cmp     byte [edi], '='
4949 0000A760 750A                     <1>         jne     short get_env_string_compare_next1
4950                                    <1>
4951 0000A762 59                       <1>         pop     ecx ; *****
4952 0000A763 5F                       <1>         pop     edi ; ****
4953 0000A764 89FE                     <1>         mov     esi, edi
4954 0000A766 5F                       <1>         pop     edi ; ***
4955 0000A767 871C24                   <1>         xchg    ebx, [esp] ; **
4956 0000A76A EB82                     <1>         jmp     short get_env_string_move_to_buff
4957                                    <1>
4958                                    <1> get_env_string_compare_next1:
4959 0000A76C 89FE                     <1>         mov     esi, edi
4960 0000A76E 59                       <1>         pop     ecx ; *****
4961 0000A76F 5F                       <1>         pop     edi ; ****
4962                                    <1> get_env_string_compare_next2:
4963 0000A770 81FEFF310900             <1>         cmp     esi, Env_Page + Env_Page_Size - 1 ; +511 (+4095)
4964 0000A776 7310                     <1>         jnb     short get_env_string_compare_not_ok
4965 0000A778 20C0                     <1>         and     al, al
4966 0000A77A AC                       <1>         lodsb
4967 0000A77B 75F3                     <1>         jnz     short get_env_string_compare_next2
4968 0000A77D 08C0                     <1>         or      al, al
4969 0000A77F 7407                     <1>         jz      short get_env_string_compare_not_ok
4970 0000A781 4E                       <1>         dec     esi ; 12/04/2016
4971 0000A782 89F7                     <1>         mov     edi, esi
4972 0000A784 89DE                     <1>         mov     esi, ebx
4973 0000A786 EBCD                     <1>         jmp     short get_env_string_compare
4974                                    <1>
4975                                    <1> get_env_string_compare_not_ok:
4976 0000A788 5F                       <1>         pop     edi ; ***
4977 0000A789 89DE                     <1>         mov     esi, ebx
4978 0000A78B 5B                       <1>         pop     ebx ; **
4979 0000A78C EBBA                     <1>         jmp     short get_env_string_stc_retn1
4980                                    <1>
4981                                    <1> set_environment_string:
4982                                    <1>         ; 13/04/2016
4983                                    <1>         ; 12/04/2016
4984                                    <1>         ; 11/04/2016
4985                                    <1>         ; 06/04/2016
4986                                    <1>         ; 05/04/2016 (TRDOS 386 = TRDOS v2.0)
```

```
4987                                    <1>       ; 02/09/2011 (TRDOS v1, MAINPROG.ASM)
4988                                    <1>       ; 29/08/2011
4989                                    <1>       ; 29/08/2011
4990                                    <1>       ; INPUT->
4991                                    <1>       ;      ESI = ASCIIZ environment string
4992                                    <1>       ; OUTPUT ->
4993                                    <1>       ;      ESI is not changed
4994                                    <1>       ;      CF = 1 -> Could not set,
4995                                    <1>       ;           insufficient environment space
4996                                    <1>       ;
4997                                    <1>       ; (EAX, EDX will be changed)
4998                                    <1>       ;
4999                                    <1>       ;    (EAX = Start address of the env string if > 0)
5000                                    <1>       ;    (EDX = Environment string length)
5001                                    <1>
5002 0000A78E 56                        <1>       push   esi ; *
5003                                    <1>
5004 0000A78F 31C0                      <1>       xor    eax, eax
5005                                    <1>
5006                                    <1> set_env_chk_validation1:
5007 0000A791 FEC4                      <1>       inc    ah ; variable (string) length
5008 0000A793 AC                        <1>       lodsb
5009 0000A794 3C3D                      <1>       cmp    al, '='
5010 0000A796 7415                      <1>       je     short set_env_chk_validation2
5011 0000A798 3C20                      <1>       cmp    al, 20h
5012 0000A79A 720F                      <1>       jb     short set_env_string_stc
5013                                    <1>
5014                                    <1>       ; 06/04/2016
5015 0000A79C 3C61                      <1>       cmp    al, 'a'
5016 0000A79E 72F1                      <1>       jb     short set_env_chk_validation1
5017 0000A7A0 3C7A                      <1>       cmp    al, 'z'
5018 0000A7A2 77ED                      <1>       ja     short set_env_chk_validation1
5019 0000A7A4 2C20                      <1>       sub    al, 'a'-'A'
5020 0000A7A6 8846FF                    <1>       mov    [esi-1], al
5021 0000A7A9 EBE6                      <1>       jmp    short set_env_chk_validation1
5022                                    <1>
5023                                    <1> set_env_string_stc:
5024 0000A7AB 5E                        <1>       pop    esi ; *
5025                                    <1>       ;stc
5026 0000A7AC C3                        <1>       retn
5027                                    <1>
5028                                    <1> set_env_chk_validation2:
5029 0000A7AD 51                        <1>       push   ecx ; **
5030 0000A7AE 53                        <1>       push   ebx ; ***
5031 0000A7AF 57                        <1>       push   edi ; ****
5032                                    <1>
5033                                    <1>       ; 12/04/2016
5034 0000A7B0 8B5C240C                  <1>       mov    ebx, [esp+12]
5035                                    <1>
5036                                    <1> set_env_chk_validation2w:
5037 0000A7B4 89F7                      <1>       mov    edi, esi
5038 0000A7B6 4F                        <1>       dec    edi
5039                                    <1>
5040 0000A7B7 807FFF20                  <1>       cmp    byte [edi-1], 20h
5041 0000A7BB 771A                      <1>       ja     short set_env_chk_validation2z
5042                                    <1>
5043 0000A7BD 56                        <1>       push   esi
5044 0000A7BE 89FE                      <1>       mov    esi, edi
5045 0000A7C0 4E                        <1>       dec    esi
5046                                    <1>
5047                                    <1> set_env_chk_validation2x:
5048 0000A7C1 4E                        <1>       dec    esi
5049                                    <1>
5050 0000A7C2 39DE                      <1>       cmp    esi, ebx
5051 0000A7C4 7207                      <1>       jb     short set_env_chk_validation2y
5052                                    <1>
5053 0000A7C6 4F                        <1>       dec    edi
5054                                    <1>
5055 0000A7C7 8A06                      <1>       mov    al, [esi]
5056 0000A7C9 8807                      <1>       mov    [edi], al
5057                                    <1>
5058 0000A7CB EBF4                      <1>       jmp    short set_env_chk_validation2x
5059                                    <1>
5060                                    <1> set_env_chk_validation2y:
5061 0000A7CD 5E                        <1>       pop    esi
5062                                    <1>
5063                                    <1>       ;mov   byte [ebx], 20h
5064                                    <1>
5065 0000A7CE 43                        <1>       inc    ebx
5066 0000A7CF 895C240C                  <1>       mov    [esp+12], ebx
5067                                    <1>
5068 0000A7D3 FECC                      <1>       dec    ah ; 13/04/2016
5069                                    <1>
5070 0000A7D5 EBDD                      <1>       jmp    short set_env_chk_validation2w
5071                                    <1>
5072                                    <1> set_env_chk_validation2z:
5073 0000A7D7 BA00300900                <1>       mov    edx, Env_Page
5074 0000A7DC 89D7                      <1>       mov    edi, edx
5075                                    <1>
5076                                    <1> set_env_chk_validation3:
5077 0000A7DE AC                        <1>       lodsb
5078 0000A7DF 3C20                      <1>       cmp    al, 20h
5079 0000A7E1 74FB                      <1>       je     short set_env_chk_validation3
5080                                    <1>
5081 0000A7E3 9C                        <1>       pushf
5082                                    <1>
5083                                    <1>       ; 12/04/2016
5084                                    <1> set_env_chk_validation3n:
5085 0000A7E4 3C61                      <1>       cmp    al, 'a'
5086 0000A7E6 720C                      <1>       jb     short set_env_chk_validation3c
5087 0000A7E8 3C7A                      <1>       cmp    al, 'z'
5088 0000A7EA 7705                      <1>       ja     short set_env_chk_validation3x
5089 0000A7EC 2C20                      <1>       sub    al, 'a'-'A'
5090 0000A7EE 8846FF                    <1>       mov    [esi-1], al
5091                                    <1>
```

```
                        <1> set_env_chk_validation3x:
0000A7F1 AC             <1>     lodsb
0000A7F2 EBF0           <1>     jmp   short set_env_chk_validation3n
                        <1>
                        <1> set_env_chk_validation3c:
0000A7F4 3C20           <1>     cmp   al, 20h
0000A7F6 73F9           <1>     jnb   short set_env_chk_validation3x
                        <1>
0000A7F8 803F00         <1>     cmp   byte [edi], 0
0000A7FB 7731           <1>     ja    short set_env_chk_validation4
                        <1>
0000A7FD 9D             <1>     popf
0000A7FE 7228           <1>     jb    short set_env_string_nothing
                        <1>
0000A800 B900020000     <1>     mov   ecx, Env_Page_Size ; 512 (4096)
                        <1>
0000A805 89DE           <1>     mov   esi, ebx ; 12/04/2016
                        <1>
                        <1> set_env_string_copy_to_envb:
0000A807 AC             <1>     lodsb
0000A808 3C20           <1>     cmp   al, 20h
0000A80A 720A           <1>     jb    short set_env_string_copy_to_envb_z
0000A80C AA             <1>     stosb
0000A80D E2F8           <1>     loop  set_env_string_copy_to_envb
                        <1>
                        <1>     ; 11/04/2016
0000A80F 89D7           <1>     mov   edi, edx ; Env_Page
0000A811 B900020000     <1>     mov   ecx, Env_Page_Size
                        <1>
                        <1> set_env_string_copy_to_envb_z:
0000A816 52             <1>     push  edx  ; Start address of the variable
0000A817 BA00020000     <1>     mov   edx, Env_Page_Size
0000A81C 29CA           <1>     sub   edx, ecx ; variable (string) length
                        <1>
0000A81E 28C0           <1>     sub   al, al ; 0
0000A820 F3AA           <1>     rep   stosb ; clear remain bytes of the env page
                        <1>
0000A822 58             <1>     pop   eax  ; Start address of the variable
                        <1>
                        <1> set_env_string_allocate_envb_retn:  ; stc or clc return
0000A823 5F             <1>     pop   edi ; ****
0000A824 5B             <1>     pop   ebx ; ***
0000A825 59             <1>     pop   ecx ; **
0000A826 5E             <1>     pop   esi ; *
0000A827 C3             <1>     retn
                        <1>
                        <1> set_env_string_nothing:
0000A828 31C0           <1>     xor   eax, eax
0000A82A 31D2           <1>     xor   edx, edx ; 11/04/2016
0000A82C EBF5           <1>     jmp   short set_env_string_allocate_envb_retn
                        <1>
                        <1> set_env_chk_validation4:
                        <1>     ; 11/04/2016
0000A82E 9D             <1>     popf
                        <1>
0000A82F 89D6           <1>     mov   esi, edx  ; Env_Page
                        <1>
                        <1> set_env_chk_validation5:
0000A831 89DF           <1>     mov   edi, ebx  ; ASCIIZ environment string address
0000A833 0FB6CC         <1>     movzx ecx, ah ; Variable (string) length (with '=')
                        <1>
                        <1> set_env_chk_validation5_loop:
0000A836 AC             <1>     lodsb
0000A837 AE             <1>     scasb
0000A838 750A           <1>     jne   short set_env_chk_validation6
0000A83A E2FA           <1>     loop  set_env_chk_validation5_loop
                        <1>
0000A83C 3C3D           <1>     cmp   al, '='
0000A83E 0F8483000000   <1>       je      set_env_change_variable
                        <1>
                        <1> set_env_chk_validation6:
0000A844 08C0           <1>     or    al, al ; 0
0000A846 7403           <1>     jz    short set_env_chk_validation7
                        <1>
0000A848 AC             <1>     lodsb
0000A849 EBF9           <1>     jmp   short set_env_chk_validation6
                        <1>
                        <1> set_env_chk_validation7:
0000A84B 88E1           <1>     mov   cl, ah
0000A84D 01F1           <1>     add   ecx, esi
0000A84F 81F9FF310900   <1>     cmp   ecx, Env_Page + Env_Page_Size - 1
                        <1>           ; 511 (4095)
                        <1>           ; strlen + '=' + 0
0000A855 72DA           <1>     jb    short set_env_chk_validation5
                        <1>
                        <1> set_env_chk_validation8: ; variable not found
0000A857 0FB6F4         <1>     movzx esi, ah  ; variable name length (with '=')
0000A85A 01DE           <1>     add   esi, ebx ; position just after of the '='
                        <1>
                        <1> set_env_chk_validation8_loop:
0000A85C AC             <1>     lodsb
0000A85D 3C20           <1>     cmp   al, 20h
0000A85F 74FB           <1>     je    short set_env_chk_validation8_loop
0000A861 72C5           <1>     jb    short set_env_string_nothing
                        <1>
                        <1> set_env_chk_validation9:
0000A863 AC             <1>     lodsb
0000A864 3C20           <1>     cmp   al, 20h
0000A866 73FB           <1>     jnb   short set_env_chk_validation9
                        <1>
                        <1>     ; End of ASCIIZ environment string
                        <1>
                        <1> set_env_add_variable:
0000A868 29DE           <1>     sub   esi, ebx ; variable+definition length
                        <1>
```

```
5197 0000A86A 56                <1>       push   esi ; *****
5198                            <1>
5199 0000A86B 89D6              <1>       mov    esi, edx ; Environment page address
5200                            <1>
5201 0000A86D B900020000        <1>       mov    ecx, Env_Page_Size ; 512 (4096)
5202                            <1>
5203                            <1> set_env_add_variable_loop:
5204 0000A872 AC                <1>       lodsb
5205 0000A873 20C0              <1>       and    al, al
5206 0000A875 7406              <1>       jz     short set_env_add_variable_chk1 ; 0
5207 0000A877 E2F9              <1>       loop   set_env_add_variable_loop
5208                            <1>
5209                            <1>       ; 11/04/2016
5210 0000A879 884EFF            <1>       mov    [esi-1], cl ; 0
5211 0000A87C 41                <1>       inc    ecx
5212                            <1>
5213                            <1> set_env_add_variable_chk1:
5214 0000A87D 49                <1>       dec    ecx
5215 0000A87E 7408              <1>       jz     short set_env_add_variable_nspc
5216 0000A880 AC                <1>       lodsb
5217 0000A881 08C0              <1>       or     al, al
5218 0000A883 740C              <1>       jz     short set_env_add_variable_chk2 ; 00
5219 0000A885 49                <1>       dec    ecx
5220 0000A886 75EA              <1>       jnz    short set_env_add_variable_loop
5221                            <1>
5222                            <1> set_env_add_variable_nspc: ; no space on environment page
5223 0000A888 58                <1>       pop    eax ; *****
5224 0000A889 B808000000        <1>       mov    eax, 8 ; No space for new environment string
5225 0000A88E F9                <1>       stc
5226 0000A88F EB92              <1>         jmp     short set_env_string_allocate_envb_retn
5227                            <1>
5228                            <1> set_env_add_variable_chk2:
5229 0000A891 8B0C24            <1>       mov    ecx, [esp] ; *****
5230 0000A894 4E                <1>       dec    esi ; beginning address of the new variable
5231 0000A895 89F0              <1>       mov    eax, esi
5232 0000A897 01C8              <1>       add    eax, ecx ; string length (with CR)
5233 0000A899 81C200020000      <1>       add    edx, Env_Page_Size ; 512 (4096)
5234 0000A89F 39D0              <1>       cmp    eax, edx
5235 0000A8A1 77E5              <1>       ja     short set_env_add_variable_nspc
5236 0000A8A3 49                <1>       dec    ecx ; except CR at the end
5237 0000A8A4 89CA              <1>       mov    edx, ecx ; 12/04/2016
5238 0000A8A6 89F7              <1>       mov    edi, esi
5239 0000A8A8 893C24            <1>       mov    [esp], edi ; ***** ; Start address of new variable
5240 0000A8AB 89DE              <1>       mov    esi, ebx ; ASCIIZ environment string address
5241 0000A8AD F3A4              <1>       rep    movsb
5242 0000A8AF 28C0              <1>       sub    al, al
5243 0000A8B1 AA                <1>       stosb
5244 0000A8B2 58                <1>       pop    eax ; ***** ; Beginning address of new variable
5245 0000A8B3 81FF00320900      <1>         cmp     edi, Env_Page + Env_Page_Size ; 12/04/2016
5246 0000A8B9 0F8364FFFFFF      <1>         jnb     set_env_string_allocate_envb_retn ; OK !
5247 0000A8BF 880F              <1>       mov    [edi], cl ; 0
5248 0000A8C1 F8                <1>       clc    ; 13/04/2016
5249 0000A8C2 E95CFFFFFF        <1>         jmp     set_env_string_allocate_envb_retn ; OK !
5250                            <1>
5251                            <1> set_env_change_variable:
5252                            <1>       ; 06/04/2016
5253                            <1>       ; esi = Variable's address in environment page (after '=')
5254                            <1>       ; edi = ASCIIZ environment string address (after '=')
5255                            <1>
5256                            <1>       ; ah = variable length from start to the '='
5257 0000A8C7 8825[F0950100]     <1>       mov    [env_var_length], ah
5258                            <1>
5259 0000A8CD 28C9              <1>       sub    cl, cl ; ecx = 0
5260                            <1>
5261 0000A8CF 57                <1>       push   edi ; *****
5262                            <1>
5263 0000A8D0 89F7              <1>       mov    edi, esi ; 11/04/2016
5264                            <1>
5265                            <1> set_env_change_variable_calc1:
5266 0000A8D2 AC                <1>       lodsb
5267 0000A8D3 08C0              <1>       or     al, al
5268 0000A8D5 7403              <1>       jz     short set_env_change_variable_calc2
5269                            <1>
5270 0000A8D7 41                <1>       inc    ecx ; length of environment string (after the '=')
5271                            <1>
5272 0000A8D8 EBF8              <1>       jmp    short set_env_change_variable_calc1
5273                            <1>
5274                            <1> set_env_change_variable_calc2:
5275 0000A8DA 8B3424            <1>       mov    esi, [esp] ; ASCIIZ environment string address
5276                            <1>
5277 0000A8DD 29D2              <1>       sub    edx, edx
5278                            <1>
5279                            <1> set_env_change_variable_calc3:
5280 0000A8DF AC                <1>       lodsb
5281 0000A8E0 3C20              <1>       cmp    al, 20h
5282 0000A8E2 7203              <1>       jb     short set_env_change_variable_calc4
5283                            <1>
5284 0000A8E4 42                <1>       inc    edx ; length of ASCIIZ string (after the '=')
5285                            <1>
5286 0000A8E5 EBF8              <1>       jmp    short set_env_change_variable_calc3
5287                            <1>
5288                            <1> set_env_change_variable_calc4:
5289 0000A8E7 C646FF00          <1>       mov    byte [esi-1], 0  ; put ZERO instead of CR
5290                            <1>
5291 0000A8EB 5E                <1>       pop    esi ; ***** ; ASCIIZ string address (after '=')
5292                            <1>
5293                            <1>       ; EDI = Old variable's address (after '=')
5294                            <1>
5295                            <1>       ; compare the new string with the old string
5296 0000A8EC 39CA              <1>       cmp    edx, ecx
5297 0000A8EE 7717              <1>       ja     short set_env_change_variable_calc5 ; longer
5298 0000A8F0 0F828F000000      <1>         jb      set_env_change_variable_calc9 ; shorter
5299                            <1>
5300                            <1>       ;same length (simple copy)
5301 0000A8F6 0FB6C4            <1>       movzx  eax, ah
```

```
5302 0000A8F9 01C2                <1>       add    edx, eax
5303 0000A8FB F7D8                <1>       neg    eax
5304 0000A8FD 01F8                <1>       add    eax, edi
5305                              <1>       ; EAX = Start address of the variable
5306                              <1>       ; EDX = Variable length (without ZERO at the end of variable)
5307                              <1>
5308 0000A8FF F3A4                <1>       rep    movsb
5309 0000A901 F8                  <1>       clc    ; 13/04/2016
5310 0000A902 E91CFFFFFF          <1>         jmp     set_env_string_allocate_envb_retn ; OK !
5311                              <1>
5312                              <1> set_env_change_variable_calc5:
5313                              <1>       ; 11/04/2016
5314 0000A907 52                  <1>       push   edx ; *****
5315 0000A908 29CA                <1>       sub    edx, ecx ; difference ; (the new string is longer)
5316 0000A90A 89F3                <1>       mov    ebx, esi
5317 0000A90C 89FE                <1>       mov    esi, edi
5318                              <1>
5319                              <1> set_env_change_variable_calc6:
5320 0000A90E AC                  <1>       lodsb
5321 0000A90F 20C0                <1>       and    al, al
5322 0000A911 75FB                <1>       jnz    short set_env_change_variable_calc6
5323                              <1>
5324 0000A913 81FE00320900        <1>       cmp    esi, Env_Page + Env_Page_Size ; 512 (4096)
5325 0000A919 0F8369FFFFFF        <1>         jnb     set_env_add_variable_nspc
5326                              <1>
5327 0000A91F 89F9                <1>       mov    ecx, edi  ; current (old) variable's address
5328 0000A921 89F7                <1>       mov    edi, esi  ; next variable's address
5329                              <1>
5330 0000A923 AC                  <1>       lodsb
5331 0000A924 08C0                <1>       or     al, al
5332 0000A926 7416                <1>       jz     short set_env_change_variable_calc8 ; 00
5333                              <1>
5334                              <1> set_env_change_variable_calc7:
5335 0000A928 AC                  <1>       lodsb
5336 0000A929 20C0                <1>       and    al, al
5337 0000A92B 75FB                <1>       jnz    short set_env_change_variable_calc7
5338                              <1>
5339 0000A92D 81FE00320900        <1>       cmp    esi, Env_Page + Env_Page_Size ; 512 (4096)
5340 0000A933 0F834FFFFFFF        <1>         jnb     set_env_add_variable_nspc
5341                              <1>
5342 0000A939 AC                  <1>       lodsb
5343 0000A93A 08C0                <1>       or     al, al
5344 0000A93C 75EA                <1>       jnz    short set_env_change_variable_calc7
5345                              <1>
5346                              <1> set_env_change_variable_calc8:
5347 0000A93E 4E                  <1>       dec    esi ; address of the second (last) 0 of the 00
5348                              <1>
5349 0000A93F 01F2                <1>       add    edx, esi ; final position of the last 0
5350                              <1>
5351 0000A941 81FA00320900        <1>       cmp    edx, Env_Page + Env_Page_Size ; 512 (4096)
5352 0000A947 0F833BFFFFFF        <1>         jnb     set_env_add_variable_nspc
5353                              <1>
5354 0000A94D 89C8                <1>       mov    eax, ecx ; old variable's address (after '=')
5355                              <1>
5356 0000A94F 89F1                <1>       mov    ecx, esi
5357 0000A951 29F9                <1>       sub    ecx, edi ; count of bytes to move forward
5358                              <1>
5359                              <1>       ; 13/04/2016
5360 0000A953 C60200              <1>       mov    byte [edx], 0
5361 0000A956 89D7                <1>       mov    edi, edx
5362 0000A958 29F2                <1>       sub    edx, esi ; difference (additional byte count)
5363 0000A95A 4F                  <1>       dec    edi ; the last zero address (first byte of the 00)
5364 0000A95B 89FE                <1>       mov    esi, edi
5365 0000A95D 29D6                <1>       sub    esi, edx ; - displacement
5366                              <1>
5367 0000A95F FA                  <1>       cli    ; disable interrupts
5368 0000A960 FD                  <1>       std    ; backward
5369                              <1>
5370 0000A961 F3A4                <1>       rep    movsb ; move ECX bytes from DS:ESI to ES:EDI
5371                              <1>
5372 0000A963 FC                  <1>       cld    ; forward (default)
5373 0000A964 FB                  <1>       sti    ; enable interrupts
5374                              <1>
5375 0000A965 89C7                <1>       mov    edi, eax
5376 0000A967 59                  <1>       pop    ecx ; ***** ; byte count (after '=')
5377 0000A968 89CA                <1>       mov    edx, ecx
5378 0000A96A 89DE                <1>       mov    esi, ebx ; ASCIIZ string address (after '=')
5379 0000A96C 89FB                <1>       mov    ebx, edi
5380                              <1>
5381 0000A96E F3A4                <1>       rep    movsb
5382                              <1>
5383 0000A970 880F                <1>       mov    [edi], cl ; 0 ; end of variable
5384                              <1>
5385 0000A972 0FB605[F0950100]    <1>       movzx  eax, byte [env_var_length]
5386 0000A979 01C2                <1>       add    edx, eax ; variable length (total)
5387 0000A97B F7D8                <1>       neg    eax
5388 0000A97D 01D8                <1>       add    eax, ebx ; start address of the variable
5389 0000A97F F8                  <1>       clc    ; 13/04/2016
5390 0000A980 E99EFEFFFF          <1>         jmp     set_env_string_allocate_envb_retn ; OK !
5391                              <1>
5392                              <1> set_env_change_variable_calc9:
5393                              <1>       ; 11/04/2016
5394 0000A985 21D2                <1>       and    edx, edx ; is empty ?
5395 0000A987 753B                <1>       jnz    short set_env_change_variable_calc15
5396                              <1>
5397 0000A989 0FB6DC              <1>       movzx  ebx, ah
5398 0000A98C F7DB                <1>       neg    ebx
5399 0000A98E 01FB                <1>       add    ebx, edi
5400                              <1>
5401                              <1>       ; EBX = Start address of the variable (in env page)
5402                              <1>       ; EDX = Variable length = 0
5403                              <1>
5404 0000A990 89FE                <1>       mov    esi, edi
5405                              <1>
5406                              <1> set_env_change_variable_calc10:
```

```
5407 0000A992 AC                <1>       lodsb
5408 0000A993 08C0              <1>       or     al, al
5409 0000A995 75FB              <1>       jnz    short set_env_change_variable_calc10
5410                            <1>
5411 0000A997 B9FF310900        <1>       mov    ecx, Env_Page + Env_Page_Size - 1
5412                            <1>
5413 0000A99C 39CE              <1>       cmp    esi, ecx ; +511 (+4095)
5414 0000A99E 7604              <1>       jna    short set_env_change_variable_calc11
5415                            <1>
5416 0000A9A0 89CE              <1>       mov    esi, ecx
5417 0000A9A2 8806              <1>       mov    [esi], al ; 0
5418                            <1>
5419                            <1> set_env_change_variable_calc11:
5420 0000A9A4 89DF              <1>       mov    edi, ebx ; old variable's start address
5421                            <1>
5422                            <1> set_env_change_variable_calc12:
5423 0000A9A6 AC                <1>       lodsb
5424 0000A9A7 AA                <1>       stosb
5425 0000A9A8 20C0              <1>       and    al, al
5426 0000A9AA 75FA              <1>       jnz    short set_env_change_variable_calc12
5427 0000A9AC 39CE              <1>       cmp    esi, ecx
5428 0000A9AE 7706              <1>       ja     short set_env_change_variable_calc13
5429 0000A9B0 AC                <1>       lodsb
5430 0000A9B1 AA                <1>       stosb
5431 0000A9B2 20C0              <1>       and    al, al
5432 0000A9B4 75F0              <1>       jnz    short set_env_change_variable_calc12
5433                            <1>
5434                            <1> set_env_change_variable_calc13:
5435 0000A9B6 29F9              <1>       sub    ecx, edi
5436 0000A9B8 7203              <1>       jb     short set_env_change_variable_calc14
5437 0000A9BA 41                <1>       inc    ecx ; 1-512 (1-4096)
5438 0000A9BB F3AA              <1>       rep    stosb ; al = 0
5439                            <1>
5440                            <1> set_env_change_variable_calc14:
5441 0000A9BD 29C0              <1>       sub    eax, eax ; Start address of the variable
5442                            <1>       ; EAX = 0 -> Variable is removed
5443                            <1>       ; EDX = Variable length = 0
5444                            <1>
5445 0000A9BF E95FFEFFFF        <1>         jmp     set_env_string_allocate_envb_retn ; OK !
5446                            <1>
5447                            <1> set_env_change_variable_calc15:
5448 0000A9C4 52                <1>       push   edx ; *****
5449 0000A9C5 F7DA              <1>       neg    edx
5450 0000A9C7 01CA              <1>       add    edx, ecx ; difference (the old string is longer)
5451 0000A9C9 89F3              <1>       mov    ebx, esi
5452 0000A9CB 89FE              <1>       mov    esi, edi
5453                            <1>
5454                            <1> set_env_change_variable_calc16:
5455 0000A9CD AC                <1>       lodsb
5456 0000A9CE 20C0              <1>       and    al, al
5457 0000A9D0 75FB              <1>       jnz    short set_env_change_variable_calc16
5458                            <1>
5459 0000A9D2 B900320900        <1>       mov    ecx, Env_Page + Env_Page_Size
5460                            <1>
5461 0000A9D7 39CE              <1>       cmp    esi, ecx ; +512 (+4096)
5462 0000A9D9 7605              <1>       jna    short set_env_change_variable_calc17
5463                            <1>
5464 0000A9DB 89CE              <1>       mov    esi, ecx
5465 0000A9DD 8846FF            <1>       mov    [esi-1], al ; 0
5466                            <1>
5467                            <1> set_env_change_variable_calc17:
5468 0000A9E0 89F9              <1>       mov    ecx, edi  ; current (old) variable's address
5469 0000A9E2 89F7              <1>       mov    edi, esi  ; next variable's address
5470                            <1>
5471 0000A9E4 AC                <1>       lodsb
5472 0000A9E5 08C0              <1>       or     al, al
5473 0000A9E7 741D              <1>       jz     short set_env_change_variable_calc20
5474                            <1>
5475                            <1> set_env_change_variable_calc18:
5476 0000A9E9 AC                <1>       lodsb
5477 0000A9EA 20C0              <1>       and    al, al
5478 0000A9EC 75FB              <1>       jnz    short set_env_change_variable_calc18
5479                            <1>
5480 0000A9EE 81FE00320900      <1>       cmp    esi, Env_Page + Env_Page_Size
5481 0000A9F4 720B              <1>       jb     short set_env_change_variable_calc19
5482 0000A9F6 740E              <1>       je     short set_env_change_variable_calc20
5483                            <1>
5484 0000A9F8 BEFF310900        <1>       mov    esi, Env_Page + Env_Page_Size - 1
5485 0000A9FD 8806              <1>       mov    [esi], al ; 0
5486 0000A9FF EB06              <1>       jmp    short set_env_change_variable_calc21
5487                            <1>
5488                            <1> set_env_change_variable_calc19:
5489 0000AA01 AC                <1>       lodsb
5490 0000AA02 08C0              <1>       or     al, al
5491 0000AA04 75E3              <1>       jnz    short set_env_change_variable_calc18
5492                            <1>
5493                            <1> set_env_change_variable_calc20:
5494 0000AA06 4E                <1>       dec    esi ; address of the second (last) 0 of the 00
5495                            <1>
5496                            <1> set_env_change_variable_calc21:
5497                            <1>       ; edx = difference (byte count)
5498                            <1>
5499 0000AA07 89C8              <1>       mov    eax, ecx ; old variable's address (after '=')
5500                            <1>
5501 0000AA09 89F1              <1>       mov    ecx, esi
5502 0000AA0B 29F9              <1>       sub    ecx, edi ; count of bytes to move backward
5503                            <1>
5504 0000AA0D 89FE              <1>       mov    esi, edi ; next variable's address
5505 0000AA0F 29D7              <1>       sub    edi, edx ; (displacement)
5506                            <1>
5507 0000AA11 F3A4              <1>       rep    movsb
5508                            <1>
5509 0000AA13 880F              <1>       mov    [edi], cl ; 0 ; 00 ; end of environment variables
5510                            <1>
5511 0000AA15 89C7              <1>       mov    edi, eax
```

```
5512 0000AA17 5A                  <1>       pop    edx ; ***** ; byte count (after '=')
5513 0000AA18 89D1                <1>       mov    ecx, edx
5514 0000AA1A 89DE                <1>       mov    esi, ebx ; ASCIIZ string address (after '=')
5515 0000AA1C 89FB                <1>       mov    ebx, edi
5516                              <1>
5517 0000AA1E F3A4                <1>       rep    movsb
5518                              <1>
5519 0000AA20 880F                <1>       mov    [edi], cl ; 0 ; end of variable
5520                              <1>
5521 0000AA22 0FB605[F0950100]    <1>       movzx  eax, byte [env_var_length]
5522 0000AA29 01C2                <1>       add    edx, eax ; variable length (total)
5523 0000AA2B F7D8                <1>       neg    eax
5524 0000AA2D 01D8                <1>       add    eax, ebx ; start address of the variable
5525 0000AA2F F8                  <1>       clc    ; 13/04/2016
5526 0000AA30 E9EEFDFFFF          <1>         jmp     set_env_string_allocate_envb_retn ; OK !
5527                              <1>
5528                              <1> mainprog_startup_configuration:
5529                              <1>       ; 22/11/2017
5530                              <1>       ; 06/05/2016
5531                              <1>       ; 14/04/2016 (TRDOS 386 = TRDOS v2.0)
5532                              <1>       ; 17/09/2011 (TRDOS v1, MAINPROG.ASM)
5533                              <1>       ;
5534                              <1> loc_load_mainprog_cfg_file:
5535 0000AA35 BE[97400100]        <1>       mov    esi, MainProgCfgFile
5536 0000AA3A 66B80018            <1>       mov    ax, 1800h ; Except volume label and dirs
5537 0000AA3E E83EEAFFFF          <1>       call   find_first_file
5538 0000AA43 7256                <1>       jc     short loc_load_mainprog_cfg_exit
5539                              <1>
5540                              <1>       ;or    eax, eax
5541                              <1>       ;jz    short loc_load_mainprog_cfg_exit
5542                              <1>
5543                              <1> loc_start_mainprog_configuration:
5544                              <1>       ; ESI = FindFile_DirEntry Location
5545                              <1>       ; EAX = File Size
5546                              <1>
5547 0000AA45 A3[748A0100]        <1>       mov    [MainProgCfg_FileSize], eax
5548                              <1>
5549 0000AA4A 668B5614            <1>       mov    dx, [esi+DirEntry_FstClusHI]
5550 0000AA4E C1E210              <1>       shl    edx, 16
5551 0000AA51 668B561A            <1>       mov    dx, [esi+DirEntry_FstClusLO]
5552 0000AA55 8915[A4950100]      <1>       mov    [csftdf_sf_cluster], edx
5553                              <1>
5554 0000AA5B 89C1                <1>       mov    ecx, eax
5555 0000AA5D 29C0                <1>       sub    eax, eax
5556                              <1>
5557                              <1>       ; TRDOS 386 (TRDOS v2.0)
5558                              <1>       ; Allocate contiguous memory block for loading the file
5559                              <1>
5560                              <1>       ; eax = 0 (Allocate memory from the beginning)
5561                              <1>       ; ecx = File (Allocation) size in bytes
5562                              <1>
5563 0000AA5F E819BAFFFF          <1>       call   allocate_memory_block
5564 0000AA64 7235                <1>       jc     short loc_load_mainprog_cfg_exit
5565                              <1>
5566 0000AA66 A3[9C950100]        <1>       mov    [csftdf_sf_mem_addr], eax ; loading address
5567 0000AA6B 890D[A0950100]      <1>       mov    [csftdf_sf_mem_bsize], ecx ; block size
5568                              <1>
5569 0000AA71 31DB                <1>       xor    ebx, ebx
5570                              <1>       ;mov   [csftdf_sf_rbytes], ebx ; 0, reset
5571                              <1>
5572 0000AA73 8A3D[868A0100]      <1>       mov    bh, [Current_Drv] ; [FindFile_Drv]
5573 0000AA79 BE00010900          <1>       mov    esi, Logical_DOSDisks
5574 0000AA7E 01DE                <1>       add    esi, ebx
5575                              <1>
5576 0000AA80 8B1D[9C950100]      <1>       mov    ebx, [csftdf_sf_mem_addr] ; memory block address
5577                              <1>
5578 0000AA86 807E0300            <1>       cmp    byte [esi+LD_FATType], 0
5579 0000AA8A 7710                <1>         ja   short loc_mcfg_load_fat_file
5580                              <1>
5581 0000AA8C C705[AC950100]0000- <1>       mov    dword [csftdf_r_size], 65536
5581 0000AA94 0100                <1>
5582 0000AA96 E9A1010000          <1>         jmp     loc_mcfg_load_fs_file
5583                              <1>
5584                              <1> loc_load_mainprog_cfg_exit:
5585 0000AA9B C3                  <1>       retn
5586                              <1>
5587                              <1> loc_mcfg_load_fat_file:
5588 0000AA9C 0FB74611            <1>       movzx  eax, word [esi+LD_BPB+BytesPerSec]
5589 0000AAA0 0FB64E13            <1>       movzx  ecx, byte [esi+LD_BPB+SecPerClust]
5590 0000AAA4 F7E1                <1>       mul    ecx
5591 0000AAA6 A3[AC950100]        <1>       mov    [csftdf_r_size], eax
5592                              <1>
5593                              <1> loc_mcfg_load_fat_file_next:
5594 0000AAAB E822010000          <1>       call   mcfg_read_fat_file_sectors
5595 0000AAB0 0F8206010000        <1>         jc      mcfg_deallocate_mem
5596                              <1>
5597 0000AAB6 09D2                <1>       or     edx, edx ; edx > 0 -> EOF
5598 0000AAB8 74F1                <1>       jz     short loc_mcfg_load_fat_file_next
5599                              <1>
5600                              <1> loc_mcfg_load_fat_file_ok:
5601                              <1>       ; 06/05/2016
5602 0000AABA C705[40960100]-     <1>       mov    dword [mainprog_return_addr], loc_mcfg_ci_return_addr
5602 0000AAC0 [7DAB0000]          <1>
5603                              <1>       ;
5604 0000AAC4 8B35[9C950100]      <1>       mov    esi, [csftdf_sf_mem_addr]
5605 0000AACA 8935[788A0100]      <1>       mov    [MainProgCfg_LineOffset], esi
5606                              <1>
5607 0000AAD0 A1[748A0100]        <1>       mov    eax, [MainProgCfg_FileSize]
5608 0000AAD5 89C2                <1>       mov    edx, eax
5609 0000AAD7 01F2                <1>       add    edx, esi
5610                              <1>
5611                              <1> loc_mcfg_process_next_line_check:
5612 0000AAD9 89C1                <1>       mov    ecx, eax
5613                              <1>
5614 0000AADB 803E2A              <1>       cmp    byte [esi], "*" ; Remark sign
```

```
5615 0000AADE 7503              <1>       jne    short loc_mcfg_process_next_line
5616 0000AAE0 46                <1>       inc    esi
5617 0000AAE1 EB17              <1>       jmp    short loc_move_mainprog_cfg_nl1
5618                            <1>
5619                            <1> loc_mcfg_process_next_line:
5620 0000AAE3 83F94F            <1>       cmp    ecx, 79
5621 0000AAE6 7605              <1>       jna    short loc_start_mainprog_cfg_process
5622                            <1>
5623 0000AAE8 B94F000000        <1>       mov    ecx, 79
5624                            <1>
5625                            <1> loc_start_mainprog_cfg_process:
5626 0000AAED BF[368B0100]      <1>       mov    edi, CommandBuffer
5627                            <1>
5628                            <1> loc_move_mainprog_cfg_line:
5629 0000AAF2 AC                <1>       lodsb
5630 0000AAF3 3C20              <1>       cmp    al, 20h
5631 0000AAF5 720C              <1>       jb     short loc_move_mainprog_cfg_nl2
5632 0000AAF7 AA                <1>       stosb
5633 0000AAF8 E2F8              <1>       loop   loc_move_mainprog_cfg_line
5634                            <1>
5635                            <1> loc_move_mainprog_cfg_nl1:
5636 0000AAFA 39D6              <1>       cmp    esi, edx ; + configuration file size
5637 0000AAFC 7312              <1>       jnb    short loc_end_of_mainprog_cfg_line
5638 0000AAFE AC                <1>       lodsb
5639 0000AAFF 3C20              <1>       cmp    al, 20h
5640 0000AB01 73F7              <1>       jnb    short loc_move_mainprog_cfg_nl1
5641                            <1>
5642                            <1> loc_move_mainprog_cfg_nl2:
5643 0000AB03 39D6              <1>       cmp    esi, edx
5644 0000AB05 7309              <1>       jnb    short loc_end_of_mainprog_cfg_line
5645 0000AB07 8A06              <1>       mov    al, [esi]
5646 0000AB09 3C20              <1>       cmp    al, 20h
5647 0000AB0B 7703              <1>       ja     short loc_end_of_mainprog_cfg_line
5648 0000AB0D 46                <1>       inc    esi
5649 0000AB0E EBF3              <1>       jmp    short loc_move_mainprog_cfg_nl2
5650                            <1>
5651                            <1> loc_end_of_mainprog_cfg_line:
5652 0000AB10 C60700            <1>       mov    byte [edi], 0
5653                            <1>
5654 0000AB13 8935[788A0100]    <1>       mov    [MainProgCfg_LineOffset], esi
5655                            <1>
5656                            <1>       ; 22/11/2017
5657 0000AB19 BE[3E8B0100]      <1>       mov    esi, CommandBuffer + 8
5658 0000AB1E 29FE              <1>       sub    esi, edi
5659 0000AB20 7606              <1>       jna    short loc_move_mainprog_cfg_command
5660 0000AB22 30C0              <1>       xor    al, al
5661                            <1> loc_mainprog_cfg_clear_chrs:
5662 0000AB24 AA                <1>       stosb
5663 0000AB25 4E                <1>       dec    esi
5664 0000AB26 75FC              <1>       jnz    short loc_mainprog_cfg_clear_chrs
5665                            <1>
5666                            <1> loc_move_mainprog_cfg_command:
5667 0000AB28 BE[368B0100]      <1>       mov    esi, CommandBuffer
5668 0000AB2D 89F7              <1>       mov    edi, esi
5669 0000AB2F 31DB              <1>       xor    ebx, ebx
5670                            <1>       ;xor   ecx, ecx
5671 0000AB31 30C9              <1>       xor    cl, cl
5672                            <1>
5673                            <1> loc_move_mcfg_first_cmd_char:
5674 0000AB33 8A041E            <1>       mov    al, [esi+ebx]
5675 0000AB36 FEC3              <1>       inc    bl
5676 0000AB38 3C20              <1>       cmp    al, 20h
5677 0000AB3A 7712              <1>       ja     short loc_move_mcfg_cmd_capitalizing
5678 0000AB3C 7237              <1>       jb     short loc_move_mcfg_cmd_arguments_ok
5679 0000AB3E 80FB4F            <1>       cmp    bl, 79
5680 0000AB41 72F0              <1>       jb     short loc_move_mcfg_first_cmd_char
5681 0000AB43 EB30              <1>       jmp    short loc_move_mcfg_cmd_arguments_ok
5682                            <1>
5683                            <1> loc_move_mcfg_next_cmd_char:
5684 0000AB45 8A041E            <1>       mov    al, [esi+ebx]
5685 0000AB48 FEC3              <1>       inc    bl
5686 0000AB4A 3C20              <1>       cmp    al, 20h
5687 0000AB4C 7614              <1>       jna    short loc_move_mcfg_cmd_ok
5688                            <1>
5689                            <1> loc_move_mcfg_cmd_capitalizing:
5690 0000AB4E 3C61              <1>       cmp    al, 61h ; 'a'
5691 0000AB50 7206              <1>       jb     short loc_move_mcfg_cmd_caps_ok
5692 0000AB52 3C7A              <1>       cmp    al, 7Ah ; 'z'
5693 0000AB54 7702              <1>       ja     short loc_move_mcfg_cmd_caps_ok
5694 0000AB56 24DF              <1>       and    al, 0DFh ; sub      al, 'a'-'A'
5695                            <1>
5696                            <1> loc_move_mcfg_cmd_caps_ok:
5697 0000AB58 AA                <1>       stosb
5698 0000AB59 FEC1              <1>       inc    cl
5699 0000AB5B 80FB4F            <1>       cmp    bl, 79
5700 0000AB5E 72E5              <1>       jb     short loc_move_mcfg_next_cmd_char
5701 0000AB60 EB13              <1>       jmp    short loc_move_mcfg_cmd_arguments_ok
5702                            <1>
5703                            <1> loc_move_mcfg_cmd_ok:
5704 0000AB62 30C0              <1>       xor    al, al ; 0
5705                            <1>
5706                            <1> loc_move_mcfg_cmd_arguments:
5707 0000AB64 8807              <1>       mov    [edi], al
5708 0000AB66 47                <1>       inc    edi
5709 0000AB67 80FB4F            <1>       cmp    bl, 79
5710 0000AB6A 7309              <1>       jnb    short loc_move_mcfg_cmd_arguments_ok
5711 0000AB6C 8A041E            <1>       mov    al, [esi+ebx]
5712 0000AB6F FEC3              <1>       inc    bl
5713 0000AB71 3C20              <1>       cmp    al, 20h
5714 0000AB73 73EF              <1>       jnb    short loc_move_mcfg_cmd_arguments
5715                            <1>
5716                            <1> loc_move_mcfg_cmd_arguments_ok:
5717 0000AB75 C60700            <1>       mov    byte [edi], 0
5718                            <1>
5719                            <1> loc_mcfg_process_cmd_interpreter:
```

```
5720 0000AB78 E825E0FFFF          <1>       call    command_interpreter
5721                              <1>
5722                              <1> loc_mcfg_ci_return_addr:
5723 0000AB7D A1[748A0100]        <1>       mov     eax, [MainProgCfg_FileSize]
5724 0000AB82 89C2                <1>       mov     edx, eax
5725 0000AB84 8B35[788A0100]      <1>       mov     esi, [MainProgCfg_LineOffset]
5726 0000AB8A 01F2                <1>       add     edx, esi
5727 0000AB8C 0305[9C950100]      <1>       add     eax, [csftdf_sf_mem_addr]
5728 0000AB92 29F0                <1>       sub     eax, esi
5729 0000AB94 0F873FFFFFFF        <1>         ja      loc_mcfg_process_next_line_check
5730                              <1>
5731 0000AB9A E81D000000          <1>       call    mcfg_deallocate_mem
5732                              <1>
5733 0000AB9F B94F000000          <1>       mov     ecx, 79 ; 80 ?
5734 0000ABA4 BF[368B0100]        <1>       mov     edi, CommandBuffer
5735 0000ABA9 30C0                <1>       xor     al, al
5736 0000ABAB F3AA                <1>       rep     stosb
5737                              <1>
5738                              <1>       ; 06/05/2016
5739 0000ABAD BE[EF4C0100]        <1>       mov     esi, nextline
5740 0000ABB2 E89EC9FFFF          <1>       call    print_msg
5741 0000ABB7 E963D6FFFF          <1>       jmp     dos_prompt
5742                              <1>
5743                              <1> mcfg_deallocate_mem:
5744 0000ABBC A1[9C950100]        <1>       mov     eax, [csftdf_sf_mem_addr] ; start address
5745 0000ABC1 8B0D[A0950100]      <1>       mov     ecx, [csftdf_sf_mem_bsize] ; block size
5746                              <1>       ;call   deallocate_memory_block
5747                              <1>       ;retn
5748 0000ABC7 E9BEBAFFFF          <1>       jmp     deallocate_memory_block
5749                              <1>
5750                              <1> mcfg_read_file_sectors:
5751                              <1>       ; 14/04/2016
5752 0000ABCC 807E0300            <1>       cmp     byte [esi+LD_FATType], 0
5753 0000ABD0 7669                <1>         jna  short mcfg_read_fs_file_sectors
5754                              <1>
5755                              <1> mcfg_read_fat_file_sectors:
5756                              <1>       ; return:
5757                              <1>       ;   CF = 0 & EDX > 0 -> END OF FILE
5758                              <1>       ;   CF = 0 & EDX = 0 -> not EOF
5759                              <1>       ;   CF = 1 -> read error (error code in AL)
5760                              <1>
5761                              <1> mcfg_read_fat_file_secs_0:
5762 0000ABD2 8B15[748A0100]      <1>       mov     edx, [MainProgCfg_FileSize]
5763 0000ABD8 2B15[B4950100]      <1>       sub     edx, [csftdf_sf_rbytes]
5764 0000ABDE 3B15[AC950100]      <1>       cmp     edx, [csftdf_r_size]
5765 0000ABE4 7306                <1>       jnb     short mcfg_read_fat_file_secs_1
5766 0000ABE6 8915[AC950100]      <1>       mov     [csftdf_r_size], edx
5767                              <1>
5768                              <1> mcfg_read_fat_file_secs_1:
5769 0000ABEC A1[AC950100]        <1>       mov     eax, [csftdf_r_size]
5770 0000ABF1 29D2                <1>       sub     edx, edx
5771 0000ABF3 0FB74E11            <1>       movzx   ecx, word [esi+LD_BPB+BytesPerSec]
5772 0000ABF7 01C8                <1>       add     eax, ecx
5773 0000ABF9 48                  <1>       dec     eax
5774 0000ABFA F7F1                <1>       div     ecx
5775 0000ABFC 89C1                <1>       mov     ecx, eax ; sector count
5776 0000ABFE A1[A4950100]        <1>       mov     eax, [csftdf_sf_cluster]
5777                              <1>
5778                              <1>       ; EBX = memory block address (current)
5779                              <1>
5780 0000AC03 E88C230000          <1>       call    read_fat_file_sectors
5781 0000AC08 7230                <1>       jc      short mcfg_read_fat_file_secs_3
5782                              <1>
5783                              <1>       ; EBX = next memory address
5784                              <1>
5785 0000AC0A A1[B4950100]        <1>       mov     eax, [csftdf_sf_rbytes]
5786 0000AC0F 0305[AC950100]      <1>       add     eax, [csftdf_r_size]
5787 0000AC15 8B15[748A0100]      <1>       mov     edx, [MainProgCfg_FileSize]
5788 0000AC1B 39D0                <1>       cmp     eax, edx
5789 0000AC1D 731B                <1>       jnb     short mcfg_read_fat_file_secs_3 ; edx > 0
5790 0000AC1F A3[B4950100]        <1>       mov     [csftdf_sf_rbytes], eax
5791                              <1>
5792 0000AC24 53                  <1>       push    ebx ; *
5793                              <1>       ; get next cluster (csftdf_r_size! bytes)
5794 0000AC25 A1[A4950100]        <1>       mov     eax, [csftdf_sf_cluster]
5795 0000AC2A E837210000          <1>       call    get_next_cluster
5796 0000AC2F 5B                  <1>       pop     ebx ; *
5797 0000AC30 7301                <1>       jnc     short mcfg_read_fat_file_secs_2
5798                              <1>
5799                              <1>       ;mov    eax, 17; Read error !
5800 0000AC32 C3                  <1>       retn
5801                              <1>
5802                              <1> mcfg_read_fat_file_secs_2:
5803 0000AC33 29D2                <1>       sub     edx, edx ; 0
5804 0000AC35 A3[A4950100]        <1>       mov     [csftdf_sf_cluster], eax ; next cluster
5805                              <1>
5806                              <1> mcfg_read_fat_file_secs_3:
5807 0000AC3A C3                  <1>       retn
5808                              <1>
5809                              <1> mcfg_read_fs_file_sectors:
5810 0000AC3B C3                  <1>       retn
5811                              <1>
5812                              <1> loc_mcfg_load_fs_file:
5813 0000AC3C C3                  <1>       retn
5814                              <1>
5815                              <1> load_and_execute_file:
5816                              <1>       ; 04/01/2017
5817                              <1>       ; 06/05/2016, 07/05/2016, 11/05/2016
5818                              <1>       ; 23/04/2016, 24/04/2016
5819                              <1>       ; 22/04/2016 (TRDOS 386 = TRDOS v2.0)
5820                              <1>       ; 05/11/2011
5821                              <1>       ; (TRDOS v1, CMDINTR.ASM, 'cmp_cmd_run', 'cmp_cmd_external')
5822                              <1>       ; ('loc_run_check_filename')
5823                              <1>       ; 29/08/2011
5824                              <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:
0000AC3D 803E20                 <1>       cmp    byte [esi], 20h
0000AC40 0F822BE3FFFF           <1>       jb     loc_cmd_failed
0000AC46 7703                   <1>       ja     short loc_run_check_filename_ok
0000AC48 46                     <1>       inc    esi
0000AC49 EBF2                   <1>       jmp    short loc_run_check_filename
                                <1>
                                <1> loc_run_check_filename_ok:
0000AC4B C605[E78A0100]00       <1>       mov    byte [CmdArgStart], 0 ; reset
0000AC52 56                     <1>       push   esi ; *
                                <1> loc_run_get_first_arg_pos:
0000AC53 46                     <1>       inc    esi
0000AC54 8A06                   <1>       mov    al, [esi]
0000AC56 3C20                   <1>       cmp    al, 20h
0000AC58 77F9                   <1>       ja     short loc_run_get_first_arg_pos
0000AC5A C60600                 <1>       mov    byte [esi], 0
                                <1> loc_run_get_external_arg_pos:
                                <1>       ; 11/05/2016
0000AC5D 46                     <1>       inc    esi
0000AC5E 8A06                   <1>       mov    al, [esi]
0000AC60 3C20                   <1>       cmp    al, 20h
0000AC62 760C                   <1>       jna    short loc_run_parse_path_name
0000AC64 89F0                   <1>       mov    eax, esi
0000AC66 2D[368B0100]           <1>       sub    eax, CommandBuffer
0000AC6B A2[E78A0100]           <1>       mov    byte [CmdArgStart], al
                                <1> loc_run_parse_path_name:
0000AC70 5E                     <1>       pop    esi ; *
0000AC71 BF[26930100]           <1>       mov    edi, FindFile_Drv
0000AC76 E8D7090000             <1>       call   parse_path_name
0000AC7B 0F82F0E2FFFF           <1>       jc     loc_cmd_failed
                                <1>
                                <1> loc_run_check_filename_exists:
0000AC81 BE[68930100]           <1>       mov    esi, FindFile_Name
0000AC86 803E20                 <1>       cmp    byte [esi], 20h
0000AC89 0F86E2E2FFFF           <1>       jna    loc_cmd_failed
                                <1>
                                <1> loc_run_check_exe_filename_ext:
0000AC8F E890020000             <1>       call   check_prg_filename_ext
0000AC94 0F82D7E2FFFF           <1>       jc     loc_cmd_failed
                                <1>
                                <1> loc_run_check_exe_filename_ext_ok:
0000AC9A 66A3[3E960100]         <1>       mov    word [EXE_ID], ax
                                <1>
                                <1> loc_run_drv:
0000ACA0 C605[3D960100]00       <1>       mov    byte [Run_Manual_Path], 0
0000ACA7 A1[808A0100]           <1>       mov    eax, [Current_Dir_FCluster]
0000ACAC A3[38960100]           <1>         mov     [Run_CDirFC], eax
                                <1>       ;
0000ACB1 8A35[868A0100]         <1>       mov    dh, [Current_Drv]
0000ACB7 8835[E2910100]         <1>       mov    [RUN_CDRV], dh
                                <1>
0000ACBD 8A15[26930100]         <1>       mov    dl, [FindFile_Drv]
0000ACC3 38F2                   <1>       cmp    dl, dh
0000ACC5 7412                   <1>       je     short loc_run_change_directory
                                <1>
0000ACC7 8005[3D960100]02       <1>       add    byte [Run_Manual_Path], 2
                                <1>
0000ACCE E80BD4FFFF             <1>       call   change_current_drive
0000ACD3 0F82C3E2FFFF           <1>       jc     loc_run_cmd_failed
                                <1>
                                <1> loc_run_change_directory:
0000ACD9 803D[27930100]20       <1>       cmp    byte [FindFile_Directory], 20h
0000ACE0 7623                   <1>       jna    short loc_run_find_executable_file
                                <1>
0000ACE2 FE05[3D960100]         <1>       inc    byte [Run_Manual_Path]
                                <1>
0000ACE8 FE05[51400100]         <1>       inc    byte [Restore_CDIR]
                                <1>
0000ACEE BE[27930100]           <1>       mov    esi, FindFile_Directory
0000ACF3 30E4                   <1>       xor    ah, ah ; CD_COMMAND sign -> 0
0000ACF5 E842030000             <1>       call   change_current_directory
0000ACFA 0F829CE2FFFF           <1>       jc     loc_run_cmd_failed
                                <1>
                                <1> loc_run_change_prompt_dir_string:
0000AD00 E857020000             <1>       call   change_prompt_dir_string
                                <1>
                                <1> loc_run_find_executable_file:
0000AD05 66C705[3C960100]00-    <1>       mov    word [Run_Auto_Path], 0
0000AD0D 00                     <1>
                                <1>
                                <1> loc_run_find_executable_file_next:
0000AD0E BE[68930100]           <1>       mov    esi, FindFile_Name
                                <1> loc_run_find_program_file_next:
0000AD13 66B80018               <1>       mov    ax, 1800h ; Except volume label and dirs
0000AD17 E865E7FFFF             <1>       call   find_first_file
                                <1>       ; ESI = Directory Entry (FindFile_DirEntry) Location
                                <1>       ; EDI = Directory Buffer Directory Entry Location
                                <1>       ; EAX = File size
0000AD1C 0F835C010000           <1>       jnc    loc_load_and_run_file
                                <1>
0000AD22 3C02                   <1>       cmp    al, 2 ; file not found
0000AD24 0F8572E2FFFF           <1>       jne    loc_run_cmd_failed
                                <1>
0000AD2A 66A1[3E960100]         <1>       mov    ax, word [EXE_ID]
0000AD30 80FC2E                 <1>       cmp    ah, '.' ; File name has extension sign
0000AD33 7424                   <1>       je     short loc_run_check_auto_path
                                <1>
0000AD35 08C0                   <1>       or     al, al
```

```
5929 0000AD37 7520                <1>       jnz    short loc_run_check_auto_path
5930                              <1>
5931 0000AD39 80FC08              <1>       cmp    ah, 8 ; count of file name chars
5932 0000AD3C 771B                <1>       ja     short loc_run_check_auto_path
5933                              <1>
5934                              <1> loc_run_change_file_ext_to_prg:
5935 0000AD3E 0FB6DC              <1>       movzx  ebx, ah ; count of file name chars
5936 0000AD41 BE[68930100]        <1>       mov    esi, FindFile_Name
5937 0000AD46 01F3                <1>       add    ebx, esi
5938                              <1>       ; 07/05/2016
5939 0000AD48 C7032E505247        <1>       mov    dword [ebx],  '.PRG'
5940 0000AD4E 66C705[3E960100]50- <1>       mov    word [EXE_ID], 'P.'
5940 0000AD56 2E                  <1>
5941 0000AD57 EBBA                <1>       jmp    short loc_run_find_program_file_next
5942                              <1>
5943                              <1> loc_run_check_auto_path:
5944                              <1>       ; NOTE: /// 07/05/2016 ///
5945                              <1>       ; If the path is given, value of byte [Run_Manual_Path]
5946                              <1>       ; will not be ZERO. If so, file searching by using
5947                              <1>       ; Automatic Path (via 'PATH' environment variable)
5948                              <1>       ; will not be applicable, because the program file
5949                              <1>       ; is already/absolutely not found.
5950                              <1>
5951 0000AD59 A0[3D960100]        <1>       mov    al, [Run_Manual_Path]
5952 0000AD5E 08C0                <1>       or     al, al
5953 0000AD60 0F850BE2FFFF        <1>       jnz    loc_cmd_failed
5954                              <1>
5955                              <1> loc_run_check_auto_path_again:
5956 0000AD66 66833D[3C960100]FF  <1>       cmp    word [Run_Auto_Path], 0FFFFh
5957                              <1>              ; 0FFFFh = Not a valid run path (in ENV block)
5958 0000AD6E 0F83FDE1FFFF        <1>       jnb    loc_cmd_failed
5959                              <1>       ; xor  al, al
5960 0000AD74 BE[1D410100]        <1>       mov    esi, Cmd_Path ; 'PATH'
5961 0000AD79 BF[868B0100]        <1>       mov    edi, TextBuffer
5962 0000AD7E E848F9FFFF          <1>       call   get_environment_string
5963 0000AD83 730E                <1>       jnc    short loc_run_chk_filename_ext_again
5964 0000AD85 66C705[3C960100]FF- <1>       mov    word [Run_Auto_Path], 0FFFFh ; invalid
5964 0000AD8D FF                  <1>
5965 0000AD8E E9DEE1FFFF          <1>       jmp    loc_cmd_failed
5966                              <1>
5967                              <1> loc_run_chk_filename_ext_again:
5968 0000AD93 89C1                <1>       mov    ecx, eax ; string length (with zero tail)
5969 0000AD95 49                  <1>       dec    ecx ; without zero tail
5970 0000AD96 66A1[3E960100]      <1>       mov    ax, [EXE_ID]
5971 0000AD9C 80FC2E              <1>       cmp    ah, '.'
5972 0000AD9F 740E                <1>       je     short loc_run_chk_auto_path_pos
5973                              <1>
5974                              <1> loc_run_change_file_ext_to_noext_again:
5975 0000ADA1 0FB6DC              <1>       movzx  ebx, ah
5976 0000ADA4 BE[68930100]        <1>       mov    esi, FindFile_Name
5977 0000ADA9 01F3                <1>       add    ebx, esi
5978 0000ADAB 29C0                <1>       sub    eax, eax
5979 0000ADAD 8903                <1>       mov    [ebx], eax ; 0 ; erase extension (.PRG)
5980                              <1>
5981                              <1> loc_run_chk_auto_path_pos:
5982                              <1>       ;movzx eax,  word [Run_Auto_Path]
5983 0000ADAF 66A1[3C960100]      <1>       mov    ax, [Run_Auto_Path]
5984 0000ADB5 39C8                <1>       cmp    eax, ecx ; ecx = string length (except zero tail)
5985 0000ADB7 0F83B4E1FFFF        <1>       jnb    loc_cmd_failed
5986                              <1>       ;or    eax, eax
5987 0000ADBD 6609C0              <1>       or     ax, ax
5988 0000ADC0 7502                <1>       jnz    short loc_run_auto_path_pos_move
5989 0000ADC2 B005                <1>       mov    al, 5
5990                              <1>
5991                              <1> loc_run_auto_path_pos_move:
5992 0000ADC4 89FE                <1>       mov    esi, edi ; offset TextBuffer
5993 0000ADC6 01C6                <1>       add    esi, eax
5994                              <1>
5995                              <1> loc_run_auto_path_pos_space_loop:
5996 0000ADC8 AC                  <1>       lodsb
5997 0000ADC9 3C20                <1>       cmp    al, 20h
5998 0000ADCB 74FB                <1>       je     short loc_run_auto_path_pos_space_loop
5999 0000ADCD 0F829EE1FFFF        <1>       jb     loc_cmd_failed
6000 0000ADD3 AA                  <1>       stosb
6001                              <1> loc_run_auto_path_pos_move_next:
6002 0000ADD4 AC                  <1>       lodsb
6003 0000ADD5 3C3B                <1>       cmp    al, ';'
6004 0000ADD7 7414                <1>       je     short loc_run_auto_path_pos_move_last_byte
6005 0000ADD9 3C20                <1>       cmp    al, 20h
6006 0000ADDB 74F7                <1>       je     short loc_run_auto_path_pos_move_next
6007 0000ADDD 7203                <1>       jb     short loc_byte_ptr_end_of_path
6008 0000ADDF AA                  <1>       stosb
6009 0000ADE0 EBF2                <1>       jmp    short loc_run_auto_path_pos_move_next
6010                              <1>
6011                              <1> loc_byte_ptr_end_of_path:
6012 0000ADE2 66C705[3C960100]FF- <1>       mov    word [Run_Auto_Path], 0FFFFh ; end of path
6012 0000ADEA FF                  <1>
6013 0000ADEB EB0D                <1>       jmp    short loc_run_auto_path_move_ok
6014                              <1>
6015                              <1> loc_run_auto_path_pos_move_last_byte:
6016 0000ADED 89F0                <1>       mov    eax, esi
6017 0000ADEF 2D[868B0100]        <1>       sub    eax, TextBuffer
6018 0000ADF4 66A3[3C960100]      <1>       mov    [Run_Auto_Path], ax ; next path position
6019                              <1>
6020                              <1> loc_run_auto_path_move_ok:
6021 0000ADFA 4F                  <1>       dec    edi
6022 0000ADFB B02F                <1>       mov    al, '/'
6023 0000ADFD 3807                <1>       cmp    [edi], al
6024 0000ADFF 7403                <1>       je     short loc_run_auto_path_move_file_name
6025 0000AE01 47                  <1>       inc    edi
6026 0000AE02 8807                <1>       mov    [edi], al
6027                              <1>
6028                              <1> loc_run_auto_path_move_file_name:
6029 0000AE04 47                  <1>       inc    edi
6030 0000AE05 BE[68930100]        <1>       mov    esi, FindFile_Name
```

```
6031                                    <1>
6032                                    <1> loc_run_auto_path_move_fn_loop:
6033 0000AE0A AC                       <1>         lodsb
6034 0000AE0B AA                       <1>         stosb
6035 0000AE0C 08C0                     <1>         or      al, al
6036 0000AE0E 75FA                     <1>         jnz     short loc_run_auto_path_move_fn_loop
6037                                    <1>
6038 0000AE10 BE[868B0100]             <1>         mov     esi, TextBuffer
6039 0000AE15 BF[26930100]             <1>         mov     edi, FindFile_Drv
6040 0000AE1A E833080000               <1>         call    parse_path_name
6041 0000AE1F 0F824CE1FFFF             <1>         jc      loc_cmd_failed
6042                                    <1>
6043 0000AE25 8A35[868A0100]           <1>         mov     dh, [Current_Drv]
6044 0000AE2B 8A15[26930100]           <1>         mov     dl, [FindFile_Drv]
6045 0000AE31 38F2                     <1>         cmp     dl, dh
6046 0000AE33 740B                     <1>         je      short loc_run_change_directory_again
6047                                    <1>
6048 0000AE35 E8A4D2FFFF               <1>         call    change_current_drive
6049 0000AE3A 0F825CE1FFFF             <1>         jc      loc_run_cmd_failed
6050                                    <1>
6051                                    <1> loc_run_change_directory_again:
6052 0000AE40 803D[27930100]20         <1>         cmp     byte [FindFile_Directory], 20h
6053 0000AE47 761D                     <1>         jna     short loc_load_executable_cdir_chk_again
6054                                    <1>
6055 0000AE49 FE05[51400100]           <1>         inc     byte [Restore_CDIR]
6056 0000AE4F BE[27930100]             <1>         mov     esi, FindFile_Directory
6057 0000AE54 30E4                     <1>         xor     ah, ah ; CD_COMMAND sign -> 0
6058 0000AE56 E8E1010000               <1>         call    change_current_directory
6059 0000AE5B 0F823BE1FFFF             <1>         jc      loc_run_cmd_failed
6060                                    <1>
6061                                    <1> loc_run_chg_prompt_dir_str_again:
6062 0000AE61 E8F6000000               <1>         call    change_prompt_dir_string
6063                                    <1>
6064                                    <1> loc_load_executable_cdir_chk_again:
6065 0000AE66 A1[808A0100]             <1>         mov     eax, [Current_Dir_FCluster]
6066 0000AE6B 3B05[38960100]           <1>         cmp     eax, [Run_CDirFC]
6067 0000AE71 0F8597FEFFFF             <1>         jne     loc_run_find_executable_file_next
6068 0000AE77 30C0                     <1>         xor     al, al ; 0
6069 0000AE79 E9E8FEFFFF               <1>         jmp     loc_run_check_auto_path_again
6070                                    <1>
6071                                    <1> loc_load_and_run_file:
6072                                    <1>         ; 13/11/2017
6073                                    <1>         ; 04/01/2017
6074                                    <1>         ; 23/04/2016
6075 0000AE7E BE[68930100]             <1>         mov     esi, FindFile_Name
6076 0000AE83 BF[868B0100]             <1>         mov     edi, TextBuffer
6077                                    <1>
6078                                    <1>         ; 24/04/2016
6079 0000AE88 31D2                     <1>         xor     edx, edx
6080 0000AE8A 668915[4A040300]         <1>         mov     word [argc], dx ; 0
6081 0000AE91 8915[8C030300]           <1>         mov     dword [u.nread], edx ; 0
6082                                    <1>
6083                                    <1> loc_load_and_run_file_1:
6084 0000AE97 AC                       <1>         lodsb
6085 0000AE98 AA                       <1>         stosb
6086 0000AE99 FF05[8C030300]           <1>         inc     dword [u.nread]
6087 0000AE9F 20C0                     <1>         and     al, al
6088 0000AEA1 75F4                     <1>         jnz     short loc_load_and_run_file_1
6089                                    <1>
6090 0000AEA3 A0[E78A0100]             <1>         mov     al, [CmdArgStart]
6091 0000AEA8 20C0                     <1>         and     al, al
6092 0000AEAA 7445                     <1>         jz      short loc_load_and_run_file_7
6093                                    <1>
6094 0000AEAC 0FB6F0                   <1>         movzx   esi, al ; 11/05/2016
6095 0000AEAF B950000000               <1>         mov     ecx, 80
6096 0000AEB4 29F1                     <1>         sub     ecx, esi
6097 0000AEB6 81C6[368B0100]           <1>         add     esi, CommandBuffer
6098                                    <1>
6099 0000AEBC 66FF05[4A040300]         <1>         inc     word [argc] ; 11/05/2016
6100                                    <1>
6101                                    <1> loc_load_and_run_file_2:
6102 0000AEC3 AC                       <1>         lodsb
6103 0000AEC4 3C20                     <1>         cmp     al, 20h
6104 0000AEC6 7717                     <1>         ja      short loc_load_and_run_file_5
6105 0000AEC8 721E                     <1>         jb      short loc_load_and_run_file_6
6106                                    <1>
6107                                    <1> loc_load_and_run_file_3:
6108 0000AECA 803E20                   <1>         cmp     byte [esi], 20h
6109 0000AECD 7707                     <1>         ja      short loc_load_and_run_file_4
6110 0000AECF 7217                     <1>         jb      short loc_load_and_run_file_6
6111 0000AED1 46                       <1>         inc     esi
6112 0000AED2 E2F6                     <1>         loop    loc_load_and_run_file_3
6113 0000AED4 EB12                     <1>         jmp     short loc_load_and_run_file_6
6114                                    <1>
6115                                    <1> loc_load_and_run_file_4:
6116 0000AED6 28C0                     <1>         sub     al, al ; 0
6117 0000AED8 66FF05[4A040300]         <1>         inc     word [argc]
6118                                    <1> loc_load_and_run_file_5:
6119 0000AEDF AA                       <1>         stosb
6120 0000AEE0 FF05[8C030300]           <1>         inc     dword [u.nread]
6121 0000AEE6 E2DB                     <1>         loop    loc_load_and_run_file_2
6122                                    <1>
6123                                    <1> loc_load_and_run_file_6:
6124 0000AEE8 30C0                     <1>         xor     al, al ; 0
6125 0000AEEA AA                       <1>         stosb
6126 0000AEEB FF05[8C030300]           <1>         inc     dword [u.nread]
6127                                    <1> loc_load_and_run_file_7:
6128 0000AEF1 8807                     <1>         mov     [edi], al ; 0
6129 0000AEF3 66FF05[4A040300]         <1>         inc     word [argc] ; 24/04/2016
6130 0000AEFA FF05[8C030300]           <1>         inc     dword [u.nread] ; 24/04/2016
6131 0000AF00 BE[868B0100]             <1>         mov     esi, TextBuffer
6132 0000AF05 8B15[94930100]           <1>         mov     edx, [FindFile_DirEntry+DirEntry_FileSize]
6133 0000AF0B 66A1[8C930100]           <1>         mov     ax, [FindFile_DirEntry+DirEntry_FstClusHI]
6134 0000AF11 C1E010                   <1>         shl     eax, 16 ; 13/11/2017
6135 0000AF14 66A1[92930100]           <1>         mov     ax, [FindFile_DirEntry+DirEntry_FstClusLO]
```

```
6136                            <1>       ; EAX = First Cluster number
6137                            <1>       ; EDX = File Size
6138                            <1>       ; ESI = Argument list address
6139                            <1>       ; [argc] = argument count
6140                            <1>       ; [u.nread] = argument list length
6141 0000AF1A E8F8630000       <1>       call   load_and_run_file ; trdosk6.s
6142                            <1>         ;jc  loc_run_cmd_failed ; 04/01/2017
6143                            <1> loc_load_and_run_file_8: ; 06/05/2016
6144 0000AF1F E98BE9FFFF       <1>       jmp    loc_file_rw_restore_retn
6145                            <1>
6146                            <1> check_prg_filename_ext:
6147                            <1>       ; 23/04/2016 (TRDOS 386 = TRDOS v2.0)
6148                            <1>       ; 10/09/2011
6149                            <1>       ; (TRDOS v1, CMDINTR.ASM, 'proc_check_exe_filename_ext')
6150                            <1>       ; 14/11/2009
6151                            <1>       ; INPUT ->
6152                            <1>       ;      ESI = Dot File Name
6153                            <1>       ; OUTPUT ->
6154                            <1>       ;     cf = 0 -> EXE_ID in AL
6155                            <1>       ;      ESI = Last char + 1 position
6156                            <1>       ;     cf = 1 -> Invalid executable file name
6157                            <1>       ;      or no file name extension if AH<=8
6158                            <1>       ;      AL = Last file name char
6159                            <1>       ;     cf = 0 -> AL='P' (PRG), AL=0 (no extension)
6160                            <1>       ;
6161                            <1>       ; (Modified registers: EAX, ESI)
6162                            <1>
6163 0000AF24 30E4             <1>       xor    ah, ah
6164                            <1> loc_run_check_filename_ext:
6165 0000AF26 AC               <1>       lodsb
6166 0000AF27 3C21             <1>       cmp    al, 21h
6167 0000AF29 7229             <1>       jb     short loc_check_exe_fn_retn
6168 0000AF2B FEC4             <1>       inc    ah
6169 0000AF2D 3C2E             <1>       cmp    al, '.'
6170 0000AF2F 75F5             <1>       jne    short loc_run_check_filename_ext
6171                            <1>
6172                            <1> loc_run_check_filename_ext_dot:
6173 0000AF31 80FC02           <1>       cmp    ah, 2 ; .??? is not valid
6174 0000AF34 88C4             <1>       mov    ah, al ; '.'
6175 0000AF36 7219             <1>       jb     short loc_check_prg_fn_retn
6176                            <1>
6177                            <1> loc_run_check_filename_ext_dot_ok:
6178 0000AF38 AC               <1>       lodsb
6179 0000AF39 24DF             <1>       and    al, 0DFh
6180                            <1>
6181                            <1> loc_run_check_filename_ext_prg:
6182 0000AF3B 3C50             <1>       cmp    al, 'P'
6183 0000AF3D 7212             <1>       jb     short loc_check_prg_fn_retn
6184 0000AF3F 7711             <1>       ja     short loc_check_prg_fn_stc
6185 0000AF41 AC               <1>       lodsb
6186 0000AF42 24DF             <1>       and    al, 0DFh
6187 0000AF44 3C52             <1>       cmp    al, 'R'
6188 0000AF46 750A             <1>       jne    short loc_check_prg_fn_stc
6189 0000AF48 AC               <1>       lodsb
6190 0000AF49 24DF             <1>       and    al, 0DFh
6191 0000AF4B 3C47             <1>       cmp    al, 'G'
6192 0000AF4D 7503             <1>       jne    short loc_check_prg_fn_stc
6193                            <1>
6194 0000AF4F B050             <1>       mov    al, 'P'
6195                            <1> loc_check_prg_fn_retn:
6196 0000AF51 C3               <1>       retn
6197                            <1>
6198                            <1> loc_check_prg_fn_stc:
6199 0000AF52 F9               <1>       stc
6200 0000AF53 C3               <1>       retn
6201                            <1>
6202                            <1> loc_check_exe_fn_retn:
6203 0000AF54 28C0             <1>       sub    al, al ; 0
6204 0000AF56 C3               <1>       retn
6205                            <1>
6206                            <1> find_and_list_files:
6207 0000AF57 C3               <1>       retn
6208                            <1> set_exec_arguments:
6209 0000AF58 C3               <1>       retn
6210                            <1> delete_fs_directory:
6211 0000AF59 31C0             <1>       xor eax, eax
6212 0000AF5B C3               <1>       retn
3082                                %include 'trdosk4.s' ; 24/01/2016
   1                            <1> ; ****************************************************************************
   2                            <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - Directory Functions : trdosk4.s
   3                            <1> ; ----------------------------------------------------------------------------
   4                            <1> ; Last Update: 29/12/2017
   5                            <1> ; ----------------------------------------------------------------------------
   6                            <1> ; Beginning: 24/01/2016
   7                            <1> ; ----------------------------------------------------------------------------
   8                            <1> ; Assembler: NASM version 2.11 (trdos386.s)
   9                            <1> ; ----------------------------------------------------------------------------
  10                            <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
  11                            <1> ; DIR.ASM (09/10/2011)
  12                            <1> ; ****************************************************************************
  13                            <1>
  14                            <1> ; DIR.ASM  [ TRDOS KERNEL - COMMAND EXECUTER SECTION - DIRECTORY FUNCTIONS ]
  15                            <1> ; (c) 2004-2010  Erdogan TAN  [ 17/01/2004 ]  Last Update: 09/10/2011
  16                            <1> ; FILE.ASM [ FILE FUNCTIONS ] Last Update: 09/10/2011
  17                            <1>
  18                            <1> change_prompt_dir_string:
  19                            <1>       ; 05/10/2016
  20                            <1>       ; 24/01/2016 (TRDOS 386 = TRDOS v2.0)
  21                            <1>       ; 27/03/2011
  22                            <1>       ; 09/10/2009
  23                            <1>       ; INPUT/OUTPUT => none
  24                            <1>       ; this procedure changes current directory string/text
  25                            <1>       ; 2005
  26                            <1>
  27 0000AF5C BE[E3910100]      <1>       mov    esi, PATH_Array
```

```
                            <1> change_prompt_dir_str: ; 05/10/2016 (call from 'set_working_path')
0000AF61 BF[8A8A0100]       <1>       mov    edi, Current_Directory
0000AF66 8A25[848A0100]     <1>       mov    ah, [Current_Dir_Level]
0000AF6C E807000000         <1>       call   set_current_directory_string
0000AF71 880D[E58A0100]     <1>       mov    [Current_Dir_StrLen], cl
                            <1>
0000AF77 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>
0000AF78 57                 <1>       push    edi
0000AF79 80FC00             <1>       cmp     ah, 0
0000AF7C 7652               <1>       jna    short pass_write_path
0000AF7E 83C610             <1>       add    esi, 16
0000AF81 89F3               <1>       mov    ebx, esi
                            <1> loc_write_path:
0000AF83 B908000000         <1>       mov    ecx, 8
                            <1> path_write_dirname1:
0000AF88 AC                 <1>       lodsb
0000AF89 3C20               <1>       cmp    al, 20h
0000AF8B 7612               <1>       jna    short pass_write_dirname1
0000AF8D AA                 <1>       stosb
0000AF8E 81FF[E48A0100]     <1>       cmp    edi, End_Of_Current_Dir_Str
0000AF94 733A               <1>       jnb    short pass_write_path
0000AF96 E2F0               <1>       loop   path_write_dirname1
0000AF98 803E20             <1>       cmp    byte [esi], 20h
0000AF9B 7624               <1>       jna    short pass_write_dirname2
0000AF9D EB0A               <1>       jmp     short loc_put_dot_cont_ext
                            <1> pass_write_dirname1:
0000AF9F 89DE               <1>       mov    esi, ebx
0000AFA1 83C608             <1>       add    esi, 8
0000AFA4 803E20             <1>       cmp    byte [esi], 20h
0000AFA7 7618               <1>       jna    short pass_write_dirname2
                            <1> loc_put_dot_cont_ext:
0000AFA9 C6072E             <1>       mov    byte [edi], "."
                            <1>       ;mov   ecx, 3
0000AFAC B103               <1>       mov    cl, 3
                            <1> loc_check_dir_name_ext:
0000AFAE AC                 <1>       lodsb
0000AFAF 47                 <1>       inc    edi
0000AFB0 3C20               <1>       cmp    al, 20h
0000AFB2 760D               <1>       jna    short pass_write_dirname2
0000AFB4 8807               <1>       mov    [edi], al
0000AFB6 81FF[E48A0100]     <1>       cmp    edi, End_Of_Current_Dir_Str
0000AFBC 7312               <1>       jnb    short pass_write_path
0000AFBE E2EE               <1>       loop    loc_check_dir_name_ext
0000AFC0 47                 <1>       inc    edi
                            <1> pass_write_dirname2:
0000AFC1 FECC               <1>       dec    ah
0000AFC3 740B               <1>       jz      short pass_write_path
0000AFC5 83C310             <1>       add    ebx, 16
0000AFC8 89DE               <1>       mov    esi, ebx
0000AFCA C6072F             <1>       mov    byte [edi],"/"
0000AFCD 47                 <1>       inc    edi
0000AFCE EBB3               <1>       jmp    short loc_write_path
                            <1> pass_write_path:
0000AFD0 C60700             <1>       mov    byte [edi], 0
0000AFD3 47                 <1>       inc    edi
0000AFD4 89F9               <1>       mov    ecx, edi
0000AFD6 5F                 <1>       pop    edi
0000AFD7 29F9               <1>       sub    ecx, edi
                            <1>       ; ECX = Current Directory String Length
0000AFD9 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:
0000AFDA 80FA00             <1>       cmp    dl, 0
0000AFDD 7708               <1>       ja     short loc_get_current_drive_1
0000AFDF 8A15[868A0100]     <1>       mov    dl, [Current_Drv]
0000AFE5 EB17               <1>       jmp    short loc_get_current_drive_2
                            <1> loc_get_current_drive_1:
0000AFE7 FECA               <1>       dec    dl
0000AFE9 3A15[50400100]     <1>       cmp    dl, [Last_DOS_DiskNo]
0000AFEF 760D               <1>       jna    short loc_get_current_drive_2
0000AFF1 B80F000000         <1>       mov    eax, 0Fh ; Invalid drive (Drive not ready!)
```

```
0000AFF6 F5                <1>       cmc    ; stc
0000AFF7 C3                <1>       retn
                           <1>
                           <1> loc_get_current_drive_not_ready_retn:
0000AFF8 5E                <1>       pop    esi
                           <1>       ;mov   eax, 15
0000AFF9 66B80F00          <1>       mov    ax, 15 ; Drive not ready
0000AFFD C3                <1>       retn
                           <1>
                           <1> loc_get_current_drive_2:
0000AFFE 31C0              <1>       xor    eax, eax
0000B000 88D4              <1>       mov    ah, dl
0000B002 56                <1>       push   esi
0000B003 BE00010900        <1>       mov    esi, Logical_DOSDisks
0000B008 01C6              <1>       add    esi, eax
0000B00A 8A06              <1>       mov    al, [esi+LD_Name]
0000B00C 3C41              <1>       cmp    al, 'A'
0000B00E 72E8              <1>       jb     short loc_get_current_drive_not_ready_retn
                           <1>
0000B010 8A667F            <1>       mov    ah, [esi+LD_CDirLevel]
0000B013 08E4              <1>       or     ah, ah
0000B015 7506              <1>       jnz    short loc_get_current_drive_3
                           <1>
                           <1>       ;xor   ah, ah ; mov ah, 0
0000B017 8826              <1>       mov    [esi], ah
0000B019 31C9              <1>       xor    ecx, ecx
0000B01B EB1C              <1>       jmp    short loc_get_current_drive_4
                           <1>
                           <1> loc_get_current_drive_3:
0000B01D BF[E3910100]      <1>         mov     edi, PATH_Array
0000B022 57                <1>       push   edi
0000B023 81C680000000      <1>       add    esi, LD_CurrentDirectory
0000B029 B920000000        <1>       mov    ecx, 32
0000B02E F3A5              <1>       rep    movsd
0000B030 5E                <1>       pop    esi ; Path Array Address
0000B031 5F                <1>       pop    edi ; pushed esi (current dir buffer offset)
                           <1>       ;
0000B032 E841FFFFFF        <1>       call   set_current_directory_string
0000B037 89FE              <1>       mov    esi, edi
                           <1>
                           <1> loc_get_current_drive_4:
0000B039 30C0              <1>       xor    al, al
0000B03B 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>
0000B03C 8825[71920100]    <1>       mov    [CD_COMMAND], ah
0000B042 803E2F            <1>       cmp    byte [esi], '/'
0000B045 7505              <1>       jne    short loc_ccd_cdir_level
0000B047 46                <1>       inc    esi
0000B048 30C0              <1>       xor    al, al
0000B04A EB05              <1>       jmp    short loc_ccd_parse_path_name
                           <1>
                           <1> loc_ccd_cdir_level:
0000B04C A0[848A0100]      <1>       mov    al, [Current_Dir_Level]
                           <1> loc_ccd_parse_path_name:
0000B051 88C4              <1>       mov    ah, al
0000B053 BF[E3910100]      <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>
0000B058 0FB6C8            <1>       movzx  ecx, al
0000B05B FEC1              <1>       inc    cl
0000B05D C0E104            <1>       shl    cl, 4
0000B060 01CF              <1>       add    edi, ecx
0000B062 B107              <1>       mov    cl, 7
0000B064 28C1              <1>       sub    cl, al
0000B066 C0E102            <1>       shl    cl, 2
0000B069 89C3              <1>       mov    ebx, eax
0000B06B 31C0              <1>       xor    eax, eax ; 0
0000B06D F3AB              <1>       rep    stosd
0000B06F 89D8              <1>       mov    eax, ebx
                           <1>
0000B071 BF[E3910100]      <1>       mov    edi, PATH_Array
                           <1>
0000B076 803E20            <1>       cmp    byte [esi], 20h
0000B079 F5                <1>       cmc
0000B07A 7305              <1>       jnc    short pass_ccd_parse_dir_name
                           <1>
```

```
                                <1>             ; ESI = Path name
                                <1>             ; AL = CCD_Level
0000B07C 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:
0000B081 9C                     <1>         pushf
                                <1>
                                <1>         ;mov   [CCD_Level], al
                                <1>           ;mov [Last_Dir_Level], ah
0000B082 66A3[67920100]         <1>         mov    [CCD_Level], ax
                                <1>
0000B088 31DB                   <1>         xor    ebx, ebx
0000B08A 8A3D[868A0100]         <1>         mov    bh, [Current_Drv]
0000B090 BE00010900             <1>         mov    esi, Logical_DOSDisks
0000B095 01DE                   <1>         add    esi, ebx
                                <1>
0000B097 9D                     <1>         popf
0000B098 720A                   <1>         jc     short loc_ccd_bad_path_name_retn
                                <1>
0000B09A 8935[63920100]         <1>         mov    [CCD_DriveDT], esi
                                <1>
0000B0A0 3C07                   <1>         cmp    al, 7
0000B0A2 7209                   <1>         jb     short loc_ccd_load_child_dir
                                <1>
                                <1> loc_ccd_bad_path_name_retn:
0000B0A4 87F7                   <1>         xchg   esi, edi
0000B0A6 B813000000             <1>         mov    eax, 19 ; Bad directory/path name
0000B0AB F9                     <1>         stc
                                <1> loc_ccd_retn_p:
0000B0AC C3                     <1>         retn
                                <1>
                                <1> loc_ccd_load_child_dir:
                                <1>         ; AL = CCD_Level
0000B0AD 08C0                   <1>         or     al, al
0000B0AF 7468                   <1>         jz     short loc_ccd_load_root_dir
                                <1>
0000B0B1 6689C1                 <1>         mov    cx, ax
0000B0B4 C0E004                 <1>         shl    al, 4
0000B0B7 0FB6F0                 <1>         movzx  esi, al
0000B0BA 01FE                   <1>         add    esi, edi  ; offset PATH_Array
                                <1>
0000B0BC 8B460C                 <1>         mov    eax, [esi+12]
0000B0BF 38E9                   <1>         cmp    cl, ch
0000B0C1 0F84FA000000           <1>           je      loc_ccd_load_sub_directory
0000B0C7 A3[808A0100]           <1>         mov    [Current_Dir_FCluster], eax
                                <1>
                                <1> loc_ccd_load_child_dir_next:
0000B0CC 83C610                 <1>         add    esi, 16 ; DOS DirEntry Format FileName Address
                                <1>
                                <1>         ; Directory attribute : 10h
0000B0CF B010                   <1>         mov    al, 00010000b ; 10h (Attrib AND mask)
                                <1>         ;mov   ah, 11001000b ; C8h
                                <1>         ; Volume name attribute: 8h
0000B0D1 B408                   <1>         mov    ah, 00001000b ; 08h (Attrib NAND, AND --> zero mask)
                                <1>
0000B0D3 6631C9                 <1>         xor    cx, cx
0000B0D6 E8B5010000             <1>         call   locate_current_dir_file
0000B0DB 7353                   <1>         jnc    short loc_ccd_set_dir_cluster_ptr
                                <1>
                                <1>          ; 19/02/2016
                                <1>         ;mov   edi, [CCD_DriveDT]
0000B0DD 8A25[67920100]         <1>         mov    ah, [CCD_Level]
0000B0E3 803D[71920100]CD       <1>         cmp    byte [CD_COMMAND], 0CDh ;'CD' command or another
0000B0EA 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.)
0000B0EC 88E1                   <1>         mov    cl, ah
0000B0EE 50                     <1>         push   eax
0000B0EF E8E3000000             <1>         call   loc_ccd_save_current_dir
0000B0F4 58                     <1>         pop    eax
                                <1> loc_ccd_load_child_dir_err:
0000B0F5 3C03                   <1>         cmp    al, 3  ; AL = 2 => File not found error
0000B0F7 7202                   <1>         jb     short loc_ccd_path_not_found_retn
0000B0F9 F9                     <1>         stc
0000B0FA C3                     <1>         retn
                                <1>
                                <1> loc_ccd_path_not_found_retn:
0000B0FB B003                   <1>         mov    al, 3  ; Path not found
0000B0FD C3                     <1>         retn
                                <1>
                                <1> loc_ccd_load_FAT_root_dir:
0000B0FE 803D[858A0100]02       <1>         cmp    byte [Current_FATType], 2
0000B105 776B                   <1>         ja     short loc_ccd_load_FAT32_root_dir
                                <1>
                                <1>         ;mov   esi, [CCD_DriveDT]
                                <1>         ;push  esi
0000B107 E8B51D0000             <1>         call   load_FAT_root_directory
                                <1>         ;pop   edi ; Dos Drv Description Table
                                <1>
0000B10C 89F7                   <1>         mov    edi, esi
0000B10E BE[E3910100]           <1>         mov    esi, PATH_Array
0000B113 7297                   <1>         jc     short loc_ccd_retn_p
                                <1>
0000B115 31C0                   <1>         xor    eax, eax
0000B117 EB78                   <1>           jmp  short loc_ccd_set_cdfc
                                <1>
                                <1> loc_ccd_load_root_dir:
0000B119 803D[858A0100]01       <1>         cmp    byte [Current_FATType], 1
0000B120 73DC                   <1>         jnb    short loc_ccd_load_FAT_root_dir
                                <1>
```

```
                            <1> loc_ccd_load_FS_root_dir:
0000B122 E8611E0000         <1>       call   load_FS_root_directory
0000B127 EB5C               <1>       jmp    short pass_ccd_load_FAT_sub_directory
                            <1>
                            <1> loc_ccd_load_FS_sub_directory_next:
0000B129 E85B1E0000         <1>       call   load_FS_sub_directory
0000B12E EB1F               <1>       jmp    short pass_ccd_set_dir_cluster_ptr
                            <1>
                            <1> loc_ccd_set_dir_cluster_ptr:
                            <1>       ; EDI = Directory Entry
0000B130 668B4714           <1>       mov    ax, [edi+20] ; First Cluster High Word
0000B134 C1E010             <1>       shl    eax, 16
0000B137 668B471A           <1>       mov    ax, [edi+26] ; First Cluster Low Word
                            <1>
0000B13B 8B35[63920100]     <1>       mov    esi, [CCD_DriveDT]
0000B141 803D[858A0100]01   <1>       cmp    byte [Current_FATType], 1
0000B148 72DF               <1>       jb     short loc_ccd_load_FS_sub_directory_next
                            <1>       ;push  esi
0000B14A 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
0000B14F BE[E3910100]       <1>       mov    esi, PATH_Array
0000B154 7264               <1>       jc     short loc_ccd_retn_c
                            <1>
0000B156 A1[B1910100]       <1>       mov    eax, [DirBuff_Cluster]
                            <1>
0000B15B FE05[67920100]     <1>       inc    byte [CCD_Level]
0000B161 0FB61D[67920100]   <1>       movzx  ebx, byte [CCD_Level]
0000B168 C0E304             <1>       shl    bl, 4 ; * 16 (<= 128)
0000B16B 01DE               <1>       add    esi, ebx ; 19/02/2016
0000B16D 89460C             <1>       mov    [esi+12], eax
0000B170 EB1F               <1>       jmp    short loc_ccd_set_cdfc
                            <1>
                            <1> loc_ccd_load_FAT32_root_dir:
0000B172 BE[E3910100]       <1>       mov    esi, PATH_Array
0000B177 8B460C             <1>       mov    eax, [esi+12]
0000B17A 8B35[63920100]     <1>       mov    esi, [CCD_DriveDT]
                            <1>
                            <1> loc_ccd_load_FAT_sub_directory:
                            <1>       ;push  esi
0000B180 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
0000B185 BE[E3910100]       <1>       mov    esi, PATH_Array
0000B18A 722E               <1>       jc     short loc_ccd_retn_c
                            <1>
0000B18C A1[B1910100]       <1>       mov    eax, [DirBuff_Cluster]
                            <1>
                            <1> loc_ccd_set_cdfc:
0000B191 8A0D[67920100]     <1>       mov    cl, [CCD_Level]
0000B197 880D[848A0100]     <1>       mov    [Current_Dir_Level], cl
0000B19D A3[808A0100]       <1>       mov    [Current_Dir_FCluster], eax
                            <1>
0000B1A2 8A2D[68920100]     <1>       mov    ch, [Last_Dir_Level]
0000B1A8 38E9               <1>       cmp    cl, ch
0000B1AA 0F821CFFFFFF       <1>       jb     loc_ccd_load_child_dir_next
                            <1>
0000B1B0 803D[71920100]CD   <1>       cmp    byte [CD_COMMAND], 0CDh ;'CD' command or another
0000B1B7 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>
0000B1B9 F8                 <1>       clc
                            <1>
                            <1> loc_ccd_retn_c:
0000B1BA 8B3D[63920100]     <1>       mov    edi, [CCD_DriveDT]
0000B1C0 C3                 <1>       retn
                            <1>
                            <1> loc_ccd_load_sub_directory:
0000B1C1 8B35[63920100]     <1>       mov    esi, [CCD_DriveDT]
0000B1C7 803D[858A0100]01   <1>       cmp    byte [Current_FATType], 1
0000B1CE 73B0               <1>       jnb    short loc_ccd_load_FAT_sub_directory
0000B1D0 E8B41D0000         <1>       call   load_FS_sub_directory
0000B1D5 EBAE               <1>       jmp    short pass_ccd_load_FAT_sub_directory
                            <1>
                            <1> loc_ccd_save_current_dir:
0000B1D7 BE[E3910100]       <1>       mov    esi, PATH_Array ; 19/02/2016
0000B1DC 8B3D[63920100]     <1>       mov    edi, [CCD_DriveDT]
0000B1E2 57                 <1>       push   edi
0000B1E3 83C77F             <1>         add     edi, LD_CDirLevel
0000B1E6 880F               <1>       mov    [edi], cl
0000B1E8 47                 <1>       inc    edi ; LD_CurrentDirectory
0000B1E9 56                 <1>       push   esi
                            <1>       ;mov   ecx, 32  ; always < 65536 (in this procedure)
0000B1EA 66B92000           <1>       mov    cx, 32
0000B1EE F3A5               <1>       rep    movsd
                            <1>       ; Current directory has been saved to
                            <1>       ; the DOS drive description table, cdir area !
0000B1F0 5E                 <1>       pop    esi  ; PATH_Array
0000B1F1 5F                 <1>       pop    edi  ; Dos Drv Description Table
                            <1>
0000B1F2 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
0000B1F3 88C4                   <1>       mov    ah, al
0000B1F5 66A3[08930100]         <1>       mov    [PATH_CDLevel], ax
                                <1> repeat_ppdn_check_slash:
0000B1FB AC                     <1>       lodsb
0000B1FC 3C2F                   <1>       cmp    al, '/'
0000B1FE 74FB                   <1>       je     short repeat_ppdn_check_slash
0000B200 3C21                   <1>       cmp    al, 21h
0000B202 7219                   <1>       jb     short loc_ppdn_retn
0000B204 57                     <1>       push   edi
                                <1> loc_ppdn_get_dir_name:
0000B205 B90C000000             <1>       mov    ecx, 12
0000B20A BF[0A930100]           <1>       mov    edi, Dir_File_Name
                                <1> repeat_ppdn_get_dir_name:
0000B20F AA                     <1>       stosb
0000B210 AC                     <1>       lodsb
0000B211 3C2F                   <1>       cmp    al, '/'
0000B213 740A                   <1>       je     short loc_check_level_dot_conv_dir_name
0000B215 3C20                   <1>       cmp    al, 20h
0000B217 7605                   <1>       jna    short loc_ppdn_end_of_path_scan
0000B219 E2F4                   <1>       loop   repeat_ppdn_get_dir_name
0000B21B 5F                     <1>       pop    edi
0000B21C F9                     <1>       stc
                                <1> loc_ppdn_retn:
0000B21D C3                     <1>       retn
                                <1>
                                <1> loc_ppdn_end_of_path_scan:
0000B21E 4E                     <1>       dec    esi
                                <1> loc_check_level_dot_conv_dir_name:
0000B21F 31C0                   <1>       xor    eax, eax
0000B221 AA                     <1>       stosb
0000B222 89F3                   <1>       mov    ebx, esi
0000B224 BE[0A930100]           <1>       mov    esi, Dir_File_Name
0000B229 AC                     <1>       lodsb
                                <1> repeat_ppdn_name_check_dot:
0000B22A 3C2E                   <1>       cmp    al, '.'
0000B22C 7509                   <1>       jne    short loc_ppdn_convert_sub_dir_name
                                <1> repeat_ppdn_name_dot_dot:
0000B22E AC                     <1>       lodsb
0000B22F 3C2E                   <1>       cmp    al, '.'
0000B231 743E                   <1>       je     short loc_ppdn_dot_dot
0000B233 3C21                   <1>       cmp    al, 21h
0000B235 7226                   <1>       jb     short pass_ppdn_convert_sub_dir_name
                                <1> loc_ppdn_convert_sub_dir_name:
0000B237 8A25[09930100]         <1>       mov    ah, [PATH_Level]
0000B23D 80FC07                 <1>       cmp    ah, 7
0000B240 731B                   <1>       jnb    short pass_ppdn_convert_sub_dir_name
0000B242 FEC4                   <1>       inc    ah
0000B244 8825[09930100]         <1>       mov    [PATH_Level], ah
0000B24A BE[0A930100]           <1>       mov    esi, Dir_File_Name
                                <1>       ;mov   edi, [PATH_Array_Ptr]
0000B24F B010                   <1>       mov    al, 16
0000B251 F6E4                   <1>       mul    ah
0000B253 8B3C24                 <1>       mov    edi, [esp]
                                <1>       ;push  edi
0000B256 01C7                   <1>       add    edi, eax
0000B258 E82A030000             <1>       call   convert_file_name
                                <1>       ;pop   edi
                                <1> pass_ppdn_convert_sub_dir_name:
0000B25D 89DE                   <1>       mov    esi, ebx
                                <1> repeat_ppdn_check_last_slash:
0000B25F AC                     <1>       lodsb
0000B260 3C2F                   <1>       cmp    al, '/'
0000B262 74FB                   <1>       je     short repeat_ppdn_check_last_slash
0000B264 3C21                   <1>       cmp    al, 21h
0000B266 739D                   <1>       jnb    short loc_ppdn_get_dir_name
                                <1> end_of_parse_dir_name:
0000B268 5F                     <1>       pop    edi
0000B269 F5                     <1>       cmc
                                <1>       ;mov   al, [PATH_CDLevel]
                                <1>       ;mov   ah, [PATH_Level]
0000B26A 66A1[08930100]         <1>       mov    ax, [PATH_CDLevel]
0000B270 C3                     <1>       retn
                                <1>
                                <1> loc_ppdn_dot_dot:
0000B271 AC                     <1>       lodsb
0000B272 3C21                   <1>       cmp    al, 21h
0000B274 73F2                   <1>       jnb    short end_of_parse_dir_name
                                <1> loc_ppdn_dot_dot_prev_level:
0000B276 66A1[08930100]         <1>       mov    ax, [PATH_CDLevel]
0000B27C 80EC01                 <1>       sub    ah, 1
0000B27F 80D400                 <1>       adc    ah, 0
0000B282 38E0                   <1>       cmp    al, ah
0000B284 7602                   <1>       jna    short pass_ppdn_set_al_to_ah
0000B286 88E0                   <1>       mov    al, ah
                                <1> pass_ppdn_set_al_to_ah:
0000B288 66A3[08930100]         <1>       mov    [PATH_CDLevel], ax
0000B28E 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>
0000B290 8935[6B920100]                 <1>       mov   [CDLF_FNAddress], esi
0000B296 66A3[69920100]                 <1>       mov   [CDLF_AttributesMask], ax
0000B29C 66890D[6F920100]               <1>       mov   [CDLF_DEType], cx
                                        <1>
0000B2A3 31DB                           <1>       xor   ebx, ebx
0000B2A5 881D[80920100]                 <1>       mov   [PreviousAttr], bl ; 0  ; 13/02/2016
                                        <1>
0000B2AB 8A3D[868A0100]                 <1>       mov   bh, [Current_Drv]
0000B2B1 381D[AC910100]                 <1>       cmp   byte [DirBuff_ValidData], bl ; 0
0000B2B7 761D                           <1>       jna   short loc_lcdf_reload_current_dir2
0000B2B9 8A1D[AA910100]                 <1>         mov     bl, [DirBuff_DRV]
0000B2BF 80EB41                         <1>       sub   bl, 'A'
0000B2C2 38DF                           <1>       cmp   bh, bl
0000B2C4 750E                           <1>       jne   short loc_lcdf_reload_current_dir1
0000B2C6 8B15[B1910100]                 <1>       mov   edx, [DirBuff_Cluster]
0000B2CC 3B15[808A0100]                 <1>       cmp   edx, [Current_Dir_FCluster]
0000B2D2 7412                           <1>       je    short loc_cdir_locatefile_search
                                        <1>
                                        <1> loc_lcdf_reload_current_dir1:
0000B2D4 30DB                           <1>       xor   bl, bl
                                        <1> loc_lcdf_reload_current_dir2:
0000B2D6 89DE                           <1>       mov   esi, ebx
0000B2D8 81C600010900                   <1>         add     esi, Logical_DOSDisks
0000B2DE E874000000                     <1>       call  reload_current_directory
0000B2E3 735D                           <1>       jnc   short loc_locatefile_search_again
0000B2E5 C3                             <1>       retn
                                        <1>
                                        <1> loc_cdir_locatefile_search:
0000B2E6 31DB                           <1>       xor   ebx, ebx
0000B2E8 55                             <1>       push  ebp ; 20/11/2017
0000B2E9 E8A6000000                     <1>       call  find_directory_entry
0000B2EE 5D                             <1>       pop   ebp ; 20/11/2017
0000B2EF 7349                           <1>       jnc   short loc_cdir_locate_file_retn
                                        <1>
                                        <1> loc_locatefile_check_stc_reason:
0000B2F1 08ED                           <1>       or    ch, ch
0000B2F3 7444                           <1>       jz    short loc_cdir_locate_file_stc_retn
                                        <1>
                                        <1> loc_locatefile_check_next_entryblock:
0000B2F5 8A3D[868A0100]                 <1>       mov   bh, [Current_Drv]
0000B2FB 28DB                           <1>       sub   bl, bl
0000B2FD 0FB7F3                         <1>       movzx esi, bx
0000B300 81C600010900                   <1>         add     esi, Logical_DOSDisks
                                        <1>
0000B306 803D[848A0100]00               <1>       cmp   byte [Current_Dir_Level], 0
0000B30D 760A                           <1>       jna   short loc_locatefile_check_FAT_type
                                        <1>
0000B30F 803D[858A0100]01               <1>       cmp   byte [Current_FATType], 1
0000B316 730A                           <1>       jnb   short loc_locatefile_load_subdir_cluster
0000B318 C3                             <1>       retn
                                        <1>
                                        <1> loc_locatefile_check_FAT_type:
0000B319 803D[858A0100]03               <1>       cmp   byte [Current_FATType], 3
0000B320 7218                           <1>       jb    short loc_cdir_locate_file_retn
                                        <1>
                                        <1> loc_locatefile_load_subdir_cluster:
0000B322 A1[B1910100]                   <1>       mov   eax, [DirBuff_Cluster]
0000B327 E83A1A0000                     <1>       call  get_next_cluster
0000B32C 730D                           <1>       jnc   short loc_locatefile_next_cluster
0000B32E 09C0                           <1>       or    eax, eax
0000B330 7507                           <1>       jnz   short loc_locatefile_drive_not_ready_read_err
0000B332 F9                             <1>       stc
                                        <1> loc_locatefile_file_notfound:
0000B333 B802000000                     <1>       mov   eax, 2 ; File/Directory/VolName not found
0000B338 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:
0000B339 F5                 <1>       cmc ;stc
                            <1> loc_cdir_locate_file_retn:
0000B33A C3                 <1>       retn
                            <1>
                            <1> loc_locatefile_next_cluster:
0000B33B E80C1C0000         <1>       call   load_FAT_sub_directory
                            <1>       ;jc    short loc_locatefile_drive_not_ready_read_err
0000B340 72F8               <1>       jc     short loc_cdir_locate_file_retn
                            <1>
                            <1> loc_locatefile_search_again:
0000B342 8B35[6B920100]     <1>       mov    esi, [CDLF_FNAddress]
0000B348 66A1[69920100]     <1>       mov    ax, [CDLF_AttributesMask]
0000B34E 668B0D[6F920100]   <1>       mov    cx, [CDLF_DEType]
0000B355 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]
0000B357 A0[858A0100]       <1>       mov    al, [Current_FATType]
0000B35C 3C02               <1>       cmp    al, 2
0000B35E 7729               <1>       ja     short loc_reload_FAT_sub_directory
0000B360 8A25[848A0100]     <1>       mov    ah, [Current_Dir_Level]
0000B366 08C0               <1>       or     al, al
0000B368 740A               <1>       jz     short loc_reload_FS_directory
0000B36A 08E4               <1>       or     ah, ah
0000B36C 751B               <1>       jnz    short loc_reload_FAT_sub_directory
                            <1> loc_reload_FAT_12_16_root_directory:
0000B36E E84E1B0000         <1>       call   load_FAT_root_directory
0000B373 C3                 <1>       retn
                            <1> loc_reload_FS_directory:
0000B374 20E4               <1>       and    ah, ah
0000B376 7506               <1>       jnz    short loc_reload_FS_sub_directory
                            <1> loc_reload_FS_root_directory:
0000B378 E80B1C0000         <1>       call   load_FS_root_directory
0000B37D C3                 <1>       retn
                            <1> loc_reload_FS_sub_directory:
0000B37E A1[808A0100]       <1>       mov    eax, [Current_Dir_FCluster]
0000B383 E8011C0000         <1>       call   load_FS_sub_directory
0000B388 C3                 <1>       retn
                            <1> loc_reload_FAT_sub_directory:
0000B389 A1[808A0100]       <1>       mov    eax, [Current_Dir_FCluster]
0000B38E E8B91B0000         <1>       call   load_FAT_sub_directory
0000B393 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>
0000B394 663B1D[AF910100]   <1>       cmp    bx, [DirBuff_LastEntry]
0000B39B 0F8739010000       <1>         ja      loc_ffde_stc_retn_255
                            <1>
                            <1>       ;mov    [DirBuff_CurrentEntry], bx
                            <1>
0000B3A1 BF00000800         <1>       mov    edi, Directory_Buffer
0000B3A6 66A3[7C920100]     <1>       mov    [FDE_AttrMask], ax
                            <1>
0000B3AC 29C0               <1>       sub    eax, eax
```

```
                                <1>
                                <1>       ;;mov  [PreviousAttr], al ; 0 ;; 13/02/2016
0000B3AE 66A3[7E920100]         <1>       mov    [AmbiguousFileName], ax ; 0
                                <1>
0000B3B4 6689D8                 <1>       mov    ax, bx
0000B3B7 66C1E005               <1>       shl    ax, 5 ; ; * 32 ; Directory entry size
0000B3BB 01C7                   <1>       add     edi, eax
                                <1>
0000B3BD 08ED                   <1>       or     ch, ch
0000B3BF 0F852C010000           <1>         jnz     loc_find_free_deleted_entry_0
                                <1>
0000B3C5 08C9                   <1>       or      cl, cl
0000B3C7 0F850D010000           <1>         jnz     loc_ffde_stc_retn_255
                                <1>
                                <1> check_find_dir_entry:
0000B3CD 66A1[7C920100]         <1>       mov    ax, [FDE_AttrMask]
0000B3D3 8A2F                   <1>       mov    ch, [edi]
0000B3D5 80FD00                 <1>       cmp     ch, 0 ; Is it never used entry?
0000B3D8 0F86FF000000           <1>       jna    loc_find_direntry_stc_retn
0000B3DE 56                     <1>       push   esi
0000B3DF 8A570B                 <1>       mov    dl, [edi+0Bh] ; File attributes
0000B3E2 80FDE5                 <1>       cmp    ch, 0E5h ; Is it a deleted file?
0000B3E5 746D                   <1>       je     short loc_find_dir_next_entry_prevdeleted
                                <1>
0000B3E7 80FA0F                 <1>       cmp     dl, 0Fh ; longname sub component check
0000B3EA 7505                   <1>       jne     short loc_check_attributes_mask
0000B3EC E8ED010000             <1>       call   save_longname_sub_component
                                <1>
                                <1> loc_check_attributes_mask:
0000B3F1 88C6                   <1>       mov    dh, al
0000B3F3 20D6                   <1>       and    dh, dl
0000B3F5 38F0                   <1>       cmp    al, dh
0000B3F7 0F85BA000000           <1>         jne     loc_find_dir_next_entry
0000B3FD 20D4                   <1>       and    ah, dl
0000B3FF 0F85B2000000           <1>         jnz     loc_find_dir_next_entry
0000B405 80FA0F                 <1>       cmp    dl, 0Fh
0000B408 751A                   <1>       jne    short pass_direntry_attr_check
                                <1>
0000B40A 3C0F                   <1>       cmp    al, 0Fh ; AL = 0Fh -> find long name
0000B40C 0F85A5000000           <1>         jne     loc_find_dir_next_entry
                                <1>
0000B412 5E                     <1>       pop    esi
0000B413 6631C0                 <1>       xor    ax, ax
0000B416 8A35[80920100]         <1>       mov    dh, [PreviousAttr]
0000B41C 66891D[AD910100]       <1>       mov    [DirBuff_CurrentEntry], bx
0000B423 C3                     <1>       retn
                                <1>
                                <1> pass_direntry_attr_check:
0000B424 89FD                   <1>       mov    ebp, edi ; 14/02/2016
0000B426 B908000000             <1>       mov    ecx, 8
                                <1> loc_lodsb_find_dir:
0000B42B AC                     <1>       lodsb
0000B42C 3C2A                   <1>       cmp    al, '*'
0000B42E 7508                   <1>       jne    short pass_fde_ambiguous1_check
0000B430 FE05[7F920100]         <1>         inc     byte [AmbiguousFileName+1]
0000B436 EB28                   <1>       jmp    short loc_check_direntry_extension
                                <1>
                                <1> pass_fde_ambiguous1_check:
0000B438 3C3F                   <1>       cmp    al, '?'
0000B43A 750D                   <1>       jne    short pass_fde_ambiguous2_check
0000B43C FE05[7E920100]         <1>       inc    byte [AmbiguousFileName]
0000B442 803F20                 <1>       cmp    byte [edi], 20h
0000B445 764E                   <1>       jna    short loc_find_dir_next_entry_ebp
0000B447 EB14                   <1>       jmp    short loc_scasb_find_dir_inc_di
                                <1>
                                <1> pass_fde_ambiguous2_check:
0000B449 3C20                   <1>       cmp    al, 20h
0000B44B 750C                   <1>       jne    short loc_scasb_find_dir
0000B44D 803F20                 <1>       cmp    byte [edi], 20h
0000B450 7543                   <1>       jne    short loc_find_dir_next_entry_ebp
0000B452 EB0C                   <1>       jmp    short loc_check_direntry_extension
                                <1>
                                <1> loc_find_dir_next_entry_prevdeleted:
0000B454 80CA80                 <1>       or     dl, 80h  ; Bit 7 -> deleted entry sign
0000B457 EB5E                   <1>       jmp    short loc_find_dir_next_entry
                                <1>
                                <1> loc_scasb_find_dir:
0000B459 3A07                   <1>       cmp    al, [edi]
0000B45B 7538                   <1>       jne    short loc_find_dir_next_entry_ebp
                                <1> loc_scasb_find_dir_inc_di:
0000B45D 47                     <1>       inc    edi
0000B45E E2CB                   <1>       loop   loc_lodsb_find_dir
                                <1>
                                <1> loc_check_direntry_extension:
0000B460 BE08000000             <1>       mov    esi, 8
0000B465 89F7                   <1>       mov    edi, esi ; 8
0000B467 033424                 <1>       add    esi, [esp] ; Sub Dir or File Name Address
0000B46A 01EF                   <1>       add    edi, ebp
0000B46C B103                   <1>       mov    cl, 3
                                <1> loc_lodsb_find_dir_ext:
0000B46E AC                     <1>       lodsb
0000B46F 3C2A                   <1>       cmp    al, '*'
0000B471 7508                   <1>       jne    short pass_fde_ambiguous3_check
0000B473 FE05[7F920100]         <1>       inc    byte [AmbiguousFileName+1]
0000B479 EB1E                   <1>       jmp    short loc_find_dir_proper_direntry
                                <1>
                                <1> pass_fde_ambiguous3_check:
0000B47B 3C3F                   <1>       cmp    al, '?'
0000B47D 750D                   <1>       jne    short pass_fde_ambiguous4_check
0000B47F FE05[7E920100]         <1>       inc    byte [AmbiguousFileName]
0000B485 803F20                 <1>       cmp    byte [edi], 20h
0000B488 760B                   <1>       jna    short loc_find_dir_next_entry_ebp
0000B48A EB49                   <1>       jmp    short loc_scasb_find_dir_ext_inc_di
                                <1>
                                <1> pass_fde_ambiguous4_check:
```

```
0000B48C 3C20                <1>       cmp    al, 20h
0000B48E 7541                <1>       jne    short loc_scasb_find_dir_ext
0000B490 803F20              <1>       cmp    byte [edi], 20h
0000B493 7404                <1>       je     short loc_find_dir_proper_direntry
                             <1>
                             <1> loc_find_dir_next_entry_ebp:
0000B495 89EF                <1>       mov    edi, ebp ; 14/02/2016
0000B497 EB1E                <1>       jmp    short loc_find_dir_next_entry
                             <1>
                             <1> loc_find_dir_proper_direntry:
0000B499 30C9                <1>       xor    cl, cl
                             <1> loc_find_dir_proper_direntry_1:
0000B49B 5E                  <1>       pop    esi
0000B49C 89EF                <1>         mov     edi, ebp
0000B49E 8A2F                <1>       mov    ch, [edi]
0000B4A0 8A570B              <1>       mov     dl, [edi+0Bh] ; Dir entry attributes
0000B4A3 66A1[7E920100]      <1>       mov    ax, [AmbiguousFileName]
                             <1> loc_find_dir_proper_direntry_2:
0000B4A9 8A35[80920100]      <1>       mov     dh, [PreviousAttr]
0000B4AF 66891D[AD910100]    <1>       mov    [DirBuff_CurrentEntry], bx
0000B4B6 C3                  <1>       retn
                             <1>
                             <1> loc_find_dir_next_entry:
0000B4B7 8815[80920100]      <1>       mov    byte [PreviousAttr], dl ; LongName check
                             <1> loc_find_dir_next_entry_1:
0000B4BD 5E                  <1>       pop    esi
0000B4BE 83C720              <1>       add    edi, 32
                             <1>       ;inc   word [DirBuff_EntryCounter]
0000B4C1 6643                <1>       inc    bx
0000B4C3 663B1D[AF910100]    <1>       cmp    bx, [DirBuff_LastEntry]
0000B4CA 770E                <1>       ja     short loc_ffde_stc_retn_255
0000B4CC E9FCFEFFFF          <1>         jmp     check_find_dir_entry
                             <1>
                             <1> loc_scasb_find_dir_ext:
0000B4D1 3A07                <1>       cmp    al, [edi]
0000B4D3 75C0                <1>       jne    short loc_find_dir_next_entry_ebp
                             <1> loc_scasb_find_dir_ext_inc_di:
0000B4D5 47                  <1>       inc    edi
0000B4D6 E296                <1>       loop    loc_lodsb_find_dir_ext
0000B4D8 EBC1                <1>       jmp    short loc_find_dir_proper_direntry_1
                             <1>
                             <1> loc_ffde_stc_retn_255:
                             <1>       ;mov   cx, 0FFFFh
0000B4DA 31C9                <1>       xor    ecx, ecx
0000B4DC 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
0000B4DD B802000000          <1>       mov    eax, 2 ; File Not Found
0000B4E2 8A35[80920100]      <1>       mov    dh, [PreviousAttr]
0000B4E8 66891D[AD910100]    <1>       mov    [DirBuff_CurrentEntry], bx
0000B4EF F9                  <1>       stc
0000B4F0 C3                  <1>       retn
                             <1>
                             <1> loc_find_free_deleted_entry_0:
0000B4F1 66A1[7C920100]      <1>       mov    ax, [FDE_AttrMask]
0000B4F7 8A2F                <1>       mov    ch, [edi]
0000B4F9 8A570B              <1>       mov    dl, [edi+0Bh] ; File attributes
0000B4FC 08C9                <1>       or     cl, cl
0000B4FE 7407                <1>       jz     short loc_check_ffde_0_repeat
                             <1>       ;cmp   cl, 0E5h
                             <1>       ;je    short pass_loc_check_ffde_0_err
0000B500 80F9FF              <1>       cmp    cl, 0FFh
0000B503 7432                <1>       je     short loc_find_free_deleted_entry_1
0000B505 EB4D                <1>       jmp    short pass_loc_check_ffde_0_err
                             <1>
                             <1> loc_check_ffde_0_repeat:
0000B507 08ED                <1>       or     ch, ch
0000B509 7511                <1>       jnz    short loc_check_ffde_0_next
                             <1>
                             <1> loc_check_ffde_retn_2:
0000B50B 6629C0              <1>       sub    ax, ax
0000B50E 8A35[80920100]      <1>       mov    dh, [PreviousAttr]
0000B514 66891D[AD910100]    <1>       mov    [DirBuff_CurrentEntry], bx
0000B51B C3                  <1>       retn
                             <1>
                             <1> loc_check_ffde_0_next:
0000B51C 6643                <1>       inc    bx
0000B51E 83C720              <1>       add    edi, 32
                             <1>       ;inc   word [DirBuff_EntryCounter]
                             <1>
0000B521 663B1D[AF910100]    <1>         cmp  bx, [DirBuff_LastEntry]
0000B528 77B0                <1>       ja     short loc_ffde_stc_retn_255
0000B52A 8815[80920100]      <1>       mov    [PreviousAttr], dl
0000B530 8A2F                <1>       mov    ch, [edi]
0000B532 8A570B              <1>       mov    dl, [edi+0Bh] ; file attributes
0000B535 EBD0                <1>       jmp    short loc_check_ffde_0_repeat
                             <1>
                             <1> loc_find_free_deleted_entry_1:
0000B537 28D2                <1>       sub    dl, dl
                             <1> loc_find_free_deleted_entry_2:
0000B539 20ED                <1>       and    ch, ch
0000B53B 74CE                <1>       jz     short loc_check_ffde_retn_2
0000B53D 80FDE5              <1>       cmp    ch, 0E5h
0000B540 74C9                <1>       je     short loc_check_ffde_retn_2
0000B542 6643                <1>       inc    bx
0000B544 83C720              <1>       add    edi, 32
0000B547 663B1D[AF910100]    <1>       cmp    bx, [DirBuff_LastEntry]
0000B54E 778A                <1>       ja     short loc_ffde_stc_retn_255
0000B550 8A2F                <1>       mov    ch, [edi]
0000B552 EBE5                <1>       jmp    short loc_find_free_deleted_entry_2
                             <1>
                             <1> pass_loc_check_ffde_0_err:
0000B554 38CD                <1>       cmp    ch, cl
```

```
 973 0000B556 741F                <1>       je     short loc_check_ffde_attrib
 974                              <1>
 975 0000B558 6643                <1>       inc    bx
 976 0000B55A 83C720              <1>       add    edi, 32
 977 0000B55D 663B1D[AF910100]    <1>       cmp    bx, [DirBuff_LastEntry]
 978 0000B564 0F8770FFFFFF        <1>         ja       loc_ffde_stc_retn_255
 979 0000B56A 8815[80920100]      <1>       mov    [PreviousAttr], dl
 980 0000B570 8A2F                <1>       mov    ch, [edi]
 981 0000B572 8A570B              <1>       mov    dl, [edi+0Bh]
 982 0000B575 EBDD                <1>       jmp    short pass_loc_check_ffde_0_err
 983                              <1>
 984                              <1> loc_check_ffde_attrib:
 985 0000B577 88C6                <1>       mov    dh, al
 986 0000B579 20D6                <1>       and    dh, dl
 987 0000B57B 38F0                <1>       cmp    al, dh
 988 0000B57D 759D                <1>       jne    short loc_check_ffde_0_next
 989 0000B57F 20D4                <1>       and    ah, dl
 990 0000B581 7599                <1>       jnz    short loc_check_ffde_0_next
 991 0000B583 30C9                <1>       xor    cl, cl
 992 0000B585 EB84                <1>         jmp      loc_check_ffde_retn_2
 993                              <1>
 994                              <1> convert_file_name:
 995                              <1>       ; 06/03/2016
 996                              <1>       ; 11/02/2016
 997                              <1>       ; 07/02/2016 (TRDOS 386 = TRDOS v2.0)
 998                              <1>       ; 06/10/2009
 999                              <1>       ; 2005
1000                              <1>       ;
1001                              <1>       ; INPUT  ->
1002                              <1>       ;      ESI = Dot File Name Location
1003                              <1>       ;      EDI = Dir Entry Format File Name Location
1004                              <1>       ; OUTPUT ->
1005                              <1>       ;      EDI = Dir Entry Format File Name Location
1006                              <1>       ;      ESI = Dot File Name Location (capitalized)
1007                              <1>       ;
1008                              <1>       ; (ECX, AL will be changed)
1009                              <1>
1010 0000B587 56                 <1>       push   esi
1011 0000B588 57                 <1>       push   edi
1012                              <1>
1013 0000B589 B90B000000         <1>       mov    ecx, 11
1014 0000B58E B020               <1>       mov    al, 20h
1015 0000B590 F3AA               <1>       rep    stosb
1016                              <1>
1017 0000B592 8B3C24             <1>       mov    edi, [esp]
1018                              <1>
1019 0000B595 B10C               <1>       mov    cl, 12 ; file name length (max.)
1020                              <1>       ; 06/03/2016
1021                              <1>       ; Directory entry name limit (11 bytes) check for
1022                              <1>       ; 'rename_directory_entry' procedure.
1023                              <1>       ; (EDI points to Directory Entry)
1024                              <1>       ; (If the file name would not contain a dot
1025                              <1>       ; and file name length would be 12, this would cause to
1026                              <1>       ; overwrite the attributes byte of the directory entry.)
1027                              <1>       ;
1028 0000B597 B50B               <1>       mov    ch, 11 ; directory entry's name length
1029                              <1> loc_check_first_dot:
1030 0000B599 8A06               <1>       mov    al, [esi]
1031 0000B59B 3C2E               <1>       cmp    al, 2Eh
1032 0000B59D 750C               <1>       jne    short pass_check_first_dot
1033 0000B59F 8807               <1>       mov    [edi], al
1034 0000B5A1 47                 <1>       inc    edi
1035 0000B5A2 46                 <1>       inc    esi
1036 0000B5A3 FEC9               <1>       dec    cl
1037 0000B5A5 75F2               <1>       jnz    short loc_check_first_dot
1038                              <1>       ;;(ecx <= 12)
1039                              <1>       ;;loop loc_check_first_dot
1040 0000B5A7 EB30               <1>       jmp    short stop_convert_file
1041                              <1>
1042                              <1> loc_get_fchar:
1043 0000B5A9 8A06               <1>       mov    al, [esi]
1044                              <1> pass_check_first_dot:
1045 0000B5AB 3C61               <1>       cmp    al, 61h ; 'a'
1046 0000B5AD 7208               <1>       jb     short pass_name_capitalize
1047 0000B5AF 3C7A               <1>       cmp    al, 7Ah ; 'z'
1048 0000B5B1 7704               <1>       ja     short pass_name_capitalize
1049 0000B5B3 24DF               <1>       and    al, 0DFh
1050 0000B5B5 8806               <1>       mov    [esi], al
1051                              <1> pass_name_capitalize:
1052 0000B5B7 3C21               <1>       cmp    al, 21h
1053 0000B5B9 721E               <1>       jb     short stop_convert_file
1054 0000B5BB 3C2E               <1>       cmp    al, 2Eh ; '.'
1055 0000B5BD 750C               <1>       jne    short pass_dot_space
1056                              <1> add_dot_space:
1057 0000B5BF 80F904             <1>       cmp    cl, 4
1058 0000B5C2 760E               <1>       jna    short inc_and_loop
1059 0000B5C4 47                 <1>       inc    edi
1060 0000B5C5 FECD               <1>       dec    ch ; 06/03/2016
1061 0000B5C7 FEC9               <1>       dec    cl
1062 0000B5C9 EBF4               <1>       jmp    short add_dot_space
1063                              <1>
1064                              <1>       ;mov   al, 4
1065                              <1>       ;cmp   cl, al
1066                              <1>       ;jna   short inc_and_loop
1067                              <1>       ;sub   cl, al
1068                              <1>       ;add   edi, ecx
1069                              <1>       ;mov   cl, al
1070                              <1>       ;jmp   short inc_and_loop
1071                              <1>
1072                              <1> pass_dot_space:
1073 0000B5CB 8807               <1>       mov    [edi], al
1074                              <1> loc_after_double_dot:
1075                              <1>       ; 06/03/2016
1076 0000B5CD FECD               <1>       dec    ch ; count down for 11 bytes dir entry limit
1077 0000B5CF 740A               <1>       jz     short stop_convert_file_x
```

```
1078 0000B5D1 47                <1>       inc    edi
1079                            <1> inc_and_loop:
1080 0000B5D2 FEC9              <1>       dec    cl ; count down for 12 bytes filename limit
1081 0000B5D4 7403              <1>       jz     short stop_convert_file
1082 0000B5D6 46                <1>       inc    esi
1083                            <1>       ;;(ecx <= 12)
1084                            <1>       ;;loop loc_get_fchar
1085 0000B5D7 EBD0              <1>       jmp    short loc_get_fchar
1086                            <1>
1087                            <1> stop_convert_file:
1088                            <1>       ; 06/03/2016
1089 0000B5D9 30ED              <1>       xor    ch, ch
1090                            <1>       ; ECX < 256 ; 'find_first_file' -> xor cl, cl
1091                            <1> stop_convert_file_x:
1092 0000B5DB 5F                <1>       pop    edi
1093 0000B5DC 5E                <1>       pop    esi
1094 0000B5DD C3                <1>       retn
1095                            <1>
1096                            <1> save_longname_sub_component:
1097                            <1>       ; 13/02/2016
1098                            <1>       ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)
1099                            <1>       ; 28/02/2010
1100                            <1>       ; 17/10/2009
1101                            <1>       ; INPUT ->
1102                            <1>       ;      EDI = Directory Entry
1103                            <1>       ;      // This procedure is called
1104                            <1>       ;      // from 'find_directory_entry' procedure.
1105                            <1>       ;      // If the last entry returns with
1106                            <1>       ;      // a non-zero LongnameFound value and
1107                            <1>       ;      // if LFN_CheckSum value is equal to
1108                            <1>       ;      // the next shortname checksum,
1109                            <1>       ;      // long name is valid.
1110                            <1>       ;      // If a longname is longer than 65 bytes,
1111                            <1>       ;      // it is invalid for trdos. (>45h)
1112                            <1>
1113 0000B5DE 57                <1>       push   edi
1114 0000B5DF 56                <1>       push   esi
1115                            <1>       ;push  ebx
1116                            <1>       ;push  ecx
1117                            <1>       ;push  edx
1118 0000B5E0 50                <1>       push   eax
1119                            <1>
1120 0000B5E1 29C9              <1>       sub    ecx, ecx
1121                            <1>       ;sub   eax, eax
1122 0000B5E3 B11A              <1>       mov    cl, 26
1123                            <1>
1124 0000B5E5 0FB607            <1>       movzx  eax, byte [edi] ; LDIR_Order
1125 0000B5E8 3C41              <1>       cmp    al, 41h  ; 40h (last long entry sign) + 1
1126 0000B5EA 722B              <1>       jb     short pass_pslnsc_last_long_entry
1127                            <1>
1128 0000B5EC 88C4              <1>       mov    ah, al
1129 0000B5EE 80EC40            <1>       sub    ah, 40h
1130 0000B5F1 8825[82920100]    <1>       mov    [LFN_EntryLength], ah
1131                            <1>
1132 0000B5F7 3C45              <1>       cmp    al, 45h  ; 40h (last long entry sign) + 5
1133                            <1>              ; Max 130 byte length is usable in TRDOS
1134                            <1> ; 26*5 = 130
1135 0000B5F9 7753              <1>       ja     short loc_pslnsc_retn
1136                            <1>
1137 0000B5FB 2407              <1>       and    al, 07h ; 0Fh
1138 0000B5FD A2[81920100]      <1>       mov    [LongNameFound], al
1139                            <1>
1140 0000B602 FEC8              <1>       dec    al
1141                            <1>       ;mov   cl, 26
1142 0000B604 F6E1              <1>       mul    cl
1143                            <1>
1144 0000B606 89C6              <1>       mov    esi, eax
1145 0000B608 01CE              <1>       add    esi, ecx
1146                            <1>              ; to make is an ASCIIZ string
1147                            <1>              ; with ax+26 bytes length
1148 0000B60A 81C6[84920100]    <1>       add    esi, LongFileName
1149 0000B610 66C7060000        <1>       mov    word [esi], 0
1150 0000B615 EB16              <1>       jmp    short loc_pslsc_move_ldir_name2
1151                            <1>
1152                            <1> pass_pslnsc_last_long_entry:
1153 0000B617 3C04              <1>       cmp    al, 04h
1154 0000B619 7733              <1>       ja     short loc_pslnsc_retn
1155 0000B61B FE0D[81920100]    <1>       dec    byte [LongNameFound]
1156 0000B621 3A05[81920100]    <1>       cmp    al, [LongNameFound]
1157 0000B627 7525              <1>       jne    short loc_pslnsc_retn
1158                            <1>
1159                            <1> loc_pslsc_move_ldir_name1:
1160 0000B629 FEC8              <1>       dec    al
1161                            <1>       ;mov   cl, 26
1162 0000B62B F6E1              <1>       mul    cl
1163                            <1>
1164                            <1> loc_pslsc_move_ldir_name2:
1165 0000B62D 8A4F0D            <1>       mov    cl, [edi+0Dh] ; long name checksum
1166 0000B630 880D[83920100]    <1>       mov    [LFN_CheckSum], cl
1167 0000B636 89FE              <1>       mov    esi, edi ; LDIR_Order
1168 0000B638 BF[84920100]      <1>       mov    edi, LongFileName
1169 0000B63D 01C7              <1>       add    edi, eax
1170 0000B63F 46                <1>       inc    esi
1171 0000B640 B105              <1>       mov    cl, 5 ; chars 1 to 5
1172 0000B642 F366A5            <1>       rep    movsw
1173 0000B645 83C603            <1>       add    esi, 3
1174 0000B648 A5                <1>       movsd  ; char 6 & 7
1175 0000B649 A5                <1>       movsd  ; char 8 & 9
1176 0000B64A A5                <1>       movsd  ; char 10 & 11
1177 0000B64B 46                <1>       inc    esi
1178 0000B64C 46                <1>       inc    esi
1179 0000B64D A5                <1>       movsd   ; char 12 & 13
1180                            <1>
1181                            <1> loc_pslnsc_retn:
1182 0000B64E 58                <1>       pop    eax
```

```
1183                            <1>       ;pop   edx
1184                            <1>       ;pop   ecx
1185                            <1>       ;pop   ebx
1186 0000B64F 5E                <1>       pop    esi
1187 0000B650 5F                <1>       pop    edi
1188                            <1>
1189 0000B651 C3                <1>       retn
1190                            <1>
1191                            <1> parse_path_name:
1192                            <1>       ; 10/02/2016
1193                            <1>       ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)
1194                            <1>       ; 10/009/2011 ('proc_parse_pathname')
1195                            <1>       ; 27/11/2009
1196                            <1>       ; 05/12/2004
1197                            <1>       ;
1198                            <1>       ; INPUT ->
1199                            <1>       ;     ESI = Beginning of ASCIIZ pathname string
1200                            <1>       ;       EDI = Destination Address
1201                            <1>       ;            (which is TR-DOS FindFile data buffer)
1202                            <1>       ; OUTPUT ->
1203                            <1>       ;     CF = 1 -> Error
1204                            <1>       ;          EAX = Error Code (AL)
1205                            <1>       ;
1206                            <1>       ; (Modified registers: eax, ecx, esi, edi)
1207                            <1>
1208                            <1>       ; Clear the pathname bytes in TR-DOS Findfile data buffer
1209 0000B652 57                <1>       push   edi
1210 0000B653 B914000000        <1>       mov    ecx, 20  ; 80 bytes
1211 0000B658 31C0              <1>       xor    eax, eax
1212 0000B65A F3AB              <1>       rep    stosd
1213 0000B65C 5F                <1>       pop    edi
1214                            <1>
1215 0000B65D 668B06            <1>       mov    ax, [esi]
1216 0000B660 80FC3A            <1>       cmp    ah, ':'
1217 0000B663 741C              <1>       je     short loc_ppn_change_drive
1218 0000B665 A0[868A0100]      <1>       mov    al, [Current_Drv]
1219 0000B66A EB33              <1>       jmp    short pass_ppn_change_drive
1220                            <1>
1221                            <1> pass_ppn_cdir:
1222 0000B66C 8B35[A6930100]    <1>       mov    esi, [First_Path_Pos]
1223 0000B672 AC                <1>       lodsb
1224                            <1> loc_ppn_get_filename:
1225 0000B673 83C741            <1>       add    edi, 65 ; FindFile_Name location
1226                            <1>       ; TRDOS Filename length must not be more than 12 bytes
1227                            <1>       ;mov   ecx, 12
1228 0000B676 B10C              <1>       mov    cl, 12
1229                            <1> loc_ppn_get_fnchar_next:
1230 0000B678 AA                <1>       stosb
1231 0000B679 AC                <1>       lodsb
1232 0000B67A 3C21              <1>       cmp    al, 21h
1233 0000B67C 7274              <1>       jb     short loc_ppn_clc_return
1234 0000B67E E2F8              <1>         loop    loc_ppn_get_fnchar_next
1235                            <1> loc_ppn_return:
1236 0000B680 C3                <1>       retn
1237                            <1>
1238                            <1> loc_ppn_change_drive:
1239 0000B681 24DF              <1>       and    al, 0DFh
1240 0000B683 2C41              <1>       sub    al, 'A'; A:
1241 0000B685 726F              <1>       jc     short loc_ppn_invalid_drive
1242 0000B687 3805[50400100]    <1>       cmp    [Last_DOS_DiskNo], al
1243 0000B68D 7267              <1>       jb     short loc_ppn_invalid_drive
1244                            <1>
1245 0000B68F 46                <1>       inc    esi
1246 0000B690 46                <1>       inc    esi
1247 0000B691 8A26              <1>       mov    ah, [esi]
1248 0000B693 80FC21            <1>       cmp    ah, 21h
1249 0000B696 7307              <1>       jnb    short pass_ppn_change_drive
1250                            <1>
1251                            <1> loc_ppn_cmd_failed:
1252                            <1>       ; File or directory name is not existing
1253 0000B698 8807              <1>       mov    [edi], al ; Drv
1254 0000B69A 66B80100          <1>       mov    ax, 1 ; eax = 1
1255                            <1>       ; TR-DOS Error Code 01h = Bad Command Argument
1256                            <1>       ; MS-DOS Error Code 01h : Invalid Function Number
1257                            <1>       ;stc
1258                            <1>       ; (MainProg ErrMsg: "Bad command or file name!")
1259 0000B69E C3                <1>       retn
1260                            <1>
1261                            <1> pass_ppn_change_drive:
1262 0000B69F 8935[A6930100]    <1>       mov    [First_Path_Pos], esi
1263 0000B6A5 C705[AA930100]0000- <1>     mov    dword [Last_Slash_Pos], 0
1263 0000B6AD 0000              <1>
1264 0000B6AF AA                <1>       stosb
1265 0000B6B0 8A06              <1>       mov    al, [esi]
1266                            <1> loc_scan_ppn_dslash:
1267 0000B6B2 3C2F              <1>       cmp    al, '/'
1268 0000B6B4 7506              <1>       jne    short loc_scan_next_slash_pos
1269 0000B6B6 8935[AA930100]    <1>       mov    [Last_Slash_Pos], esi
1270                            <1> loc_scan_next_slash_pos:
1271 0000B6BC 46                <1>       inc    esi
1272 0000B6BD 8A06              <1>       mov    al, [esi]
1273 0000B6BF 3C20              <1>       cmp    al, 20h
1274 0000B6C1 77EF              <1>       ja     short loc_scan_ppn_dslash
1275 0000B6C3 833D[AA930100]00  <1>       cmp    dword [Last_Slash_Pos], 0
1276 0000B6CA 76A0              <1>       jna    short pass_ppn_cdir
1277                            <1>
1278 0000B6CC 8B0D[AA930100]    <1>       mov    ecx, [Last_Slash_Pos]
1279 0000B6D2 8B35[A6930100]    <1>       mov    esi, [First_Path_Pos]
1280 0000B6D8 29F1              <1>       sub    ecx, esi
1281 0000B6DA 41                <1>       inc    ecx
1282                            <1>       ;cmp   ecx, 64
1283 0000B6DB 80F940            <1>       cmp    cl, 64
1284 0000B6DE 7715              <1>       ja     short loc_ppn_invalid_drive_stc
1285                            <1>
1286 0000B6E0 89F8              <1>       mov    eax, edi ; Dest Dir String Location (65 byte)
```

```
1287 0000B6E2 F3A4               <1>       rep    movsb
1288                             <1>       ;mov   [edi], cl ; 0, End of Dir String
1289 0000B6E4 8B35[AA930100]     <1>       mov    esi, [Last_Slash_Pos]
1290 0000B6EA 46                 <1>       inc    esi
1291 0000B6EB 89C7               <1>       mov    edi, eax
1292 0000B6ED AC                 <1>       lodsb
1293 0000B6EE 3C21               <1>       cmp    al, 21h
1294 0000B6F0 7381               <1>       jnb    short loc_ppn_get_filename
1295                             <1> loc_ppn_clc_return:
1296                             <1>       ;clc
1297 0000B6F2 31C0               <1>       xor    eax, eax
1298 0000B6F4 C3                 <1>       retn
1299                             <1>
1300                             <1> loc_ppn_invalid_drive_stc:
1301 0000B6F5 F5                 <1>       cmc     ; stc
1302                             <1> loc_ppn_invalid_drive:
1303                             <1>       ; cf = 1
1304                             <1>       ; The Drive Letter/Char < "A" or > "Z"
1305 0000B6F6 66B80F00           <1>       mov    ax, 0Fh
1306                             <1>       ; MS-DOS Error Code 0Fh = Disk Drive Invalid
1307                             <1>       ; (MainProg ErrMsg: "Drive not ready or read error!")
1308 0000B6FA C3                 <1>       retn
1309                             <1>
1310                             <1> find_longname:
1311                             <1>       ; 13/02/2016 (TRDOS 386 = TRDOS v2.0)
1312                             <1>       ; 24/01/2010 (DIR.ASM, 'proc_find_longname')
1313                             <1>       ; 17/10/2009
1314                             <1>
1315                             <1>       ; INPUT ->
1316                             <1>       ;      ESI = DOS short file name address
1317                             <1>       ;      for example: "filename.ext"
1318                             <1>       ;
1319                             <1>       ; OUTPUT ->
1320                             <1>       ;      ESI = ASCIIZ longname address (cf = 0)
1321                             <1>       ;      cf = 1 -> error number returns in EAX (AL)
1322                             <1>       ;      AL = 0 & CF=1 -> longname not found
1323                             <1>       ;           the file/directory has no longname
1324                             <1>       ;      cf = 0 -> AL = FAT Type
1325                             <1>
1326                             <1>       ; 17/10/2009
1327                             <1>       ; ASCIIZ string will be returned
1328                             <1>       ; as LongFileName
1329                             <1>       ; clearing/reset is not needed
1330                             <1>       ;mov   ecx, 33
1331                             <1>       ;mov   edi, LongFileName
1332                             <1>       ;sub   ax, ax ; 0
1333                             <1>       ;rep   stosw
1334                             <1>
1335                             <1>       ;mov   byte [LongNameFound], 0
1336                             <1>
1337                             <1>       ; ESI = ASCIIZ file/directory name address
1338                             <1>       ;   AL = Attributes AND mask
1339                             <1>       ;      (Result of AND must be equal to AL)
1340                             <1>       ;   AH = Negative attributes mask
1341                             <1>       ;      (Result of AND must be ZERO)
1342 0000B6FB 66B80008           <1>       mov    ax, 0800h
1343                             <1>              ; it must not be volume name or longname
1344 0000B6FF E87DDDFFFF         <1>       call   find_first_file
1345 0000B704 7216               <1>       jc     short loc_fln_retn
1346                             <1>
1347                             <1> loc_fln_check_FAT_Type:
1348 0000B706 803D[858A0100]01   <1>       cmp    byte [Current_FATType], 1
1349 0000B70D 7306               <1>       jnb    short loc_fln_check_longname_yes_sign
1350                             <1>
1351 0000B70F E839000000         <1>       call   get_fs_longname
1352 0000B714 C3                 <1>       retn
1353                             <1>
1354                             <1> loc_fln_check_longname_yes_sign:
1355 0000B715 08FF               <1>       or     bh, bh
1356 0000B717 7504               <1>       jnz    short loc_fln_check_longnamefound_number
1357                             <1> loc_fln_longname_not_found_retn:
1358 0000B719 31C0               <1>       xor    eax, eax
1359                             <1>       ; cf = 1 & al = 0 -> longname not found
1360 0000B71B F9                 <1>       stc
1361                             <1> loc_fln_retn:
1362 0000B71C C3                 <1>       retn
1363                             <1>
1364                             <1> loc_fln_check_longnamefound_number:
1365                             <1>       ; 'LongNameFound' is set by
1366                             <1>         ; by 'save_longname_sub_component'
1367                             <1>       ; which is called from
1368                             <1>       ; 'find_directory_entry'
1369                             <1>       ; which is called from
1370                             <1>       ; 'find_first_file'
1371                             <1>       ; It must 1 if the longname is valid
1372 0000B71D 803D[81920100]01   <1>         cmp     byte [LongNameFound], 1
1373 0000B724 75F3               <1>       jne    short loc_fln_longname_not_found_retn
1374                             <1>
1375                             <1> loc_fln_calculate_checksum:
1376 0000B726 E813000000         <1>       call   calculate_checksum
1377                             <1>       ; AL = shortname checksum
1378                             <1>
1379                             <1> loc_fln_longname_validation:
1380                             <1>       ; 'LFN_CheckSum' has been set already
1381                             <1>       ; by 'save_longname_sub_component'
1382                             <1>       ; which is called from
1383                             <1>       ; 'find_directory_entry'
1384                             <1>       ; which is called from
1385                             <1>       ; 'find_first_file'
1386 0000B72B 3805[83920100]     <1>       cmp    [LFN_CheckSum], al
1387 0000B731 75E6               <1>       jne    short loc_fln_longname_not_found_retn
1388                             <1>
1389 0000B733 BE[84920100]       <1>       mov    esi, LongFileName
1390 0000B738 A0[858A0100]       <1>       mov    al, [Current_FATType]
1391 0000B73D 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>
0000B73E 30C0                   <1>       xor    al, al
0000B740 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
0000B745 D0C8                   <1>       ror    al, 1 ; 17/10/2009
0000B747 0206                   <1>       add    al, [esi]
0000B749 46                     <1>       inc    esi
                                <1>       ;add   al, ah
0000B74A E2F9                   <1>       loop   loc_next_sum
0000B74C C3                     <1>       retn
                                <1>
                                <1> get_fs_longname:
                                <1>       ; temporary (13/02/2016)
0000B74D 31C0                   <1>       xor eax, eax
0000B74F F9                     <1>       stc
0000B750 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>
0000B751 80E107                 <1>       and    cl, 07h
0000B754 880D[00940100]         <1>       mov    byte [mkdir_attrib], cl
                                <1>
0000B75A 56                     <1>       push   esi
0000B75B 31DB                   <1>       xor    ebx, ebx
0000B75D 8A3D[868A0100]         <1>       mov    bh, [Current_Drv]
0000B763 BE00010900             <1>       mov    esi, Logical_DOSDisks
0000B768 01DE                   <1>       add    esi, ebx
0000B76A 5B                     <1>       pop    ebx
                                <1>
                                <1>       ; 10/07/2010 -> 1st writable disk check for trdos
                                <1>       ; LD_DiskType = 0 for write protection (read only)
0000B76B 807E0101               <1>       cmp    byte [esi+LD_DiskType], 1 ; 0 = Invalid
0000B76F 730B                   <1>       jnb    short loc_mkdir_check_file_sytem
                                <1>       ; 16/10/2016 (13h -> 30)
0000B771 B81E000000             <1>       mov    eax, 30 ; 'Disk write-protected' error
0000B776 BA00000000             <1>       mov    edx, 0
                                <1>       ; err retn: EDX = 0, EBX = Dir name offset
                                <1>       ;ESI = Logical DOS drive description table address
0000B77B 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:
0000B77C 807E0301               <1>       cmp    byte [esi+LD_FATType], 1
0000B780 730B                   <1>       jnb    short loc_mkdir_check_free_sectors
                                <1>
                                <1> loc_make_fs_directory:
0000B782 A1[808A0100]           <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)
0000B787 E8D5150000             <1>       call   make_fs_directory
0000B78C C3                     <1>       retn
                                <1>
                                <1> loc_mkdir_check_free_sectors:
0000B78D 0FB64613               <1>         movzx   eax, byte [esi+LD_BPB+SecPerClust]
```

```
1497 0000B791 8B4E74              <1>       mov    ecx, [esi+LD_FreeSectors]
1498 0000B794 39C1                <1>       cmp    ecx, eax
1499 0000B796 7255                <1>       jb     short loc_mkdir_insufficient_disk_space
1500                              <1>
1501                              <1> loc_make_fat_directory:
1502 0000B798 891D[F0930100]      <1>       mov    [mkdir_DirName_Offset], ebx
1503 0000B79E 890D[FC930100]      <1>       mov    [mkdir_FreeSectors], ecx
1504                              <1>
1505                              <1>       ;mov   al, [esi+LD_BPB+SecPerClust]
1506 0000B7A4 A2[02940100]        <1>       mov    byte [mkdir_SecPerClust], al
1507                              <1>
1508                              <1> loc_mkdir_gffc_1:
1509 0000B7A9 E80F180000          <1>       call   get_first_free_cluster
1510 0000B7AE 722A                <1>       jc     short loc_mkdir_gffc_retn
1511                              <1>
1512                              <1> ;loc_mkdir_gffc_1_cont:
1513                              <1>       ;cmp   eax, 2
1514                              <1>       ;jb    short loc_mkdir_gffc_insufficient_disk_space
1515                              <1>
1516                              <1> ;loc_mkdir_gffc_1_save_fcluster:
1517 0000B7B0 A3[F4930100]        <1>       mov    [mkdir_FFCluster], eax
1518                              <1>
1519                              <1> loc_mkdir_locate_ffe:
1520                              <1>       ; Current directory fcluster <> Directory buffer cluster
1521                              <1>       ; Current directory will be reloaded by
1522                              <1>       ; 'locate_current_dir_file' procedure
1523                              <1>       ;
1524                              <1>       ; ESI = Logical DOS Drive Description Table Address
1525                              <1>       ;push  esi ; 27/02/2016
1526 0000B7B5 31C0                <1>       xor    eax, eax
1527 0000B7B7 89C1                <1>         mov  ecx, eax
1528 0000B7B9 6649                <1>       dec    cx ; FFFFh
1529                              <1>       ; CX = FFFFh -> find first deleted or free entry
1530                              <1>       ; ESI would be ASCIIZ filename address if the call
1531                              <1>       ; would not be for first free or deleted dir entry
1532 0000B7BB E8D0FAFFFF          <1>       call   locate_current_dir_file
1533 0000B7C0 734C                <1>       jnc    short loc_mkdir_set_ff_dir_entry_1
1534                              <1>       ;pop   esi
1535                              <1>       ; ESI = Logical DOS Drive Description Table Address
1536 0000B7C2 83F802              <1>       cmp    eax, 2  ; cmp al, 2 ; File/Dir not found !
1537 0000B7C5 752B                <1>       jne    short loc_mkdir_stc_return
1538                              <1>
1539                              <1> loc_mkdir_add_new_cluster:
1540 0000B7C7 3805[858A0100]      <1>       cmp    byte [Current_FATType], al ; 2
1541                              <1>       ;cmp   byte ptr [esi+LD_FATType], 2
1542 0000B7CD 770C                <1>       ja     short loc_mkdir_add_new_cluster_check_fsc
1543 0000B7CF 803D[848A0100]01    <1>       cmp    byte [Current_Dir_Level], 1
1544                              <1>       ;cmp   byte [esi+LD_CDirLevel], 1
1545 0000B7D6 7303                <1>       jnb    short loc_mkdir_add_new_cluster_check_fsc
1546                              <1>
1547 0000B7D8 B00C                <1>       mov    al, 12 ; No more files
1548                              <1> loc_mkdir_gffc_retn:
1549 0000B7DA C3                  <1>       retn
1550                              <1>
1551                              <1> loc_mkdir_add_new_cluster_check_fsc:
1552 0000B7DB 8B0D[FC930100]      <1>       mov    ecx, [mkdir_FreeSectors]
1553                              <1>       ;movzx eax, byte [mkdir_SecPerClust]
1554 0000B7E1 A0[02940100]        <1>       mov    al, [mkdir_SecPerClust]
1555 0000B7E6 66D1E0              <1>       shl    ax, 1 ; AX = 2 * AX
1556 0000B7E9 39C1                <1>       cmp    ecx, eax
1557 0000B7EB 7350                <1>       jnb    short loc_mkdir_add_new_subdir_cluster
1558                              <1>
1559                              <1> loc_mkdir_insufficient_disk_space:
1560                              <1>       ;mov   edx, ecx
1561                              <1> ;loc_mkdir_gffc_insufficient_disk_space:
1562 0000B7ED 66B82700              <1>       mov    ax, 27h ; MSDOS err => insufficient disk space
1563                              <1>       ; err retn: EDX = Free sectors, EBX = Dir name offset
1564                              <1>         ; ESI -> Dos drive description table address
1565                              <1>       ;; ecx = edx
1566                              <1>       ;
1567 0000B7F1 C3                  <1>       retn
1568                              <1>
1569                              <1> loc_mkdir_stc_return:
1570 0000B7F2 F9                  <1>       stc
1571 0000B7F3 C3                  <1>       retn
1572                              <1>
1573                              <1> loc_mkdir_gffc_2:
1574 0000B7F4 E8C4170000          <1>       call   get_first_free_cluster
1575 0000B7F9 72DF                <1>       jc     short loc_mkdir_gffc_retn
1576                              <1>
1577                              <1> ;loc_mkdir_gffc_1_cont:
1578                              <1>       ;cmp   eax, 2
1579                              <1>       ;jb    short loc_mkdir_gffc_insufficient_disk_space
1580                              <1>
1581                              <1> ;loc_mkdir_gffc_2_save_fcluster:
1582 0000B7FB A3[F4930100]        <1>       mov    [mkdir_FFCluster], eax
1583                              <1>
1584 0000B800 A1[F8930100]        <1>       mov    eax, [mkdir_LastDirCluster]
1585                              <1>
1586 0000B805 E842170000          <1>       call   load_FAT_sub_directory
1587 0000B80A 72CE                <1>       jc     short loc_mkdir_gffc_retn
1588                              <1>
1589 0000B80C 31FF                <1>       xor    edi, edi
1590                              <1> loc_mkdir_set_ff_dir_entry_1:
1591                              <1>       ; 27/02/2016
1592 0000B80E 56                  <1>       push   esi ; Logical DOS Drv Desc. Tbl. address
1593                              <1>       ; EDI = Directory Entry Address
1594 0000B80F 8B35[F0930100]      <1>       mov    esi, [mkdir_DirName_Offset]
1595 0000B815 A1[F4930100]        <1>       mov    eax, [mkdir_FFCluster]
1596                              <1>
1597 0000B81A 66B91000            <1>       mov    cx, 10h      ; CL = Directory attribute
1598                              <1>                   ; CH = 0 -> File size is 0
1599 0000B81E 0A0D[00940100]      <1>       or     cl, [mkdir_attrib] ; S, H, R
1600 0000B824 E8B0010000          <1>       call   make_directory_entry
1601                              <1>
```

```
1602 0000B829 5E                  <1>       pop    esi
1603                              <1>
1604 0000B82A C605[AC910100]02    <1>       mov    byte [DirBuff_ValidData], 2
1605 0000B831 E880020000          <1>       call   save_directory_buffer
1606 0000B836 0F83DA000000        <1>         jnc     loc_mkdir_set_ff_dir_entry_2
1607                              <1>
1608                              <1> loc_mkdir_return:
1609 0000B83C C3                  <1>       retn
1610                              <1>
1611                              <1> loc_mkdir_add_new_subdir_cluster:
1612 0000B83D 8B15[B1910100]      <1>       mov    edx, [DirBuff_Cluster]
1613 0000B843 8915[F8930100]      <1>       mov    [mkdir_LastDirCluster], edx
1614                              <1>
1615 0000B849 A1[F4930100]        <1>       mov    eax, [mkdir_FFCluster]
1616 0000B84E E8F9160000          <1>       call   load_FAT_sub_directory
1617 0000B853 72E7                <1>       jc     short loc_mkdir_return
1618                              <1>       ; eax = 0
1619                              <1>       ; ecx =  directory buffer sector count (<= 128)
1620                              <1>
1621                              <1> pass_mkdir_add_new_subdir_cluster:
1622 0000B855 29FF                <1>       sub    edi, edi ; 0
1623                              <1>       ;mov   al, 128 ; double word
1624                              <1>       ;mul   ecx ; ecx =  directory buffer sector count
1625                              <1>       ;mov   ecx, eax
1626                              <1>       ;shl   cx, 7 ; 128 * sector count
1627 0000B857 668B4611            <1>       mov    ax, [esi+LD_BPB+BytesPerSec] ; 512
1628 0000B85B 66C1E802            <1>       shr    ax, 2 ; 'byte count / 4' for 'stosd'
1629 0000B85F 66F7E1              <1>       mul    cx ; max = 128*(512/4) -> 16384 (stosd)
1630 0000B862 6689C1              <1>       mov    cx, ax
1631 0000B865 6629C0              <1>       sub    ax, ax ; 0
1632 0000B868 F3AB                <1>       rep    stosd ; clear directory buffer
1633                              <1>
1634 0000B86A C605[AC910100]02    <1>       mov    byte [DirBuff_ValidData], 2
1635 0000B871 E840020000          <1>       call   save_directory_buffer
1636 0000B876 72C4                <1>       jc     short loc_mkdir_return
1637                              <1>
1638                              <1> loc_mkdir_save_added_cluster:
1639 0000B878 A1[F8930100]        <1>       mov    eax, [mkdir_LastDirCluster]
1640 0000B87D 8B0D[F4930100]      <1>       mov    ecx, [mkdir_FFCluster]
1641                              <1>       ; 01/03/2016
1642 0000B883 31D2                <1>       xor    edx, edx
1643 0000B885 8915[A2910100]      <1>       mov    [FAT_ClusterCounter], edx ; 0 ; reset
1644 0000B88B E800180000          <1>       call   update_cluster
1645 0000B890 7304                <1>       jnc    short loc_mkdir_save_fat_buffer_0
1646 0000B892 09C0                <1>       or     eax, eax ; EAX = 0 -> cluster value is 0 or eocc
1647 0000B894 7518                <1>       jnz    short loc_mkdir_save_fat_buffer_stc_retn
1648                              <1>
1649                              <1> loc_mkdir_save_fat_buffer_0:
1650 0000B896 A1[F4930100]        <1>       mov    eax, [mkdir_FFCluster]
1651 0000B89B A3[F8930100]        <1>       mov    [mkdir_LastDirCluster], eax
1652                              <1>
1653 0000B8A0 31C9                <1>       xor    ecx, ecx
1654 0000B8A2 49                  <1>       dec    ecx ; FFFFFFFFh
1655                              <1>       ; ESI = Logical DOS Drive Description Table address
1656 0000B8A3 E8E8170000          <1>       call   update_cluster
1657 0000B8A8 731A                <1>       jnc    short loc_mkdir_save_fat_buffer_1
1658 0000B8AA 09C0                <1>       or     eax, eax
1659 0000B8AC 7416                <1>       jz     short loc_mkdir_save_fat_buffer_1
1660                              <1>
1661                              <1> loc_mkdir_save_fat_buffer_stc_retn:
1662                              <1>       ; 01/03/2016
1663 0000B8AE 803D[A2910100]01    <1>       cmp    byte [FAT_ClusterCounter], 1
1664 0000B8B5 720C                <1>       jb     short loc_mkdir_save_fat_buffer_retn
1665                              <1>
1666 0000B8B7 66BB00FF            <1>       mov    bx, 0FF00h ; recalculate free space (BL = 0)
1667                              <1>                         ; (BH = FFh -> Use ESI as Drv Param. Tbl.)
1668 0000B8BB 50                  <1>       push   eax
1669 0000B8BC E8211B0000          <1>       call   calculate_fat_freespace
1670 0000B8C1 58                  <1>       pop    eax
1671 0000B8C2 F9                  <1>       stc
1672                              <1> loc_mkdir_save_fat_buffer_retn:
1673 0000B8C3 C3                  <1>       retn
1674                              <1>
1675                              <1> loc_mkdir_save_fat_buffer_1:
1676                              <1>       ; byte [FAT_BuffValidData] = 2
1677 0000B8C4 E8841A0000          <1>       call   save_fat_buffer
1678 0000B8C9 72E3                <1>       jc     short loc_mkdir_save_fat_buffer_stc_retn
1679                              <1>
1680                              <1>       ; 01/03/2016
1681 0000B8CB 803D[A2910100]01    <1>       cmp    byte [FAT_ClusterCounter], 1
1682 0000B8D2 721B                <1>       jb     short loc_mkdir_save_fat_buffer_2
1683                              <1>
1684                              <1>       ; ESI = Logical DOS Drive Description Table address
1685 0000B8D4 A1[A2910100]        <1>       mov    eax, [FAT_ClusterCounter]
1686 0000B8D9 66BB01FF            <1>       mov    bx, 0FF01h ; add free clusters
1687 0000B8DD E8001B0000          <1>       call   calculate_fat_freespace
1688                              <1>
1689                              <1>       ;inc   eax ; 0FFFFFFFFh -> 0 ; recalculation is needed!
1690                              <1>       ;jnz   short loc_mkdir_save_fat_buffer_2
1691                              <1>
1692                              <1>       ; ecx > 0 -> Recalculation is needed
1693 0000B8E2 09C9                <1>       or     ecx, ecx
1694 0000B8E4 7409                <1>       jz     short loc_mkdir_save_fat_buffer_2
1695                              <1>
1696 0000B8E6 66BB00FF            <1>       mov    bx, 0FF00h ; ; recalculate free space
1697 0000B8EA E8F31A0000          <1>       call   calculate_fat_freespace
1698                              <1>
1699                              <1> loc_mkdir_save_fat_buffer_2:
1700 0000B8EF C605[03940100]01    <1>       mov    byte [mkdir_add_new_cluster], 1
1701 0000B8F6 E9C4000000          <1>       jmp    loc_mkdir_upd_parent_dir_lmdt
1702                              <1>
1703                              <1> loc_mkdir_update_sub_dir_cluster:
1704 0000B8FB A1[F4930100]        <1>       mov    eax, [mkdir_FFCluster]
1705 0000B900 29C9                <1>       sub    ecx, ecx ; 0
1706                              <1>       ; 01/03/2016
```

```
1707 0000B902 890D[A2910100]      <1>       mov    [FAT_ClusterCounter], ecx ; 0 ; Reset
1708 0000B908 49                  <1>       dec    ecx ; 0FFFFFFFFh
1709                              <1>
1710                              <1>       ; ESI = Logical DOS Drive Descisption Table address
1711 0000B909 E882170000          <1>       call   update_cluster
1712 0000B90E 7379                <1>       jnc    short loc_mkdir_save_fat_buffer_3
1713 0000B910 09C0                <1>       or     eax, eax ; EAX = 0 -> cluster value is 0 or eocc
1714 0000B912 7475                <1>       jz     short loc_mkdir_save_fat_buffer_3
1715                              <1>       ; 01/03/2016
1716 0000B914 EB98                <1>       jmp    short loc_mkdir_save_fat_buffer_stc_retn
1717                              <1>
1718                              <1> loc_mkdir_set_ff_dir_entry_2:
1719                              <1>       ; ESI = Logical DOS Drive Description Table address
1720 0000B916 A1[F4930100]        <1>       mov    eax, [mkdir_FFCluster]
1721                              <1>       ; Load disk sectors as a directory cluster
1722 0000B91B E82C160000          <1>       call   load_FAT_sub_directory
1723 0000B920 7266                <1>       jc     short retn_make_fat_directory
1724                              <1>
1725                              <1>       ; eax = 0
1726                              <1>       ; ecx =  directory buffer sector count (<= 128)
1727                              <1>
1728 0000B922 BF40000800          <1>       mov    edi, Directory_Buffer + 64 ; 26/02/2016
1729                              <1>
1730                              <1>       ; 02/03/2016
1731 0000B927 668B4611            <1>       mov    ax, [esi+LD_BPB+BytesPerSec] ; 512
1732 0000B92B 66C1E802            <1>       shr    ax, 2 ; 'byte count / 4' for 'stosd'
1733 0000B92F F7E1                <1>       mul    ecx
1734 0000B931 89C1                <1>       mov    ecx, eax
1735 0000B933 6629C0              <1>       sub    ax, ax
1736 0000B936 F3AB                <1>       rep    stosd
1737                              <1>
1738                              <1>       ;;mov  al, 128 ; double word
1739                              <1>       ;;mul  ecx ; ecx =  directory buffer sector count
1740                              <1>       ;;mov  ecx, eax
1741                              <1>       ;shl   cx, 7 ; 128 * sector count
1742                              <1>       ;;sub  eax, eax
1743                              <1>       ;;sub  al, al ; 0
1744                              <1>       ;rep   stosd ; clear directory buffer
1745                              <1>
1746 0000B938 BF00000800          <1>       mov    edi, Directory_Buffer ; 26/02/2016
1747                              <1>
1748 0000B93D 56                  <1>       push   esi
1749                              <1>
1750 0000B93E BE[04940100]        <1>       mov    esi, mkdir_Name
1751 0000B943 66C7062E00          <1>       mov    word [esi], 2Eh ; db '.', '0'
1752                              <1>
1753 0000B948 A1[F4930100]        <1>       mov    eax, [mkdir_FFCluster]
1754 0000B94D 66B91000            <1>       mov    cx, 10h ; CL = Directory attribute
1755                              <1>                      ; CH = 0 -> File size is 0
1756 0000B951 E883000000          <1>       call   make_directory_entry
1757                              <1>
1758 0000B956 BF20000800          <1>       mov    edi, Directory_Buffer + 32 ; 26/02/2016
1759                              <1>
1760                              <1>       ; 03/03/2016
1761                              <1>       ; Following modification has been done according to
1762                              <1>       ; 'Microsoft Extensible Firmware Initiative
1763                              <1>       ; FAT32 File System Specification' document,
1764                              <1>       ; 'FAT: General Overview of On-Disk Format—Page 25'.
1765                              <1>       ; "Finally, you set DIR_FstClusLO and DIR_FstClusHI
1766                              <1>       ; for the dotdot entry (the second entry) to the
1767                              <1>       ; first cluster number of the directory in which you
1768                              <1>       ; just created the directory (value is 0 if this directory
1769                              <1>       ; is the root directory even for FAT32 volumes)."
1770                              <1>       ; (Correctness of this modification has been verified
1771                              <1>       ;  by using Windows 98 'scandisk.exe'.)
1772                              <1>
1773 0000B95B 29C0                <1>       sub    eax, eax
1774 0000B95D 3805[848A0100]      <1>       cmp    byte [Current_Dir_Level], al ; 0
1775 0000B963 7605                <1>       jna    short loc_mkdir_set_ff_dir_entry_3
1776 0000B965 A1[808A0100]        <1>       mov    eax, [Current_Dir_FCluster] ; parent dir
1777                              <1> loc_mkdir_set_ff_dir_entry_3:
1778 0000B96A 66C746012E00        <1>       mov    word [esi+1], 2Eh ; db '.', '0'
1779                              <1>
1780                              <1>       ;mov   cx, 10h
1781 0000B970 E864000000          <1>       call   make_directory_entry
1782                              <1>
1783 0000B975 5E                  <1>       pop    esi
1784                              <1>
1785 0000B976 C605[AC910100]02    <1>       mov    byte [DirBuff_ValidData], 2
1786 0000B97D E834010000          <1>       call   save_directory_buffer
1787 0000B982 0F8373FFFFFF        <1>         jnc     loc_mkdir_update_sub_dir_cluster
1788                              <1>
1789                              <1> retn_make_fat_directory:
1790 0000B988 C3                  <1>       retn
1791                              <1>
1792                              <1> loc_mkdir_save_fat_buffer_3:
1793                              <1>       ; 01/03/2016
1794                              <1>       ; byte [FAT_BuffValidData] = 2
1795 0000B989 E8BF190000          <1>       call   save_fat_buffer
1796 0000B98E 0F821AFFFFFF        <1>         jc      loc_mkdir_save_fat_buffer_stc_retn
1797                              <1>
1798 0000B994 803D[A2910100]01    <1>       cmp    byte [FAT_ClusterCounter], 1
1799 0000B99B 721B                <1>       jb     short loc_mkdir_save_fat_buffer_4
1800                              <1>
1801                              <1>       ; ESI = Logical DOS Drive Description Table address
1802 0000B99D A1[A2910100]        <1>       mov    eax, [FAT_ClusterCounter]
1803 0000B9A2 66BB01FF            <1>       mov    bx, 0FF01h ; add free clusters
1804 0000B9A6 E8371A0000          <1>       call   calculate_fat_freespace
1805                              <1>
1806                              <1>       ;inc   eax ; 0FFFFFFFFh -> 0 ; recalculation is needed!
1807                              <1>         ;jnz    short loc_mkdir_save_fat_buffer_4
1808                              <1>
1809                              <1>       ; ecx > 0 -> Recalculation is needed
1810 0000B9AB 09C9                <1>       or     ecx, ecx
1811 0000B9AD 7409                <1>         jz      short loc_mkdir_save_fat_buffer_4
```

```
                                <1>
0000B9AF 66BB00FF               <1>       mov    bx, 0FF00h ; recalculate free space
0000B9B3 E82A1A0000             <1>       call   calculate_fat_freespace
                                <1>
                                <1> loc_mkdir_save_fat_buffer_4:
0000B9B8 C605[03940100]00       <1>       mov    byte [mkdir_add_new_cluster], 0
                                <1>
                                <1> loc_mkdir_upd_parent_dir_lmdt:
0000B9BF E88D010000             <1>       call   update_parent_dir_lmdt
                                <1>
                                <1>       ; 01/03/2016
0000B9C4 803D[03940100]00       <1>       cmp    byte [mkdir_add_new_cluster], 0
0000B9CB 0F8723FEFFFF           <1>         ja      loc_mkdir_gffc_2
                                <1>
                                <1> loc_mkdir_retn_new_dir_cluster:
0000B9D1 A1[F4930100]           <1>       mov    eax, [mkdir_FFCluster]
0000B9D6 31D2                   <1>       xor    edx, edx
                                <1> loc_mkdir_retn:
0000B9D8 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>
0000B9D9 51                     <1>       push   ecx
                                <1>
0000B9DA 884F0B                 <1>       mov    [edi+11], cl ; Attributes
0000B9DD 6689471A               <1>       mov    [edi+26], ax ; FClusterLw, 26
0000B9E1 C1E810                 <1>       shr    eax, 16
0000B9E4 66894714               <1>       mov    [edi+20], ax ; FClusterHw, 20
0000B9E8 6631C0                 <1>       xor    ax, ax
0000B9EB 6689470C               <1>       mov    [edi+12], ax ; NTReserved, 12
                                <1>                           ; CrtTimeTenth, 13
0000B9EF 08ED                   <1>       or     ch, ch
0000B9F1 7402                   <1>       jz     short loc_make_direntry_set_filesize
                                <1>
0000B9F3 8B03                   <1>       mov    eax, [ebx]
                                <1>
                                <1> loc_make_direntry_set_filesize:
0000B9F5 89471C                 <1>       mov    [edi+28], eax ; FileSize, 28
                                <1>
0000B9F8 E88AFBFFFF             <1>       call   convert_file_name
                                <1>       ;EDI = Dir Entry Format File Name Location
                                <1>       ;ESI = Dot File Name Location (capitalized)
                                <1>
0000B9FD E816000000             <1>       call   convert_current_date_time
                                <1>       ; OUTPUT -> DX = Date in dos dir entry format
                                <1>         ;       AX = Time in dos dir entry format
0000BA02 6689470E               <1>       mov    [edi+14], ax ; CrtTime, 14
0000BA06 66895710               <1>       mov    [edi+16], dx ; CrtDate, 16
0000BA0A 66895712               <1>       mov    [edi+18], dx ; LastAccDate, 18
0000BA0E 66894716               <1>       mov    [edi+22], ax ; WrtTime, 14
0000BA12 66895718               <1>       mov    [edi+24], dx ; WrtDate, 16
0000BA16 59                     <1>       pop    ecx
                                <1>
0000BA17 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>
0000BA18 B404                   <1>       mov    ah, 04h ; Return Current Date
0000BA1A E817B0FFFF             <1>       call   int1Ah
                                <1>
0000BA1F 88E8                   <1>       mov    al, ch ; <- century BCD
0000BA21 240F                   <1>       and    al, 0Fh
0000BA23 88EC                   <1>       mov    ah, ch
0000BA25 C0EC04                 <1>       shr    ah, 4
0000BA28 D50A                   <1>       aad
0000BA2A 88C5                   <1>       mov    ch, al ; -> century
                                <1>
0000BA2C 88C8                   <1>       mov    al, cl ; <- year BCD
0000BA2E 240F                   <1>       and    al, 0Fh
0000BA30 88CC                   <1>       mov    ah, cl
0000BA32 C0EC04                 <1>       shr    ah, 4
0000BA35 D50A                   <1>       aad
0000BA37 88C1                   <1>       mov    cl, al ; -> year
                                <1>
0000BA39 88E8                   <1>       mov    al, ch
0000BA3B B464                   <1>       mov    ah, 100
0000BA3D F6E4                   <1>       mul    ah
0000BA3F 30ED                   <1>       xor    ch, ch
0000BA41 6601C8                 <1>       add    ax, cx
```

```
1917 0000BA44 662DBC07            <1>       sub   ax, 1980 ; ms-dos epoch
1918 0000BA48 6689C1              <1>       mov   cx, ax
1919                              <1>
1920 0000BA4B 88F0                <1>       mov   al, dh ; <- month in bcd
1921 0000BA4D 240F                <1>       and   al, 0Fh
1922 0000BA4F 88F4                <1>       mov   ah, dh
1923 0000BA51 C0EC04              <1>       shr   ah, 4
1924 0000BA54 D50A                <1>       aad
1925 0000BA56 88C6                <1>       mov   dh, al ; -> month
1926                              <1>
1927 0000BA58 88D0                <1>       mov   al, dl ; <- day BCD
1928 0000BA5A 240F                <1>       and   al, 0Fh
1929 0000BA5C 88D4                <1>       mov   ah, dl
1930 0000BA5E C0EC04              <1>       shr   ah, 4
1931 0000BA61 D50A                <1>       aad
1932 0000BA63 88C2                <1>       mov   dl, al ; -> day
1933                              <1>
1934 0000BA65 88C8                <1>       mov   al, cl ; count of years from 1980
1935 0000BA67 66C1E004            <1>       shl   ax, 4
1936 0000BA6B 08F0                <1>       or    al, dh ; month of year, 1 to 12
1937 0000BA6D 66C1E005            <1>       shl   ax, 5
1938 0000BA71 08D0                <1>       or    al, dl ; day of year, 1 to 31
1939                              <1>
1940 0000BA73 6650                <1>       push  ax ; push date
1941                              <1>
1942 0000BA75 B402                <1>       mov   ah, 02h ; Return Current Time
1943 0000BA77 E8BAAFFFFF          <1>       call  int1Ah
1944                              <1>
1945 0000BA7C 88E8                <1>       mov   al, ch ; <- hours BCD
1946 0000BA7E 240F                <1>       and   al, 0Fh
1947 0000BA80 88EC                <1>       mov   ah, ch
1948 0000BA82 C0EC04              <1>       shr   ah, 4
1949 0000BA85 D50A                <1>       aad
1950 0000BA87 88C5                <1>       mov   ch, al ; -> hours
1951                              <1>
1952 0000BA89 88C8                <1>       mov   al, cl ; <- minutes BCD
1953 0000BA8B 240F                <1>       and   al, 0Fh
1954 0000BA8D 88CC                <1>       mov   ah, cl
1955 0000BA8F C0EC04              <1>       shr   ah, 4
1956 0000BA92 D50A                <1>       aad
1957 0000BA94 88C1                <1>       mov   cl, al ; -> minutes
1958                              <1>
1959 0000BA96 88F0                <1>       mov   al, dh ; <- seconds BCD
1960 0000BA98 240F                <1>       and   al, 0Fh
1961 0000BA9A 88F4                <1>       mov   ah, dh
1962 0000BA9C C0EC04              <1>       shr   ah, 4
1963 0000BA9F D50A                <1>       aad
1964 0000BAA1 88C6                <1>       mov   dh, al ; -> seconds
1965                              <1>
1966 0000BAA3 88E8                <1>       mov   al, ch ; hours
1967 0000BAA5 66C1E006            <1>       shl   ax, 6
1968 0000BAA9 08C8                <1>       or    al, cl ; minutes
1969 0000BAAB 66C1E005            <1>       shl   ax, 5
1970 0000BAAF D0EE                <1>       shr   dh, 1 ; 2 seconds
1971                              <1>       ; There is a bug in TRDOS v1 here !
1972                              <1>       ; it was 'or al, dl' !
1973 0000BAB1 08F0                <1>       or    al, dh ; seconds
1974                              <1>
1975 0000BAB3 665A                <1>       pop   dx ; pop date
1976                              <1>
1977 0000BAB5 C3                  <1>       retn
1978                              <1>
1979                              <1> save_directory_buffer:
1980                              <1>       ; 15/10/2016
1981                              <1>       ; 23/03/2016
1982                              <1>       ; 26/02/2016
1983                              <1>       ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)
1984                              <1>       ; 01/08/2011
1985                              <1>       ; 14/03/2010
1986                              <1>       ; INPUT ->
1987                              <1>       ;       none
1988                              <1>       ; OUTPUT ->
1989                              <1>       ;  cf = 0 -> write OK...
1990                              <1>       ;  cf = 1 -> error code in AL (EAX)
1991                              <1>       ;  cf = 1 & AL = 0Dh => CH & CL = FS & FAT type
1992                              <1>       ;  EBX = Directory Buffer Address
1993                              <1>       ;
1994                              <1>       ;  (EAX, ECX, EDX will be modified)
1995                              <1>
1996 0000BAB6 BB00000800          <1>       mov   ebx, Directory_Buffer
1997 0000BABB 803D[AC910100]02    <1>       cmp   byte [DirBuff_ValidData], 2
1998 0000BAC2 7403                <1>       je    short loc_save_dir_buffer
1999 0000BAC4 31C0                <1>       xor   eax, eax
2000 0000BAC6 C3                  <1>       retn
2001                              <1>
2002                              <1> loc_save_dir_buffer:
2003 0000BAC7 56                  <1>       push  esi
2004 0000BAC8 31DB                <1>       xor   ebx, ebx
2005 0000BACA 8A3D[AA910100]      <1>         mov     bh, [DirBuff_DRV]
2006 0000BAD0 80EF41              <1>       sub   bh, 'A'
2007 0000BAD3 BE00010900          <1>         mov     esi, Logical_DOSDisks
2008 0000BAD8 01DE                <1>       add   esi, ebx
2009 0000BADA 668B4E03            <1>         mov     cx, [esi+LD_FATType]
2010                              <1>       ; CH = FS Type (A1h for FS)
2011                              <1>       ; CL = FAT Type (0 for FS)
2012 0000BADE 08C9                <1>       or    cl, cl
2013 0000BAE0 7433                <1>       jz    short loc_save_dir_buff_stc_retn
2014                              <1>
2015                              <1> loc_save_dir_buffer_check_cluster_no:
2016 0000BAE2 A1[B1910100]        <1>       mov   eax, [DirBuff_Cluster]
2017 0000BAE7 28FF                <1>       sub   bh, bh ; ebx = 0
2018 0000BAE9 09C0                <1>       or    eax, eax
2019 0000BAEB 7540                <1>       jnz   short loc_save_sub_dir_buffer
2020 0000BAED 8A25[AB910100]      <1>       mov   ah, [DirBuff_FATType]
2021 0000BAF3 FEC3                <1>       inc   bl ;  bl = 1
```

```
2022 0000BAF5 38DC              <1>       cmp    ah, bl
2023 0000BAF7 721D              <1>       jb     short loc_save_dir_buff_inv_data_retn
2024 0000BAF9 FEC3              <1>       inc    bl ; bl = 2
2025 0000BAFB 38E3              <1>       cmp    bl, ah
2026 0000BAFD 7217              <1>       jb     short loc_save_dir_buff_inv_data_retn
2027                            <1>
2028                            <1> loc_save_root_dir_buffer:
2029 0000BAFF 668B5E17          <1>       mov    bx, [esi+LD_BPB+RootDirEnts]
2030 0000BB03 6683C30F          <1>       add    bx, 15
2031 0000BB07 66C1EB04          <1>       shr    bx, 4 ; 16 dir entries per sector
2032 0000BB0B 6609DB            <1>       or     bx, bx
2033 0000BB0E 7405              <1>       jz     short loc_save_dir_buff_stc_retn
2034                            <1>       ;mov   ecx, ebx
2035 0000BB10 8B4664            <1>       mov    eax, [esi+LD_ROOTBegin] ; 26/02/2016
2036 0000BB13 EB23              <1>       jmp    short loc_write_directory_to_disk
2037                            <1>
2038                            <1> loc_save_dir_buff_stc_retn:
2039 0000BB15 F9                <1>       stc
2040                            <1> loc_save_dir_buff_inv_data_retn:
2041                            <1>       ; 15/10/2016 (0Dh -> 29)
2042 0000BB16 B01D              <1>       mov    al, 29 ; Invalid data !
2043 0000BB18 C605[AC910100]00  <1>       mov    byte [DirBuff_ValidData], 0
2044 0000BB1F EB05              <1>       jmp    short loc_save_dir_buff_retn
2045                            <1>
2046                            <1> loc_write_directory_to_disk_err:
2047                            <1>       ; 15/10/2016 (disk write error code, 1Dh -> 18)
2048 0000BB21 B812000000        <1>       mov    eax, 18 ; Drive not ready or write error
2049                            <1>
2050                            <1> loc_save_dir_buff_retn:
2051 0000BB26 BB00000800        <1>       mov    ebx, Directory_Buffer
2052 0000BB2B 5E                <1>       pop    esi
2053 0000BB2C C3                <1>       retn
2054                            <1>
2055                            <1> loc_save_sub_dir_buffer:
2056                            <1>       ; ebx  = 0
2057 0000BB2D 83E802            <1>       sub    eax, 2
2058 0000BB30 8A5E13            <1>       mov    bl, [esi+LD_BPB+SecPerClust]
2059 0000BB33 F7E3              <1>       mul    ebx
2060 0000BB35 034668            <1>         add     eax, [esi+LD_DATABegin]
2061                            <1>       ;mov   ecx, ebx
2062                            <1>
2063                            <1> loc_write_directory_to_disk:
2064 0000BB38 89D9              <1>       mov    ecx, ebx
2065 0000BB3A BB00000800        <1>       mov    ebx, Directory_Buffer
2066 0000BB3F E8F86F0000        <1>       call   disk_write
2067 0000BB44 72DB              <1>       jc     short loc_write_directory_to_disk_err
2068                            <1>
2069                            <1> loc_save_dir_buff_validate_retn:
2070 0000BB46 C605[AC910100]01  <1>       mov    byte [DirBuff_ValidData], 1
2071 0000BB4D 31C0              <1>       xor    eax, eax
2072                            <1>       ; 26/02/2016
2073 0000BB4F EBD5              <1>       jmp    short loc_save_dir_buff_retn
2074                            <1>
2075                            <1> update_parent_dir_lmdt:
2076                            <1>       ; 29/12/2017
2077                            <1>       ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)
2078                            <1>       ; 01/08/2011
2079                            <1>       ; 16/10/2010
2080                            <1>       ;
2081                            <1>       ; INPUT ->
2082                            <1>       ;      none
2083                            <1>       ; OUTPUT ->
2084                            <1>       ;      (last modification date & time of the parent dir
2085                            <1>       ;      will be changed/updated)
2086                            <1>       ;
2087                            <1>       ; (EAX, EBX, ECX, EDX, EDI will be changed)
2088                            <1>
2089 0000BB51 29C0              <1>       sub    eax, eax
2090 0000BB53 8A25[848A0100]    <1>       mov    ah, [Current_Dir_Level]
2091 0000BB59 A0[858A0100]      <1>       mov    al, [Current_FATType]
2092 0000BB5E 3C01              <1>       cmp    al, 1
2093 0000BB60 723A              <1>       jb     short loc_UPDLMDT_proc_retn
2094                            <1>
2095                            <1> loc_update_parent_dir_lm_date_time:
2096 0000BB62 08E4              <1>       or     ah, ah
2097 0000BB64 7436              <1>       jz     short loc_UPDLMDT_proc_retn
2098                            <1>
2099 0000BB66 56                <1>       push   esi ; *
2100 0000BB67 8825[24940100]    <1>       mov    [UPDLMDT_CDirLevel], ah
2101 0000BB6D 8B15[808A0100]    <1>       mov    edx, [Current_Dir_FCluster]
2102 0000BB73 8915[25940100]    <1>       mov    [UPDLMDT_CDirFCluster], edx
2103                            <1>
2104 0000BB79 FECC              <1>       dec    ah
2105 0000BB7B B90C000000        <1>       mov    ecx, 12
2106 0000BB80 BE[E3910100]      <1>         mov     esi, PATH_Array
2107                            <1>
2108 0000BB85 8825[848A0100]    <1>       mov    [Current_Dir_Level], ah
2109 0000BB8B 08E4              <1>       or     ah, ah
2110 0000BB8D 750E              <1>       jnz    short loc_update_parent_dir_lmdt_load_sub_dir_1
2111 0000BB8F 803D[858A0100]02  <1>       cmp    byte [Current_FATType], 2
2112 0000BB96 770B              <1>       ja     short loc_update_parent_dir_lmdt_load_sub_dir_2
2113 0000BB98 28C0              <1>       sub    al, al ; eax = 0
2114 0000BB9A EB0A              <1>       jmp    short loc_update_parent_dir_lmdt_load_sub_dir_3
2115                            <1>
2116                            <1> loc_UPDLMDT_proc_retn:
2117 0000BB9C C3                <1>       retn
2118                            <1>
2119                            <1> loc_update_parent_dir_lmdt_load_sub_dir_1:
2120 0000BB9D B010              <1>       mov    al, 16
2121 0000BB9F F6E4              <1>       mul    ah
2122 0000BBA1 01C6              <1>       add    esi, eax
2123                            <1>
2124                            <1> loc_update_parent_dir_lmdt_load_sub_dir_2:
2125 0000BBA3 8B460C            <1>       mov    eax, [esi+12] ; Parent Dir First Cluster
2126                            <1>
```

```
                                <1> loc_update_parent_dir_lmdt_load_sub_dir_3:
0000BBA6 A3[808A0100]           <1>       mov   [Current_Dir_FCluster], eax
                                <1>
0000BBAB 83C610                 <1>       add   esi, 16
0000BBAE 66BF[0A93]             <1>       mov   di, Dir_File_Name
0000BBB2 F3A4                   <1>       rep   movsb
                                <1>
0000BBB4 BE00010900             <1>       mov   esi, Logical_DOSDisks
0000BBB9 29DB                   <1>       sub   ebx, ebx
0000BBBB 8A3D[868A0100]         <1>       mov   bh, [Current_Drv]
0000BBC1 01DE                   <1>       add   esi, ebx
0000BBC3 E88FF7FFFF             <1>       call  reload_current_directory
0000BBC8 7230                   <1>       jc    short loc_update_parent_dir_lmdt_restore_cdirlevel
                                <1>
                                <1> loc_update_parent_dir_lmdt_locate_dir:
0000BBCA BE[0A930100]           <1>       mov   esi, Dir_File_Name
0000BBCF 6631C9                 <1>       xor   cx, cx
0000BBD2 66B81008               <1>       mov   ax, 0810h ; Only directories
0000BBD6 E8B5F6FFFF             <1>         call    locate_current_dir_file
                                <1>       ; EDI = DirBuff Directory Entry Address
0000BBDB 721D                   <1>       jc    short loc_update_parent_dir_lmdt_restore_cdirlevel
                                <1>
0000BBDD E836FEFFFF             <1>       call  convert_current_date_time
0000BBE2 66895712               <1>       mov   [edi+18], dx ; Last Access Date
0000BBE6 66895718               <1>       mov   [edi+24], dx ; Last Write Date
0000BBEA 66894716               <1>       mov   [edi+22], ax ; Last Write Time
                                <1>
0000BBEE C605[AC910100]02       <1>       mov   byte [DirBuff_ValidData], 2
0000BBF5 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
0000BBFA 8A25[24940100]         <1>       mov   ah, [UPDLMDT_CDirLevel]
0000BC00 8825[848A0100]         <1>       mov   [Current_Dir_Level], ah
0000BC06 8B15[25940100]         <1>         mov     edx, [UPDLMDT_CDirFCluster]
0000BC0C 8915[808A0100]         <1>       mov   [Current_Dir_FCluster], edx
                                <1>
0000BC12 5E                     <1>       pop   esi ; *
0000BC13 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>
0000BC14 66A3[54940100]         <1>       mov   [DLN_EntryNumber], ax
0000BC1A C605[56940100]40       <1>         mov     byte [DLN_40h], 40h
                                <1>
0000BC21 E858000000             <1>       call  locate_current_dir_entry
0000BC26 7308                   <1>       jnc   short loc_dln_check_attributes
0000BC28 C3                     <1>       retn
                                <1>
                                <1> loc_dln_longname_not_found:
0000BC29 B802000000             <1>       mov   eax, 2
0000BC2E F9                     <1>       stc
0000BC2F C3                     <1>       retn
                                <1>
                                <1> loc_dln_check_attributes:
0000BC30 B00F                   <1>       mov   al, 0Fh  ; long name
0000BC32 8A670B                 <1>       mov   ah, [edi+0Bh] ; dir entry attributes
0000BC35 38C4                   <1>       cmp   ah, al
0000BC37 75F0                   <1>       jne   short loc_dln_longname_not_found
0000BC39 8A27                   <1>       mov   ah, [edi]
0000BC3B 2A25[56940100]         <1>       sub   ah, [DLN_40h]
0000BC41 76E6                   <1>       jna   short loc_dln_longname_not_found
0000BC43 80FC14                 <1>       cmp   ah, 14h ; 84-64=20 -> 20*13=260 bytes
0000BC46 77E1                   <1>       ja    short loc_dln_longname_not_found
                                <1>
0000BC48 C607E5                 <1>       mov   byte [edi], 0E5h  ; deleted sign
0000BC4B C605[AC910100]02       <1>       mov   byte [DirBuff_ValidData], 2 ; changed/write sign
0000BC52 C605[56940100]00       <1>       mov   byte [DLN_40h], 0 ; 40h -> 0
                                <1>
                                <1> loc_dln_delete_next_ln_entry:
0000BC59 80FC01                 <1>       cmp   ah, 1
0000BC5C 7616                   <1>       jna   short loc_dln_longname_retn
                                <1> loc_dln_delete_next_ln_entry_0:
0000BC5E 66FF05[54940100]       <1>       inc   word [DLN_EntryNumber]
0000BC65 0FB705[54940100]       <1>       movzx eax, word [DLN_EntryNumber]
0000BC6C E80D000000             <1>       call  locate_current_dir_entry
0000BC71 73BD                   <1>       jnc   short loc_dln_check_attributes
                                <1>
                                <1> loc_dln_longname_stc_retn:
0000BC73 C3                     <1>       retn
                                <1>
                                <1> loc_dln_longname_retn:
                                <1>       ;cmp  byte [DirBuff_ValidData], 2
                                <1>       ;jne  short loc_dln_longname_retn_xor_eax
0000BC74 E83DFEFFFF             <1>       call  save_directory_buffer
0000BC79 72F8                   <1>       jc    short loc_dln_longname_stc_retn
                                <1>
                                <1> loc_dln_longname_retn_xor_eax:
0000BC7B 31C0                   <1>       xor   eax, eax
0000BC7D C3                     <1>       retn
                                <1>
                                <1> locate_current_dir_entry:
                                <1>       ; 16/10/2016
```

```
2232                            <1>       ; 15/10/2016
2233                            <1>       ; 23/03/2016
2234                            <1>       ; 27/02/2016 (TRDOS 386 = TRDOS v2.0)
2235                            <1>       ; 01/08/2011 (DIR.ASM, 'proc_locate_current_dir_entry')
2236                            <1>       ; 07/03/2010
2237                            <1>       ; INPUT ->
2238                            <1>       ;     EAX = Directory Entry (Index) Number (< 65536)
2239                            <1>       ; OUTPUT ->
2240                            <1>       ;     EDI = Directory Entry Address
2241                            <1>       ;     EAX = Cluster Number of Directory Buffer
2242                            <1>       ;     EBX = Directory Buffer Entry Offset
2243                            <1>       ;     ECX = DirBuff Valid Data identifier (CL)
2244                            <1>       ;     If CF = 0 and CL = 2 then
2245                            <1>       ;        directory buffer modified and
2246                            <1>       ;        must be written to disk.
2247                            <1>       ;     If CF = 0  and CL = 1 then
2248                            <1>       ;        dir buffer has been written to disk, already.
2249                            <1>       ;     CF = 1 -> Error code in EAX (AL)
2250                            <1>       ;
2251                            <1>       ; (Modified registers: EAX, EDX, ECX, EBX, EDI)
2252                            <1>
2253                            <1> loc_locate_current_dir_entry:
2254 0000BC7E 56               <1>       push   esi
2255 0000BC7F 89C1             <1>       mov    ecx, eax
2256 0000BC81 BA20000000       <1>       mov    edx, 32
2257 0000BC86 F7E2             <1>       mul    edx
2258 0000BC88 A3[60940100]     <1>       mov    [LCDE_ByteOffset], eax
2259 0000BC8D 31DB             <1>       xor    ebx, ebx
2260 0000BC8F 8A3D[868A0100]   <1>       mov    bh, [Current_Drv]
2261 0000BC95 A0[AA910100]     <1>         mov     al, [DirBuff_DRV]
2262 0000BC9A 2C41             <1>       sub    al, 'A'
2263 0000BC9C BE00010900       <1>         mov     esi, Logical_DOSDisks
2264 0000BCA1 01DE             <1>       add    esi, ebx
2265 0000BCA3 38C7             <1>       cmp    bh, al
2266 0000BCA5 0F8592000000     <1>         jne     loc_lcde_reload_current_directory
2267                            <1> loc_lcde_cdl_check:
2268 0000BCAB 803D[848A0100]00 <1>       cmp    byte [Current_Dir_Level], 0
2269 0000BCB2 772A             <1>       ja     short loc_lcde_calc_dirbuff_cluster_offset
2270                            <1>       ; 27/02/2016
2271                            <1>       ; TRDOS v1 has bug here for FAT32 fs !
2272                            <1>       ; (Root Directory Entries for FAT32 = 0)
2273 0000BCB4 807E0303           <1>       cmp    byte [esi+LD_FATType], 3  ; FAT32
2274 0000BCB8 7324             <1>       jnb    short loc_lcde_calc_dirbuff_cluster_offset
2275                            <1>
2276                            <1> loc_lcde_cdl_check_FAT12_16:
2277 0000BCBA 668B4617         <1>       mov    ax, [esi+LD_BPB+RootDirEnts]
2278 0000BCBE 6648             <1>       dec    ax
2279                            <1>       ;xor   dx, dx
2280 0000BCC0 6639C8           <1>       cmp    ax, cx ; cx = Directory Entry (Index) Number
2281 0000BCC3 720E             <1>       jb     short loc_lcde_stc_12h_retn
2282 0000BCC5 66890D[58940100] <1>       mov    [LCDE_EntryIndex], cx
2283 0000BCCC 31C0             <1>       xor    eax, eax
2284 0000BCCE E993000000       <1>         jmp     loc_lcde_check_dir_buffer_cluster
2285                            <1>
2286                            <1> loc_lcde_stc_12h_retn:
2287 0000BCD3 5E               <1>       pop    esi
2288 0000BCD4 89CB             <1>       mov    ebx, ecx
2289 0000BCD6 89D1             <1>       mov    ecx, edx
2290                            <1>       ; 16/10/2016 (12h -> 12)
2291 0000BCD8 B80C000000       <1>       mov    eax, 12 ; No more files
2292 0000BCDD C3               <1>       retn
2293                            <1>
2294                            <1> loc_lcde_calc_dirbuff_cluster_offset:
2295 0000BCDE 8A5E13           <1>       mov    bl, [esi+LD_BPB+SecPerClust]
2296 0000BCE1 30FF             <1>       xor    bh, bh
2297 0000BCE3 668B4611         <1>       mov    ax, [esi+LD_BPB+BytesPerSec]
2298 0000BCE7 66F7E3           <1>       mul    bx
2299 0000BCEA 6609D2           <1>       or     dx, dx ; If bytes per cluster > 32KB it is invalid
2300 0000BCED 755D             <1>       jnz    short loc_lcde_invalid_format
2301                            <1>       ;mov   ecx, eax
2302 0000BCEF 6689C1           <1>       mov    cx, ax ; BYTES PER CLUSTER
2303 0000BCF2 A1[60940100]     <1>       mov    eax, [LCDE_ByteOffset]
2304                            <1>       ;sub   edx, edx
2305 0000BCF7 F7F1             <1>       div    ecx
2306 0000BCF9 3DFFFF0000       <1>       cmp    eax, 65535
2307 0000BCFE 774C             <1>       ja     short loc_lcde_invalid_format
2308                            <1>
2309                            <1>       ; cluster sequence number of directory (< 65536)
2310 0000BD00 66A3[5A940100]   <1>       mov    [LCDE_ClusterSN], ax
2311                            <1>
2312 0000BD06 6689D0           <1>       mov    ax, dx ; byte offset in cluster (directory buffer)
2313 0000BD09 66BB2000         <1>       mov    bx, 32 ; ; 1 dir entry = 32 bytes
2314 0000BD0D 6629D2           <1>         sub     dx, dx  ; 0
2315 0000BD10 66F7F3           <1>       div    bx
2316 0000BD13 66A3[58940100]   <1>       mov    [LCDE_EntryIndex], ax ; dir entry index/sequence number
2317                            <1>                                    ; (in directory buffer/cluster)
2318                            <1> loc_lcde_get_current_sub_dir_fcluster:
2319 0000BD19 A1[808A0100]     <1>       mov    eax, [Current_Dir_FCluster]
2320                            <1>
2321                            <1> loc_lcde_get_next_cluster:
2322 0000BD1E 66833D[5A940100]00 <1>       cmp    word [LCDE_ClusterSN], 0
2323 0000BD26 763E             <1>       jna    short loc_lcde_check_dir_buffer_cluster
2324 0000BD28 A3[5C940100]     <1>       mov    [LCDE_Cluster], eax
2325 0000BD2D E834100000       <1>       call   get_next_cluster
2326 0000BD32 7220             <1>       jc     short loc_lcde_check_gnc_error
2327 0000BD34 66FF0D[5A940100] <1>       dec    word [LCDE_ClusterSN]
2328 0000BD3B EBE1             <1>       jmp    short loc_lcde_get_next_cluster
2329                            <1>
2330                            <1> loc_lcde_reload_current_directory:
2331 0000BD3D 51               <1>       push   ecx
2332 0000BD3E E814F6FFFF       <1>       call   reload_current_directory
2333 0000BD43 59               <1>       pop    ecx
2334 0000BD44 0F8361FFFFFF     <1>         jnc     loc_lcde_cdl_check
2335 0000BD4A 5E               <1>       pop    esi
2336 0000BD4B C3               <1>       retn
```

```
2337                                    <1>
2338                                    <1> loc_lcde_invalid_format:
2339                                    <1>         ; 15/10/2016 (0Bh -> 28)
2340 0000BD4C B81C000000                <1>         mov     eax, 28 ; Invalid Format !
2341                                    <1> loc_lcde_drive_not_ready_read_err:
2342 0000BD51 F9                        <1>         stc
2343 0000BD52 5E                        <1>         pop     esi
2344 0000BD53 C3                        <1>         retn
2345                                    <1>
2346                                    <1> loc_lcde_check_gnc_error:
2347 0000BD54 09C0                      <1>         or      eax, eax
2348 0000BD56 75F9                      <1>         jnz     short loc_lcde_drive_not_ready_read_err
2349 0000BD58 66FF0D[5A940100]          <1>         dec     word [LCDE_ClusterSN]
2350 0000BD5F 75EB                      <1>         jnz     short loc_lcde_invalid_format
2351 0000BD61 A1[5C940100]              <1>         mov     eax, [LCDE_Cluster]
2352                                    <1>
2353                                    <1> loc_lcde_check_dir_buffer_cluster:
2354 0000BD66 3B05[B1910100]            <1>         cmp     eax, [DirBuff_Cluster]
2355 0000BD6C 755C                      <1>         jne     short loc_lcde_load_dir_cluster
2356 0000BD6E 803D[AC910100]00          <1>         cmp     byte [DirBuff_ValidData], 0
2357 0000BD75 7727                      <1>         ja      short lcde_check_dir_buffer_cluster_next
2358 0000BD77 803D[848A0100]00          <1>         cmp     byte [Current_Dir_Level], 0
2359 0000BD7E 775F                      <1>         ja      short loc_lcde_load_dir_cluster_0
2360                                    <1>         ; 27/02/2016
2361                                    <1>         ; TRDOS v1 has bug here for FAT32 fs !
2362 0000BD80 807E0303                  <1>         cmp     byte [esi+LD_FATType], 3  ; FAT32
2363 0000BD84 7359                      <1>         jnb     short loc_lcde_load_dir_cluster_0
2364                                    <1>         ;
2365 0000BD86 0FB74E17                  <1>         movzx   ecx, word [esi+LD_BPB+RootDirEnts]
2366 0000BD8A 6683C10F                  <1>         add     cx, 15 ; round up (16 entries per sector)
2367 0000BD8E 66C1E904                  <1>         shr     cx, 4 ; 1 sector contains 16 dir entries
2368                                    <1>
2369 0000BD92 8B4664                    <1>           mov     eax, [esi+LD_ROOTBegin]
2370 0000BD95 EB54                      <1>         jmp     short loc_lcde_load_dir_cluster_1
2371                                    <1>
2372                                    <1> loc_lcde_validate_dirBuff:
2373 0000BD97 C605[AC910100]01          <1>         mov     byte [DirBuff_ValidData], 1
2374                                    <1>
2375                                    <1> lcde_check_dir_buffer_cluster_next:
2376 0000BD9E 0FB71D[58940100]          <1>         movzx   ebx, word [LCDE_EntryIndex]
2377 0000BDA5 663B1D[AF910100]          <1>         cmp     bx, [DirBuff_LastEntry]
2378 0000BDAC 779E                      <1>         ja      short loc_lcde_invalid_format
2379 0000BDAE B820000000                <1>         mov     eax, 32
2380 0000BDB3 F7E3                      <1>         mul     ebx
2381                                    <1>         ;or     edx, edx
2382                                    <1>         ;jnz    short loc_lcde_invalid_format
2383                                    <1>
2384 0000BDB5 BF00000800                <1>         mov     edi, Directory_Buffer
2385 0000BDBA 01C7                      <1>         add     edi, eax ; add entry offset to buffer address
2386                                    <1>
2387                                    <1> loc_lcde_dir_buffer_last_check:
2388 0000BDBC A1[B1910100]              <1>         mov     eax, [DirBuff_Cluster]
2389 0000BDC1 0FB60D[AC910100]          <1>         movzx   ecx, byte [DirBuff_ValidData]
2390                                    <1>
2391                                    <1> loc_lcde_retn:
2392 0000BDC8 5E                        <1>         pop     esi
2393 0000BDC9 C3                        <1>         retn
2394                                    <1>
2395                                    <1> loc_lcde_load_dir_cluster:
2396                                    <1>         ;cmp    byte [DirBuff_ValidData], 2
2397                                    <1>         ;jne    short loc_lcde_load_dir_cluster_n2
2398 0000BDCA 50                        <1>         push    eax
2399 0000BDCB E8E6FCFFFF                <1>         call    save_directory_buffer
2400 0000BDD0 58                        <1>         pop     eax
2401 0000BDD1 72F5                      <1>         jc      short loc_lcde_retn
2402                                    <1>
2403                                    <1> loc_lcde_load_dir_cluster_n2:
2404 0000BDD3 C605[AC910100]00          <1>         mov     byte [DirBuff_ValidData], 0
2405 0000BDDA A3[B1910100]              <1>         mov     [DirBuff_Cluster], eax
2406                                    <1>
2407                                    <1> loc_lcde_load_dir_cluster_0:
2408 0000BDDF 83E802                    <1>         sub     eax, 2
2409 0000BDE2 0FB64E13                  <1>         movzx   ecx, byte [esi+LD_BPB+SecPerClust]
2410 0000BDE6 F7E1                      <1>         mul     ecx
2411 0000BDE8 034668                    <1>           add     eax, [esi+LD_DATABegin]
2412                                    <1>
2413                                    <1> loc_lcde_load_dir_cluster_1:
2414 0000BDEB BB00000800                <1>         mov     ebx, Directory_Buffer
2415                                    <1>         ; ecx = sector count
2416 0000BDF0 E8566D0000                <1>         call    disk_read
2417 0000BDF5 73A0                      <1>         jnc     short loc_lcde_validate_dirBuff
2418                                    <1>
2419                                    <1>         ; 15/10/2016
2420                                    <1>         ; (Disk read error instead of drv not ready err)
2421 0000BDF7 B811000000                <1>         mov     eax, 17 ; Drive not ready or read error !
2422 0000BDFC EBCA                      <1>         jmp     short loc_lcde_retn
2423                                    <1>
2424                                    <1>
2425                                    <1> remove_file:
2426                                    <1>         ; 15/10/2016
2427                                    <1>         ; 28/02/2016 (TRDOS 386 = TRDOS v2.0)
2428                                    <1>         ; 10/04/2011 (FILE.ASM, 'proc_delete_file')
2429                                    <1>         ; 09/08/2010
2430                                    <1>         ; INPUT ->
2431                                    <1>         ;       EDI = Directory Buffer Entry Address
2432                                    <1>         ;        CX = Directory Buffer Entry Counter/Index
2433                                    <1>         ;        BL = Longname Entry Length
2434                                    <1>         ;        BH = Logical DOS Drive Number
2435                                    <1>
2436 0000BDFE 29C0                      <1>         sub     eax, eax
2437 0000BE00 88FC                      <1>         mov     ah, bh
2438 0000BE02 BE00010900                <1>         mov     esi, Logical_DOSDisks
2439 0000BE07 01C6                      <1>         add     esi, eax
2440                                    <1>
2441 0000BE09 807E0301                  <1>         cmp     byte [esi+LD_FATType], 1
```

```
2442 0000BE0D 7312              <1>       jnb    short loc_del_fat_file
2443                            <1>
2444 0000BE0F 807E04A1          <1>       cmp    byte [esi+LD_FSType], 0A1h
2445 0000BE13 7406              <1>       je     short loc_del_fs_file
2446                            <1>
2447                            <1> loc_del_file_invalid_format:
2448 0000BE15 30E4              <1>       xor    ah, ah
2449                            <1>       ; 15/10/2016 (0Bh -> 28)
2450 0000BE17 B01C              <1>       mov    al, 28  ; Invalid Format
2451 0000BE19 F9                <1>       stc
2452 0000BE1A C3                <1>       retn
2453                            <1>
2454                            <1> loc_del_fs_file:
2455 0000BE1B E83F0F0000        <1>       call   delete_fs_file
2456 0000BE20 C3                <1>       retn
2457                            <1>
2458                            <1> loc_del_fat_file:
2459 0000BE21 E808000000        <1>       call   delete_directory_entry
2460 0000BE26 7205              <1>       jc     short loc_del_file_err_retn
2461                            <1>
2462                            <1> loc_delfile_unlink_cluster_chain:
2463 0000BE28 E863170000        <1>       call   truncate_cluster_chain
2464                            <1>       ;jc    short loc_del_file_err_retn
2465                            <1>
2466                            <1> loc_delfile_return:
2467                            <1> loc_del_file_err_retn:
2468 0000BE2D C3                <1>       retn
2469                            <1>
2470                            <1> delete_directory_entry:
2471                            <1>       ; 15/10/2016
2472                            <1>       ; 28/02/2016 (TRDOS 386 = TRDOS v2.0)
2473                            <1>       ; 01/08/2011 (DIR.ASM, 'proc_delete_directory_entry')
2474                            <1>       ; 10/04/2011
2475                            <1>       ; INPUT ->
2476                            <1>       ;      ESI = Logical Dos Drive Descripton Table Address
2477                            <1>       ;      EDI = Directory Buffer Entry Address
2478                            <1>       ;       CX = Directory Buffer Entry Counter/Index
2479                            <1>       ;       BL = Longname Entry Length
2480                            <1>       ; OUTPUT ->
2481                            <1>       ;      ESI = Logical dos drive descripton table address
2482                            <1>       ;      EAX = First cluster to be truncated/unlinked
2483                            <1>       ;       CF = 1 -> Error code in EAX (AL)
2484                            <1>       ;       CF = 0 & BH <> 0 -> LMDT write error  (BH = 1)
2485                            <1>       ;       CF = 0 & BL <> 0 -> Long name delete error (BL = FFh)
2486                            <1>       ;
2487                            <1>       ;  (EDI, EBX, ECX register contents will be changed)
2488                            <1>
2489 0000BE2E 881D[EE930100]     <1>       mov    [DelFile_LNEL], bl
2490 0000BE34 66890D[EC930100]  <1>       mov    [DelFile_EntryCounter], cx
2491                            <1>
2492 0000BE3B 668B4714          <1>       mov    ax, [edi+20] ; First Cluster High Word
2493 0000BE3F C1E010            <1>       shl    eax, 16
2494 0000BE42 668B471A          <1>       mov    ax, [edi+26] ; First Cluster Low Word
2495                            <1>
2496 0000BE46 A3[E8930100]      <1>       mov    [DelFile_FCluster], eax
2497                            <1>
2498                            <1> loc_del_short_name:
2499 0000BE4B C607E5            <1>       mov    byte [edi], 0E5h  ; Deleted sign
2500                            <1>
2501 0000BE4E C605[AC910100]02  <1>       mov    byte [DirBuff_ValidData], 2
2502 0000BE55 E85CFCFFFF        <1>       call   save_directory_buffer
2503 0000BE5A 723D              <1>       jc     short loc_delete_direntry_err_return
2504                            <1>
2505                            <1> loc_del_long_name:
2506 0000BE5C 0FB615[EE930100]  <1>       movzx  edx, byte [DelFile_LNEL]
2507 0000BE63 08D2              <1>       or     dl, dl
2508 0000BE65 7416              <1>       jz     short loc_del_dir_entry_update_parent_dir_lm_date
2509                            <1>
2510 0000BE67 8835[EE930100]    <1>       mov    byte [DelFile_LNEL], dh ; 0
2511                            <1>
2512 0000BE6D 0FB705[EC930100]  <1>       movzx  eax,  word [DelFile_EntryCounter]
2513 0000BE74 29D0              <1>       sub    eax, edx
2514                            <1>       ;jnc   short loc_del_long_name_continue
2515 0000BE76 7205              <1>       jc     short loc_del_dir_entry_update_parent_dir_lm_date
2516                            <1>
2517                            <1> ;loc_del_direntry_inv_data_return: ; 15/10/2016 (0Dh -> 29)
2518                            <1> ;     mov    eax, 29 ; 0Dh (TRDOS 8086) ; Invalid data
2519                            <1> ;     retn
2520                            <1>
2521                            <1> loc_del_long_name_continue:
2522                            <1>       ; AX = Directory Entry Number of the long name last entry
2523 0000BE78 E897FDFFFF          <1>       call   delete_longname
2524                            <1>       ;jc    short loc_delete_direntry_err_return
2525                            <1>
2526                            <1> loc_del_dir_entry_update_parent_dir_lm_date:
2527 0000BE7D 801D[EE930100]00  <1>       sbb    byte [DelFile_LNEL], 0 ; 0FFh if cf = 1
2528                            <1>
2529 0000BE84 E8C8FCFFFF        <1>       call   update_parent_dir_lmdt
2530 0000BE89 B700              <1>       mov    bh, 0
2531 0000BE8B 80D700            <1>       adc    bh, 0
2532                            <1>
2533 0000BE8E 8A1D[EE930100]    <1>       mov    bl, byte [DelFile_LNEL]
2534                            <1>
2535                            <1> loc_delete_direntry_return:
2536 0000BE94 A1[E8930100]      <1>       mov    eax, [DelFile_FCluster]
2537                            <1> loc_delete_direntry_err_return:
2538 0000BE99 C3                <1>       retn
2539                            <1>
2540                            <1> rename_directory_entry:
2541                            <1>       ; 13/11/2017
2542                            <1>       ; 15/10/2016
2543                            <1>       ; 06/03/2016 (TRDOS 386 = TRDOS v2.0)
2544                            <1>       ; 01/08/2011 (DIR.ASM, 'proc_rename_directory_entry')
2545                            <1>       ; 19/11/2010
2546                            <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>
0000BE9A 803D[858A0100]00       <1>       cmp    byte [Current_FATType], 0
0000BEA1 7706                   <1>       ja     short loc_rename_directory_entry
                                <1>
0000BEA3 E8B80E0000             <1>       call   rename_fs_file_or_directory
0000BEA8 C3                     <1>       retn
                                <1>
                                <1> loc_rename_directory_entry:
0000BEA9 881D[EE930100]         <1>       mov    [DelFile_LNEL], bl
0000BEAF 66890D[EC930100]       <1>       mov    [DelFile_EntryCounter], cx
0000BEB6 A3[E8930100]           <1>       mov    [DelFile_FCluster], eax
                                <1>
0000BEBB 0FB7C1                 <1>       movzx  eax, cx
0000BEBE E8BBFDFFFF             <1>       call   locate_current_dir_entry
0000BEC3 7308                   <1>       jnc    short loc_rename_direntry_check_fcluster
                                <1>
                                <1> loc_rename_direntry_pop_retn:
0000BEC5 C3                     <1>       retn
                                <1>
                                <1> loc_rename_direntry_pop_invd_retn:
0000BEC6 F9                     <1>       stc
                                <1> loc_rename_direntry_invd_retn:
                                <1>       ; 15/10/2016 (0Dh -> 29)
0000BEC7 B81D000000             <1>       mov    eax, 29 ; Invalid data
                                <1> loc_rename_retn:
0000BECC C3                     <1>       retn
                                <1>
                                <1> loc_rename_direntry_check_fcluster:
0000BECD 668B5714               <1>       mov    dx, [edi+20] ; First Cluster HW
0000BED1 C1E210                 <1>       shl    edx, 16 ; 13/11/2017
0000BED4 668B571A               <1>       mov    dx, [edi+26] ; First Cluster LW
0000BED8 3B15[E8930100]         <1>       cmp    edx, [DelFile_FCluster]
0000BEDE 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.)
0000BEE0 E8A2F6FFFF             <1>       call   convert_file_name
                                <1>
0000BEE5 C605[AC910100]02       <1>         mov     byte [DirBuff_ValidData], 2
0000BEEC E8C5FBFFFF             <1>       call   save_directory_buffer
0000BEF1 72D9                   <1>       jc     short loc_rename_retn
                                <1>
                                <1> loc_rename_direntry_del_ln:
0000BEF3 0FB615[EE930100]       <1>       movzx  edx, byte [DelFile_LNEL]
0000BEFA 08D2                   <1>       or     dl, dl
0000BEFC 7410                   <1>       jz     short loc_rename_direntry_update_parent_dir_lm_date
                                <1>
0000BEFE 0FB705[EC930100]       <1>       movzx  eax, word [DelFile_EntryCounter]
0000BF05 29D0                   <1>       sub    eax, edx
0000BF07 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
0000BF09 E806FDFFFF             <1>       call   delete_longname
                                <1>
                                <1> loc_rename_direntry_update_parent_dir_lm_date:
0000BF0E E83EFCFFFF             <1>       call   update_parent_dir_lmdt
0000BF13 31C0                   <1>       xor    eax, eax
0000BF15 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>
0000BF16 3C02           <1>       cmp    al, 2
0000BF18 0F847F010000   <1>       je     msftdf_df2_check_directory
0000BF1E A2[6E950100]   <1>       mov    [move_cmd_phase], al
                        <1>
                        <1> msftdf_parse_sf_path:
                        <1>       ; ESI = ASCIIZ pathname (Source)
0000BF23 57             <1>       push   edi
0000BF24 BF[6C940100]   <1>       mov    edi, SourceFile_Drv
0000BF29 E824F7FFFF     <1>       call   parse_path_name
0000BF2E 5E             <1>       pop    esi
0000BF2F 7211           <1>       jc     short msftdf_psf_retn
                        <1>
                        <1> msftdf_parse_df_path:
                        <1>       ; ESI = ASCIIZ pathname    (Destination)
0000BF31 BF[EC940100]   <1>       mov    edi, DestinationFile_Drv
0000BF36 E817F7FFFF     <1>       call   parse_path_name
0000BF3B 7306           <1>       jnc    short msftdf_check_sf_drv
                        <1>
0000BF3D 3C01           <1>       cmp    al, 1 ; File or directory name is not existing
0000BF3F 7602           <1>       jna    short msftdf_check_sf_drv
                        <1>
                        <1> msftdf_stc_retn:
0000BF41 F9             <1>       stc
                        <1> msftdf_psf_retn:
0000BF42 C3             <1>       retn
                        <1>
                        <1> msftdf_check_sf_drv:
0000BF43 A0[6C940100]   <1>       mov    al, [SourceFile_Drv]
                        <1>
                        <1> msftdf_check_df_drv:
0000BF48 8A15[EC940100] <1>       mov    dl, [DestinationFile_Drv]
                        <1>
                        <1> msftdf_compare_sf_df_drv:
0000BF4E 29DB           <1>       sub    ebx, ebx
0000BF50 8A3D[868A0100] <1>       mov    bh, [Current_Drv]
0000BF56 38C2           <1>       cmp    dl, al
0000BF58 7409           <1>       je     short msftdf_check_sf_df_drv_ok
                        <1>
                        <1> msftdf_not_same_drv:
                        <1>         ; DL = source file's drive number
0000BF5A 88C6           <1>       mov    dh, al ; destination file's drive number
                        <1>       ; 15/10/2016 (11h -> 21)
0000BF5C B815000000     <1>       mov    eax, 21 ; Not the same drive
0000BF61 F9             <1>       stc
0000BF62 C3             <1>       retn
                        <1>
                        <1> msftdf_check_sf_df_drv_ok:
0000BF63 8815[6F950100] <1>       mov    [msftdf_sf_df_drv], dl
                        <1>
0000BF69 29C0           <1>         sub  eax, eax
0000BF6B 88D4           <1>       mov    ah, dl
0000BF6D 0500010900     <1>       add    eax, Logical_DOSDisks
0000BF72 A3[70950100]   <1>       mov    [msftdf_drv_offset], eax
                        <1>
0000BF77 38FA           <1>       cmp    dl, bh ; byte [Current_Drv]
0000BF79 7407           <1>       je     short msftdf_df_check_directory
                        <1>
                        <1> msftdf_change_drv:
0000BF7B E85EC1FFFF     <1>       call   change_current_drive
0000BF80 726D           <1>       jc     short msftdf_df_error_retn
                        <1>
                        <1> msftdf_check_destination_file:
                        <1> msftdf_df_check_directory:
0000BF82 BE[ED940100]   <1>       mov    esi, DestinationFile_Directory
0000BF87 803E20         <1>       cmp    byte [esi], 20h
0000BF8A 760F           <1>       jna    short msftdf_df_find_1
                        <1>
                        <1> msftdf_df_change_directory:
0000BF8C FE05[51400100] <1>       inc    byte [Restore_CDIR]
0000BF92 30E4           <1>       xor    ah, ah ; CD_COMMAND sign -> 0
0000BF94 E8A3F0FFFF     <1>       call   change_current_directory
0000BF99 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:
0000BF9B BE[2E950100]   <1>         mov     esi, DestinationFile_Name
0000BFA0 803E20         <1>       cmp    byte [esi], 20h
0000BFA3 7631           <1>       jna    short msftdf_df_copy_sf_name
                        <1>
                        <1> msftdf_df_find_2:
0000BFA5 6631C0         <1>       xor    ax, ax ; DestinationFile_AttributesMask -> any/zero
0000BFA8 E8D4D4FFFF     <1>       call   find_first_file
0000BFAD 0F838D000000   <1>       jnc    msftdf_permission_denied_retn
                        <1>
                        <1> msftdf_df_check_error_code:
                        <1>       ;cmp   eax, 2 ; File not found error
0000BFB3 3C02           <1>       cmp    al, 2
0000BFB5 7537           <1>       jne    short msftdf_df_stc_retn
                        <1>
                        <1> msftdf_df_check_fname:
                        <1>       ; 15/10/2016
0000BFB7 BE[2E950100]   <1>       mov    esi, DestinationFile_Name ; *
0000BFBC E87ED8FFFF     <1>       call   check_filename
```

```
2757 0000BFC1 7307                <1>       jnc    short msftdf_convert_df_direntry_name
2758                              <1>       ; invalid file name chars !
2759 0000BFC3 B81A000000          <1>       mov    eax, ERR_INV_FILE_NAME  ; 26
2760 0000BFC8 EB24                <1>       jmp    short msftdf_df_stc_retn
2761                              <1>
2762                              <1> msftdf_convert_df_direntry_name:
2763                              <1>       ; mov  esi, DestinationFile_Name ; *
2764 0000BFCA BF[3E950100]        <1>       mov    edi, DestinationFile_DirEntry
2765 0000BFCF E8B3F5FFFF          <1>       call   convert_file_name
2766 0000BFD4 EB1A                <1>       jmp    short msftdf_restore_current_dir_1
2767                              <1>
2768                              <1> msftdf_df_copy_sf_name:
2769 0000BFD6 89F7                <1>       mov    edi, esi
2770 0000BFD8 57                  <1>       push   edi
2771 0000BFD9 BE[AE940100]        <1>         mov     esi, SourceFile_Name
2772 0000BFDE B90C000000          <1>       mov    ecx, 12
2773                              <1> msftdf_df_copy_sf_name_loop:
2774 0000BFE3 AC                  <1>       lodsb
2775 0000BFE4 AA                  <1>         stosb
2776 0000BFE5 08C0                <1>       or     al, al
2777 0000BFE7 7402                <1>       jz     short msftdf_df_copy_sf_name_ok
2778 0000BFE9 E2F8                <1>         loop    msftdf_df_copy_sf_name_loop
2779                              <1> msftdf_df_copy_sf_name_ok:
2780 0000BFEB 5E                  <1>       pop    esi
2781 0000BFEC EBB7                <1>       jmp    short msftdf_df_find_2
2782                              <1>
2783                              <1> msftdf_df_stc_retn:
2784 0000BFEE F9                  <1>       stc
2785                              <1> msftdf_restore_cdir_failed:
2786                              <1> msftdf_df_error_retn:
2787 0000BFEF C3                  <1>       retn
2788                              <1>
2789                              <1> msftdf_restore_current_dir_1:
2790 0000BFF0 803D[51400100]00    <1>       cmp    byte [Restore_CDIR], 0
2791 0000BFF7 760D                <1>       jna    short msftdf_sf_check_directory
2792 0000BFF9 8B35[70950100]      <1>       mov    esi, [msftdf_drv_offset]
2793 0000BFFF E891C1FFFF          <1>       call   restore_current_directory
2794 0000C004 72E9                <1>       jc     short msftdf_restore_cdir_failed
2795                              <1>
2796                              <1> msftdf_sf_check_directory:
2797 0000C006 BE[6D940100]        <1>       mov    esi, SourceFile_Directory
2798 0000C00B 803E20              <1>       cmp    byte [esi], 20h
2799 0000C00E 760F                <1>       jna    short msftdf_sf_find
2800                              <1> msftdf_sf_change_directory:
2801 0000C010 FE05[51400100]      <1>       inc    byte [Restore_CDIR]
2802 0000C016 30E4                <1>       xor    ah, ah ; CD_COMMAND sign -> 0
2803 0000C018 E81FF0FFFF          <1>       call   change_current_directory
2804 0000C01D 7227                <1>       jc     short msftdf_return
2805                              <1>
2806                              <1> ;msftdf_sf_change_prompt_dir_string:
2807                              <1> ;     call   change_prompt_dir_string
2808                              <1>
2809                              <1> msftdf_sf_find:
2810 0000C01F BE[AE940100]        <1>         mov     esi, SourceFile_Name  ; Offset 66
2811 0000C024 66B80018            <1>       mov    ax, 1800h ; Only files
2812 0000C028 E854D4FFFF          <1>       call   find_first_file
2813 0000C02D 7217                <1>       jc     short msftdf_return
2814                              <1>
2815                              <1> msftdf_sf_ambgfn_check:
2816 0000C02F 6609D2              <1>       or     dx, dx ; Ambiguous filename chars used sign (DX>0)
2817 0000C032 7407                <1>       jz     short msftdf_sf_found
2818                              <1>
2819                              <1> msftdf_ambiguous_file_name_error:
2820 0000C034 B802000000          <1>       mov    eax, 2 ; File not found error
2821 0000C039 F9                  <1>       stc
2822 0000C03A C3                  <1>       retn
2823                              <1>
2824                              <1> msftdf_sf_found:
2825 0000C03B 80E31F              <1>       and    bl, 1Fh ; Attributes, D-V-S-H-R
2826 0000C03E 7416                <1>       jz     short msftdf_save_sf_structure
2827                              <1>
2828                              <1> msftdf_permission_denied_retn:
2829 0000C040 B805000000          <1>       mov    eax, 05h ; Access (Permission) denied !
2830 0000C045 F9                  <1>       stc
2831                              <1> msftdf_rest_cdir_err_retn:
2832                              <1> msftdf_return:
2833 0000C046 C3                  <1>       retn
2834                              <1>
2835                              <1> msftdf_phase_1_return:
2836 0000C047 31C0                <1>       xor    eax, eax
2837 0000C049 A2[6E950100]        <1>       mov    [move_cmd_phase], al ; 0
2838 0000C04E FEC0                <1>       inc    al ; mov al, 1
2839 0000C050 BB[9DC00000]        <1>       mov    ebx, msftdf_df2_check_directory
2840                              <1>       ;mov   edx, 0FFFFFFFFh
2841 0000C055 C3                  <1>       retn
2842                              <1>
2843                              <1> msftdf_save_sf_structure:
2844 0000C056 BE[78930100]        <1>       mov    esi, FindFile_DirEntry
2845 0000C05B BF[BE940100]        <1>       mov    edi, SourceFile_DirEntry
2846 0000C060 B908000000          <1>       mov    ecx, 8
2847 0000C065 F3A5                <1>       rep    movsd
2848                              <1>
2849                              <1> msftdf_df_copy_sf_parameters:
2850 0000C067 BE0B000000          <1>       mov    esi, 11
2851 0000C06C 89F7                <1>       mov    edi, esi
2852 0000C06E 81C6[BE940100]      <1>       add    esi, SourceFile_DirEntry
2853 0000C074 81C7[3E950100]      <1>       add    edi, DestinationFile_DirEntry
2854                              <1>       ;mov   ecx, 21
2855 0000C07A B115                <1>       mov    cl, 21
2856 0000C07C F3A4                <1>       rep    movsb
2857                              <1>
2858                              <1> msftdf_restore_current_dir_2:
2859 0000C07E 803D[51400100]00    <1>       cmp    byte [Restore_CDIR], 0
2860 0000C085 760D                <1>       jna    short msftdf_df2_check_move_cmd_phase
2861 0000C087 8B35[70950100]      <1>       mov    esi, [msftdf_drv_offset]
```

```
2862 0000C08D E803C1FFFF          <1>       call   restore_current_directory
2863 0000C092 72B2                <1>       jc     short msftdf_rest_cdir_err_retn
2864                              <1>
2865                              <1> msftdf_df2_check_move_cmd_phase:
2866 0000C094 803D[6E950100]01    <1>       cmp    byte [move_cmd_phase], 1
2867 0000C09B 74AA                <1>       je     short msftdf_phase_1_return
2868                              <1>
2869                              <1> msftdf_df2_check_directory:
2870 0000C09D BE[ED940100]        <1>       mov    esi, DestinationFile_Directory
2871 0000C0A2 803E20              <1>       cmp    byte [esi], 20h
2872 0000C0A5 760F                <1>       jna    short msftdf_make_dfde_locate_ffe_on_directory
2873                              <1> msftdf_df2_change_directory:
2874 0000C0A7 FE05[51400100]      <1>       inc    byte [Restore_CDIR]
2875 0000C0AD 30E4                <1>       xor    ah, ah ; CD_COMMAND sign -> 0
2876 0000C0AF E888EFFFFF          <1>       call   change_current_directory
2877 0000C0B4 7290                <1>       jc     short msftdf_return
2878                              <1>
2879                              <1> ;msftdf_df2_change_prompt_dir_string:
2880                              <1> ;     call   change_prompt_dir_string
2881                              <1>
2882                              <1> msftdf_make_dfde_locate_ffe_on_directory:
2883                              <1>       ; Current directory fcluster <> Directory buffer cluster
2884                              <1>       ; Current directory will be reloaded by
2885                              <1>       ; 'locate_current_dir_file' procedure
2886                              <1>       ;
2887                              <1>       ;xor   ax, ax
2888 0000C0B6 31C0                <1>       xor    eax, eax
2889 0000C0B8 89C1                <1>       mov    ecx, eax
2890 0000C0BA 6649                <1>       dec    cx ; FFFFh
2891                              <1>              ; CX = FFFFh -> find first deleted or free entry
2892                              <1>              ; ESI would be ASCIIZ filename address if the call
2893                              <1>              ; would not be for first free or deleted dir entry
2894 0000C0BC E8CFF1FFFF          <1>       call   locate_current_dir_file
2895 0000C0C1 733F                <1>       jnc    msftdf_make_dfde_set_ff_dir_entry
2896                              <1>
2897                              <1>       ;cmp   eax, 2
2898 0000C0C3 3C02                <1>         cmp  al, 2
2899 0000C0C5 7537                <1>       jne    short msftdf_error_retn
2900                              <1>
2901                              <1> msftdf_add_new_dir_entry_check_fs:
2902 0000C0C7 8B35[70950100]      <1>       mov    esi, [msftdf_drv_offset]
2903 0000C0CD A1[B1910100]        <1>       mov    eax, [DirBuff_Cluster]
2904 0000C0D2 807E0300            <1>       cmp    byte [esi+LD_FATType], 0
2905 0000C0D6 7711                <1>       ja     short msftdf_add_new_subdir_cluster
2906                              <1>
2907                              <1> msftdf_add_new_fs_subdir_section:
2908                              <1>       ;CL=0, CH=E5h --> deleted entry, CH=0 --> free entry
2909                              <1>         ;xor cx, cx
2910 0000C0D8 30ED                <1>       xor    ch, ch ; cx = 0 --> add a new subdir section
2911 0000C0DA E8830C0000          <1>       call   add_new_fs_section
2912 0000C0DF 721E                <1>         jc   short msftdf_dsfde_error_retn
2913                              <1>       ;mov   [createfile_LastDirCluster], eax
2914                              <1>
2915 0000C0E1 E8A30E0000          <1>       call   load_FS_sub_directory
2916                              <1>       ;mov   ebx, Directory_Buffer
2917 0000C0E6 7318                <1>       jnc    short msftdf_add_new_fs_subdir_section_ok
2918 0000C0E8 C3                  <1>       retn
2919                              <1>
2920                              <1> msftdf_add_new_subdir_cluster:
2921 0000C0E9 E881150000          <1>       call   add_new_cluster
2922 0000C0EE 720F                <1>       jc     short msftdf_dsfde_error_retn
2923                              <1>
2924                              <1>       ;mov   [createfile_LastDirCluster], eax
2925                              <1>
2926 0000C0F0 E8570E0000          <1>       call   load_FAT_sub_directory
2927 0000C0F5 7309                <1>       jnc    short msftdf_add_new_subdir_cluster_ok
2928                              <1>       ; EBX = Directory buffer address
2929                              <1>
2930                              <1> msftdf_ansdc_update_parent_dir_lmdt:
2931                              <1> msftdf_make_dfde_err_upd_pdir_lmdt:
2932 0000C0F7 50                  <1>       push   eax
2933 0000C0F8 E854FAFFFF          <1>       call   update_parent_dir_lmdt
2934 0000C0FD 58                  <1>       pop    eax
2935                              <1>
2936                              <1> msftdf_error_retn:
2937 0000C0FE F9                  <1>       stc
2938                              <1> msftdf_dsfde_restore_cdir_failed:
2939                              <1> msftdf_dsfde_error_retn:
2940 0000C0FF C3                  <1>       retn
2941                              <1>
2942                              <1> msftdf_add_new_fs_subdir_section_ok:
2943                              <1> msftdf_add_new_subdir_cluster_ok:
2944 0000C100 89DF                <1>       mov    edi, ebx ; Directory buffer address
2945                              <1>
2946                              <1> msftdf_make_dfde_set_ff_dir_entry:
2947 0000C102 8B15[808A0100]      <1>       mov    edx, [Current_Dir_FCluster]
2948 0000C108 8915[D4950100]      <1>       mov    [createfile_FFCluster], edx
2949                              <1>       ; EDI = Directory entry offset
2950 0000C10E BE[3E950100]        <1>       mov    esi, DestinationFile_DirEntry
2951 0000C113 B908000000          <1>       mov    ecx, 8
2952 0000C118 F3A5                <1>       rep    movsd
2953                              <1>
2954 0000C11A C605[AC910100]02    <1>       mov    byte [DirBuff_ValidData], 2
2955 0000C121 E890F9FFFF          <1>       call   save_directory_buffer
2956 0000C126 72CF                <1>       jc     short msftdf_make_dfde_err_upd_pdir_lmdt
2957                              <1>
2958                              <1> msftdf_make_dfde_update_pdir_lmdt:
2959 0000C128 E824FAFFFF          <1>       call   update_parent_dir_lmdt
2960                              <1>
2961                              <1> msftdf_dsfde_restore_current_dir_1:
2962 0000C12D 803D[51400100]00    <1>       cmp    byte [Restore_CDIR], 0
2963 0000C134 760D                <1>       jna    short msftdf_dsfde_check_directory
2964 0000C136 8B35[70950100]      <1>       mov    esi, [msftdf_drv_offset]
2965 0000C13C E854C0FFFF          <1>       call   restore_current_directory
2966 0000C141 72BC                <1>       jc     short msftdf_dsfde_restore_cdir_failed
```

```
                                <1>
                                <1> msftdf_dsfde_check_directory:
0000C143 BE[6D940100]           <1>         mov    esi, SourceFile_Directory
0000C148 803E20                 <1>         cmp    byte [esi], 20h
0000C14B 760F                   <1>         jna    short msftdf_dsfde_find_file
                                <1>
                                <1> msftdf_dsfde_change_directory:
0000C14D FE05[51400100]         <1>         inc    byte [Restore_CDIR]
0000C153 28E4                   <1>         sub    ah, ah ; CD_COMMAND sign -> 0
0000C155 E8E2EEFFFF             <1>         call   change_current_directory
0000C15A 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:
0000C15C BE[AE940100]           <1>         mov    esi, SourceFile_Name  ; Offset 66
0000C161 668B460E               <1>         mov    ax, [esi+14] ; 80 -> SourceFile_AttributesMask
0000C165 E817D3FFFF             <1>         call   find_first_file
0000C16A 7293                   <1>         jc     short msftdf_dsfde_error_retn
                                <1>
                                <1> msftdf_dsfde_delete_direntry:
0000C16C 8B35[70950100]         <1>         mov    esi, [msftdf_drv_offset]
                                <1>
0000C172 807E0300               <1>         cmp    byte [esi+LD_FATType], 0
0000C176 770A                   <1>         ja     short msftdf_delete_FAT_direntry
                                <1>
0000C178 30DB                   <1>         xor    bl, bl
                                <1>         ; BL = 0 -> File
                                <1>         ; EDI -> Directory buffer entry offset/address
0000C17A E8E40B0000             <1>         call   delete_fs_directory_entry
0000C17F 7315                   <1>         jnc    short msftdf_dsfde_restore_current_dir_2
0000C181 C3                     <1>         retn
                                <1>
                                <1> msftdf_delete_FAT_direntry:
0000C182 8A1D[75930100]         <1>         mov    bl, [FindFile_LongNameEntryLength]
0000C188 668B0D[A0930100]       <1>         mov    cx, [FindFile_DirEntryNumber]
                                <1>         ; ESI = Logical DOS drive description table address
                                <1>         ; EDI = Directory buffer entry offset/address
0000C18F E89AFCFFFF             <1>         call   delete_directory_entry
0000C194 721C                   <1>         jc     short msftdf_retn
                                <1>
                                <1> msftdf_dsfde_restore_current_dir_2:
0000C196 803D[51400100]00       <1>         cmp    byte [Restore_CDIR], 0
0000C19D 7607                   <1>         jna    short msftdf_new_dir_fcluster_retn
                                <1>         ;mov   esi, [msftdf_drv_offset]
0000C19F E8F1BFFFFF             <1>         call   restore_current_directory
0000C1A4 720C                   <1>         jc     short msftdf_retn
                                <1>
                                <1> msftdf_new_dir_fcluster_retn:
0000C1A6 31C9                   <1>         xor    ecx, ecx
0000C1A8 A1[D4950100]           <1>         mov    eax, [createfile_FFCluster]
0000C1AD BB[EC940100]           <1>         mov    ebx, DestinationFile_Drv
                                <1>
                                <1> msftdf_retn:
0000C1B2 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 ->
```

```
3072                                 <1>       ;      cf = 0 -> OK
3073                                 <1>       ;      EAX = Destination file first cluster
3074                                 <1>       ;
3075                                 <1>       ;        CL > 0 if there is file reading error before EOF
3076                                 <1>       ;             (incomplete copy)
3077                                 <1>       ;        CH > 0 if file is (full) loaded at memory
3078                                 <1>       ;
3079                                 <1>       ;      cf = 1 -> Error code in AL (EAX)
3080                                 <1>       ;
3081                                 <1>       ; (EBX, ECX, ESI, EDI register contents will be changed)
3082                                 <1>
3083                                 <1>
3084 0000C1B3 3C02                  <1>       cmp    al, 2
3085 0000C1B5 0F845A020000          <1>       je     csftdf2_check_cdrv
3086                                 <1>
3087                                 <1> ; Phase 1
3088                                 <1>
3089 0000C1BB A2[94950100]          <1>       mov    byte [copy_cmd_phase], al
3090                                 <1>
3091 0000C1C0 57                    <1>       push   edi ; *
3092                                 <1>
3093                                 <1> csftdf_parse_sf_path:
3094 0000C1C1 BF[6C940100]          <1>       mov    edi, SourceFile_Drv
3095 0000C1C6 E887F4FFFF            <1>       call   parse_path_name
3096 0000C1CB 721C                  <1>       jc     short csftdf_parse_sf_path_failed
3097                                 <1>
3098                                 <1> csftdf_parse_df_path:
3099 0000C1CD 5E                    <1>       pop    esi ; * (pushed edi)
3100                                 <1>
3101                                 <1> csftdf_sf_check_filename_exists:
3102 0000C1CE 803D[AE940100]21      <1>       cmp    byte [SourceFile_Name], 21h
3103 0000C1D5 7215                  <1>       jb     short csftdf_sf_file_not_found_error
3104                                 <1>
3105 0000C1D7 BF[EC940100]          <1>       mov    edi, DestinationFile_Drv
3106 0000C1DC E871F4FFFF            <1>       call   parse_path_name
3107 0000C1E1 7310                  <1>       jnc    short csftdf_check_sf_cdrv
3108                                 <1>
3109 0000C1E3 3C01                  <1>       cmp    al, 1 ; File or directory name is not existing
3110 0000C1E5 760C                  <1>       jna    short csftdf_check_sf_cdrv
3111                                 <1>
3112                                 <1> csftdf_parse_df_path_failed:
3113 0000C1E7 F9                    <1>       stc
3114                                 <1> csftdf_sf_error_retn:
3115 0000C1E8 C3                    <1>       retn
3116                                 <1>
3117                                 <1> csftdf_parse_sf_path_failed:
3118 0000C1E9 5F                    <1>       pop    edi ; *
3119 0000C1EA EBFC                  <1>       jmp    short csftdf_sf_error_retn
3120                                 <1>
3121                                 <1> csftdf_sf_file_not_found_error:
3122 0000C1EC B802000000            <1>       mov    eax, 2 ; File not found
3123 0000C1F1 EBF5                  <1>       jmp    short csftdf_sf_error_retn
3124                                 <1>
3125                                 <1> csftdf_check_sf_cdrv:
3126 0000C1F3 8A3D[868A0100]        <1>       mov    bh, [Current_Drv]
3127                                 <1>
3128 0000C1F9 883D[97950100]        <1>       mov    [csftdf_cdrv], bh ; 23/03/2016
3129                                 <1>
3130 0000C1FF 8A15[6C940100]        <1>       mov    dl, [SourceFile_Drv]
3131 0000C205 38FA                  <1>       cmp    dl, bh ; byte [Current_Drv]
3132 0000C207 7407                  <1>       je     short csftdf_sf_check_directory
3133                                 <1>
3134 0000C209 E8D0BEFFFF            <1>       call   change_current_drive
3135 0000C20E 72D8                  <1>       jc     short csftdf_sf_error_retn
3136                                 <1>
3137                                 <1> csftdf_sf_check_directory:
3138 0000C210 BE[6D940100]          <1>       mov    esi, SourceFile_Directory
3139 0000C215 803E20                <1>       cmp    byte [esi], 20h
3140 0000C218 760F                  <1>       jna    short csftdf_find_sf
3141                                 <1>
3142                                 <1> csftdf_sf_change_directory:
3143 0000C21A FE05[51400100]        <1>       inc    byte [Restore_CDIR]
3144 0000C220 30E4                  <1>       xor    ah, ah ; CD_COMMAND sign -> 0
3145 0000C222 E815EEFFFF            <1>       call   change_current_directory
3146 0000C227 72BF                  <1>       jc     short csftdf_sf_error_retn
3147                                 <1>
3148                                 <1> ;csftdf_sf_change_prompt_dir_string:
3149                                 <1> ;     call   change_prompt_dir_string
3150                                 <1>
3151                                 <1> csftdf_find_sf:
3152 0000C229 BE[AE940100]          <1>       mov    esi, SourceFile_Name
3153 0000C22E 66B80018              <1>       mov    ax, 1800h ; Except volume label and dirs
3154 0000C232 E84AD2FFFF            <1>       call   find_first_file
3155 0000C237 72AF                  <1>       jc     short csftdf_sf_error_retn
3156                                 <1>
3157                                 <1> csftdf_sf_ambgfn_check:
3158 0000C239 6621D2                <1>       and    dx, dx ; Ambiguous filename chars used sign (DX>0)
3159 0000C23C 7407                  <1>       jz     short csftdf_sf_found
3160                                 <1>
3161                                 <1> csftdf_ambiguous_file_name_error:
3162 0000C23E B802000000            <1>       mov    eax, 2 ; File not found error
3163 0000C243 F9                    <1>       stc
3164 0000C244 C3                    <1>       retn
3165                                 <1>
3166                                 <1> csftdf_sf_found:
3167 0000C245 A3[98950100]          <1>       mov    [csftdf_filesize], eax
3168                                 <1>
3169 0000C24A 09C0                  <1>       or     eax, eax
3170 0000C24C 7507                  <1>       jnz    short csftdf_set_source_file_direntry
3171                                 <1>
3172                                 <1> csftdf_sf_file_size_zero:
3173 0000C24E B814000000            <1>       mov    eax, 20 ; TRDOS zero length (file size) error
3174 0000C253 F9                    <1>       stc
3175 0000C254 C3                    <1>       retn
3176                                 <1>
```

```
                                <1> csftdf_set_source_file_direntry:
0000C255 BE[78930100]           <1>         mov     esi, FindFile_DirEntry
0000C25A BF[BE940100]           <1>         mov     edi, SourceFile_DirEntry
0000C25F B908000000             <1>         mov     ecx, 8
0000C264 F3A5                   <1>         rep     movsd
                                <1>
                                <1> csftdf_sf_restore_cdrv:
                                <1>         ; 22/03/2016
0000C266 8A15[97950100]         <1>         mov     dl, [csftdf_cdrv]
0000C26C 3A15[868A0100]         <1>         cmp     dl, [Current_Drv]
0000C272 7407                   <1>         je      short csftdf_sf_restore_cdir
0000C274 E865BEFFFF             <1>         call    change_current_drive
0000C279 724F                   <1>         jc      short csftdf_df_error_retn ; 30/03/2016
                                <1>
                                <1> csftdf_sf_restore_cdir:
0000C27B 803D[51400100]00       <1>         cmp     byte [Restore_CDIR], 0
0000C282 7612                   <1>         jna     short csftdf_df_check_filename_exists
0000C284 29C0                   <1>         sub     eax, eax
0000C286 BE00010900             <1>         mov     esi, Logical_DOSDisks
0000C28B 88D4                   <1>         mov     ah, dl ; byte [csftdf_cdrv]
0000C28D 01C6                   <1>         add     esi, eax
0000C28F E801BFFFFF             <1>         call    restore_current_directory
0000C294 7234                   <1>         jc      short csftdf_df_error_retn
                                <1>
                                <1> csftdf_df_check_filename_exists:
0000C296 803D[2E950100]20       <1>         cmp     byte [DestinationFile_Name], 20h
0000C29D 7716                   <1>         ja      short csftdf_check_df_cdrv
                                <1>
                                <1> csftdf_copy_sf_name:
0000C29F BF[2E950100]           <1>         mov     edi, DestinationFile_Name
0000C2A4 BE[AE940100]           <1>         mov     esi, SourceFile_Name
0000C2A9 B10C                   <1>         mov     cl, 12
                                <1>
                                <1> csftdf_df_copy_sf_name_loop:
0000C2AB AC                     <1>         lodsb
0000C2AC AA                     <1>         stosb
0000C2AD 08C0                   <1>         or      al, al
0000C2AF 7404                   <1>         jz      short csftdf_check_df_cdrv
0000C2B1 FEC9                   <1>         dec     cl
0000C2B3 75F6                   <1>         jnz     csftdf_df_copy_sf_name_loop
                                <1>
                                <1> csftdf_check_df_cdrv:
0000C2B5 8A15[EC940100]         <1>         mov     dl, [DestinationFile_Drv]
0000C2BB 3A15[868A0100]         <1>         cmp     dl, [Current_Drv]
0000C2C1 7408                   <1>         je      short csftdf_df_check_directory
                                <1>
0000C2C3 E816BEFFFF             <1>         call    change_current_drive
0000C2C8 7301                   <1>         jnc     short csftdf_df_check_directory
                                <1>
                                <1> csftdf_df_error_retn:
0000C2CA C3                     <1>         retn
                                <1>
                                <1> csftdf_df_check_directory:
0000C2CB BE[ED940100]           <1>         mov     esi, DestinationFile_Directory
0000C2D0 803E20                 <1>           cmp     byte [esi], 20h
0000C2D3 760F                   <1>         jna     short csftdf_find_df
                                <1>
                                <1> csftdf_df_change_directory:
0000C2D5 FE05[51400100]         <1>         inc     byte [Restore_CDIR]
0000C2DB 28E4                   <1>         sub     ah, ah ; CD_COMMAND sign -> 0
0000C2DD E85AEDFFFF             <1>         call    change_current_directory
0000C2E2 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
0000C2E4 29DB                   <1>         sub     ebx, ebx
0000C2E6 8A3D[EC940100]         <1>         mov     bh,  [DestinationFile_Drv]
0000C2EC 81C300010900           <1>         add     ebx, Logical_DOSDisks
0000C2F2 891D[C4950100]         <1>         mov     [csftdf_df_drv_dt], ebx
                                <1>
0000C2F8 BE[2E950100]           <1>         mov     esi, DestinationFile_Name
0000C2FD 6631C0                 <1>         xor     ax, ax
                                <1>                 ; DestinationFile_AttributesMask -> any/zero
0000C300 E87CD1FFFF             <1>         call    find_first_file
0000C305 7218                   <1>         jc      short csftdf_df_check_error_code
                                <1>
                                <1> csftdf_df_ambgfn_check:
0000C307 6609D2                 <1>         or      dx, dx ; Ambiguous filename chars used sign (DX>0)
0000C30A 752A                   <1>         jnz     short csftdf_df_error_inv_fname
                                <1>
                                <1> csftdf_df_found:
0000C30C C605[96950100]01       <1>         mov     byte [DestinationFileFound], 1
                                <1>         ; 17/10/2016 (cl -> bl)
0000C313 80E31F                 <1>         and     bl, 1Fh ; Attributes, D-V-S-H-R
0000C316 745F                   <1>         jz      short csftdf_df_save_first_cluster
                                <1>
                                <1> csftdf_df_permission_denied_retn:
0000C318 B805000000             <1>         mov     eax, 05h ; Access/Permission denied.
                                <1> csftdf_df_error_stc_retn:
0000C31D F9                     <1>         stc
0000C31E C3                     <1>         retn
                                <1>
                                <1> csftdf_df_check_error_code:
                                <1>         ;cmp    eax, 2
0000C31F 3C02                   <1>         cmp     al, 2
0000C321 75FA                   <1>         jne     short csftdf_df_error_stc_retn
                                <1>
0000C323 C605[96950100]00       <1>         mov     byte [DestinationFileFound], 0
                                <1>
                                <1>         ; 15/10/2016
0000C32A BE[68930100]           <1>         mov     esi, FindFile_Name ; *
0000C32F E80BD5FFFF             <1>         call    check_filename
```

```
3282 0000C334 7307              <1>       jnc    short csftdf_df_valid_fname
3283                            <1> csftdf_df_error_inv_fname: ; 'invalid file name !'
3284 0000C336 B81A000000        <1>       mov    eax, ERR_INV_FILE_NAME  ; 26
3285 0000C33B F9                <1>       stc
3286 0000C33C C3                <1>       retn
3287                            <1>
3288                            <1> csftdf_df_valid_fname:
3289                            <1>       ; 21/03/2016
3290                            <1>       ; (Capitalized file name)
3291                            <1>       ;mov   esi, FindFile_Name ; * ; 15/10/2016
3292 0000C33D BF[2E950100]      <1>       mov    edi, DestinationFile_Name
3293 0000C342 A5                <1>       movsd
3294 0000C343 A5                <1>       movsd
3295 0000C344 A5                <1>       movsd
3296                            <1>       ;movsb
3297                            <1>
3298                            <1> csftdf_check_disk_free_size_0:
3299 0000C345 A1[DA940100]      <1>       mov    eax, [SourceFile_DirEntry+DirEntry_FileSize]
3300                            <1>
3301                            <1> csftdf_check_disk_free_size_1:
3302                            <1>       ;sub   ebx, ebx
3303                            <1>       ;mov   esi, Logical_DOSDisks
3304                            <1>       ;mov   bh,  [DestinationFile_Drv]
3305                            <1>       ;add   esi, ebx
3306                            <1>
3307 0000C34A 8B35[C4950100]    <1>       mov    esi, [csftdf_df_drv_dt] ; 23/03/2016
3308                            <1>
3309 0000C350 0FB74E11          <1>       movzx  ecx, word [esi+LD_BPB+BytesPerSec] ; 17, LD_BPB + 0Bh
3310 0000C354 01C8              <1>       add    eax, ecx
3311 0000C356 48                <1>       dec    eax  ; file size (additional bytes) + 511 (round up)
3312                            <1> csftdf_check_disk_free_size_3: ; 16/03/2016
3313 0000C357 29D2              <1>       sub    edx, edx
3314 0000C359 F7F1              <1>       div    ecx ; bytes per sector
3315                            <1>
3316                            <1> csftdf_check_disk_free_size:
3317 0000C35B 3B4674            <1>       cmp    eax, [esi+LD_FreeSectors]
3318 0000C35E 0F8294000000      <1>         jb      csftdf_check_disk_free_size_ok
3319 0000C364 770A              <1>       ja     short csftdf_df_insufficient_disk_space
3320                            <1>
3321 0000C366 807E0300          <1>       cmp    byte [esi+LD_FATType], 0 ; FS needs FDT sector also.
3322 0000C36A 0F8788000000      <1>         ja      csftdf_check_disk_free_size_ok
3323                            <1>
3324                            <1> csftdf_df_insufficient_disk_space:
3325 0000C370 B827000000        <1>       mov    eax, 27h ; insufficient disk space
3326 0000C375 EBA6              <1>       jmp    short csftdf_df_error_stc_retn
3327                            <1>
3328                            <1> csftdf_df_save_first_cluster:
3329                            <1>       ; ESI = FindFile_DirEntry (for the old destination file)
3330                            <1>       ; EAX = Old destination file size
3331                            <1>       ; 24/03/2016
3332                            <1>       ; EDI = Directory entry address (within Dir Buffer boundaries)
3333 0000C377 81EF00000800        <1>       sub    edi, Directory_Buffer  ; (<65536)
3334 0000C37D 66C1EF05          <1>       shr    di, 5 ; Convert entry offset to entry index/number
3335 0000C381 66893D[66950100]  <1>       mov    [DestinationFile_DirEntryNumber], di ; (<2048)
3336                            <1>
3337                            <1> csftdf_df_check_sf_df_fcluster:
3338 0000C388 668B5614          <1>       mov    dx, [esi+DirEntry_FstClusHI]
3339 0000C38C C1E210            <1>       shl    edx, 16
3340 0000C38F 668B561A          <1>       mov    dx, [esi+DirEntry_FstClusLO]
3341 0000C393 8915[A8950100]    <1>       mov    [csftdf_df_cluster], edx
3342                            <1> csftdf_df_check_sf_df_fcluster_1:
3343 0000C399 668B15[D2940100]  <1>       mov    dx, [SourceFile_DirEntry+DirEntry_FstClusHI]
3344 0000C3A0 C1E210            <1>       shl    edx, 16
3345 0000C3A3 668B15[D8940100]  <1>       mov    dx, [SourceFile_DirEntry+DirEntry_FstClusLO]
3346 0000C3AA 3B15[A8950100]    <1>       cmp    edx, [csftdf_df_cluster]
3347 0000C3B0 7512              <1>       jne    short csftdf_df_check_sf_df_fcluster_ok
3348                            <1> csftdf_df_check_sf_df_drv:
3349 0000C3B2 8A15[6C940100]    <1>       mov    dl, [SourceFile_Drv]
3350 0000C3B8 3A15[EC940100]    <1>       cmp    dl, [DestinationFile_Drv]
3351 0000C3BE 7504              <1>       jne    short csftdf_df_check_sf_df_fcluster_ok
3352                            <1>
3353                            <1>       ; source and destination files are same !
3354                            <1>       ; (they have same first cluster value on same logical disk)
3355                            <1>
3356 0000C3C0 31C0              <1>       xor    eax, eax ; mov eax, 0 -> Bad command or file name !
3357 0000C3C2 F9                <1>       stc
3358 0000C3C3 C3                <1>       retn
3359                            <1>
3360                            <1> csftdf_df_check_sf_df_fcluster_ok:
3361                            <1> csftdf_df_move_findfile_struct:
3362                            <1>       ; mov  esi, FindFile_DirEntry
3363 0000C3C4 BF[3E950100]      <1>       mov    edi, DestinationFile_DirEntry
3364 0000C3C9 B908000000        <1>       mov    ecx, 8
3365 0000C3CE F3A5              <1>       rep    movsd
3366                            <1>
3367                            <1> csftdf_check_disk_free_size_2:
3368 0000C3D0 89C2              <1>       mov    edx, eax ; Old destination file size
3369                            <1>
3370                            <1>       ;mov   eax, [SourceFile_DirEntry+DirEntry_FileSize]
3371 0000C3D2 A1[98950100]      <1>       mov    eax, [csftdf_filesize] ; 23/03/2016
3372                            <1>
3373                            <1>       ;;sub  ecx, ecx ; 0
3374                            <1>       ;mov   esi, Logical_DOSDisks
3375                            <1>       ;mov   ch,  [DestinationFile_Drv]
3376                            <1>       ;add   esi, ecx
3377                            <1>       ;
3378                            <1>       ;mov   [csftdf_df_drv_dt], esi
3379                            <1>
3380 0000C3D7 8B35[C4950100]    <1>       mov    esi, [csftdf_df_drv_dt] ; 23/03/2016
3381                            <1>
3382 0000C3DD 668B4E11          <1>       mov    cx, [esi+LD_BPB+BytesPerSec] ; 17, LD_BPB + 0Bh
3383 0000C3E1 01CA              <1>       add    edx, ecx ; + 512
3384 0000C3E3 01C8              <1>       add    eax, ecx ; + 512
3385 0000C3E5 4A                <1>       dec    edx ; old file size + 511 (round up)
3386 0000C3E6 48                <1>       dec    eax ; new file size + 511 (round up)
```

```
3387 0000C3E7 F7D9              <1>       neg    ecx ; -512 ; 0FFFFFE00h
3388 0000C3E9 21CA              <1>       and    edx, ecx ; = old sector count * 512
3389 0000C3EB 21C8              <1>       and    eax, ecx ; = new sector count * 512
3390                            <1>
3391 0000C3ED 29D0              <1>       sub    eax, edx ; new file size - old file size (on disk)
3392 0000C3EF 7607              <1>       jna    short csftdf_check_disk_free_size_ok
3393                            <1>
3394 0000C3F1 F7D9              <1>       neg    ecx ; 512 (bytes per sector) ; 200h
3395                            <1>       ; check free space for additional sectors
3396                            <1>       ; eax = number of additional sectors * bytes per sector
3397                            <1>       ; esi = Logical DOS drive number (of destination disk)
3398 0000C3F3 E95FFFFFFF        <1>         jmp     csftdf_check_disk_free_size_3
3399                            <1>
3400                            <1> csftdf_check_disk_free_size_ok:
3401                            <1>       ; 18/03/2016
3402                            <1> csftdf_df_check_copy_cmd_phase:
3403 0000C3F8 A0[94950100]      <1>       mov    al, [copy_cmd_phase]
3404 0000C3FD 3C01              <1>       cmp    al, 1
3405 0000C3FF 7514              <1>       jne    short csftdf2_check_cdrv
3406                            <1>
3407 0000C401 31C0              <1>       xor    eax, eax
3408 0000C403 A2[94950100]      <1>       mov    [copy_cmd_phase], al ; 0
3409                            <1>
3410 0000C408 8A15[96950100]    <1>       mov    dl, [DestinationFileFound]
3411 0000C40E 8A35[C9940100]    <1>       mov    dh, [SourceFile_DirEntry+11] ; Attributes
3412                            <1>
3413                            <1> csftdf_return:
3414 0000C414 C3                <1>       retn
3415                            <1>
3416                            <1> ; Phase 2
3417                            <1>
3418                            <1> csftdf2_check_cdrv:
3419                            <1>       ; 18/03/2016
3420                            <1>       ; Here, destination drive and directory are ready !
3421                            <1>       ; (checking/restoring is not needed)
3422                            <1>       ; (Since at the end of the phase 1)
3423                            <1>
3424                            <1> ;     mov    dl, [DestinationFile_Drv]
3425                            <1> ;     cmp    dl, [Current_Drv]
3426                            <1> ;     je     short csftdf2_df_check_directory
3427                            <1> ;
3428                            <1> ;     call   change_current_drive
3429                            <1> ;     jc     short csftdf2_read_error
3430                            <1> ;
3431                            <1> ;csftdf2_df_check_directory:
3432                            <1> ;     mov    esi, DestinationFile_Directory
3433                            <1> ;     cmp    byte [esi], 20h
3434                            <1> ;     jna    short csftdf2_df_check_found_or_not
3435                            <1> ;
3436                            <1> ;csftdf2_df_change_directory:
3437                            <1> ;     inc    byte [Restore_CDIR]
3438                            <1> ;     xor    ah, ah ; CD_COMMAND sign -> 0
3439                            <1> ;     call   change_current_directory
3440                            <1> ;     jc     short csftdf2_stc_return
3441                            <1> ;
3442                            <1> ;;csftdf2_df_change_prompt_dir_string:
3443                            <1> ;;    call   change_prompt_dir_string
3444                            <1>
3445                            <1> csftdf2_df_check_found_or_not:
3446                            <1>       ; 21/03/2016
3447 0000C415 803D[96950100]00  <1>       cmp    byte [DestinationFileFound], 0
3448 0000C41C 7739              <1>       ja     short csftdf2_set_sf_percentage
3449                            <1>
3450                            <1> csftdf2_create_file:
3451 0000C41E BE[2E950100]      <1>       mov    esi, DestinationFile_Name
3452 0000C423 A1[98950100]      <1>       mov    eax, [csftdf_filesize]
3453 0000C428 30C9              <1>       xor    cl, cl ; 0
3454                            <1>
3455 0000C42A 31DB              <1>       xor    ebx, ebx ; 0
3456 0000C42C 4B                <1>       dec    ebx ; 0FFFFFFFFh
3457                            <1>
3458                            <1>       ; INPUT ->
3459                            <1>       ;      EAX -> File Size
3460                            <1>       ;      ESI = ASCIIZ File name
3461                            <1>       ;       CL = File attributes
3462                            <1>       ;      EBX = FFFFFFFFh -> empty file sign for FAT fs
3463                            <1>       ;      EBX <> FFFFFFFFh -> use file size for FAT fs
3464                            <1>       ;
3465                            <1>       ; OUTPUT ->
3466                            <1>       ;      EAX = New file's first cluster
3467                            <1>       ;      ESI = Logical Dos Drv Descr. Table Addr.
3468                            <1>       ;      EBX = CreateFile_Size address
3469                            <1>       ;      ECX = Sectors per cluster (<256)
3470                            <1>       ;      EDX = Directory Entry Index/Number (<65536)
3471                            <1>       ;
3472                            <1>       ;      cf = 1 -> error code in AL (EAX)
3473                            <1>
3474 0000C42D E8EC050000        <1>       call   create_file
3475                            <1>       ;pop   esi
3476 0000C432 0F82A3050000      <1>         jc      csftdf2_rw_error
3477                            <1>
3478                            <1> csftdf2_create_file_OK:
3479 0000C438 A3[A8950100]      <1>       mov    [csftdf_df_cluster], eax
3480                            <1>
3481                            <1>       ; 24/03/2016
3482 0000C43D 668915[66950100]  <1>       mov    [DestinationFile_DirEntryNumber], dx
3483                            <1>
3484                            <1>       ; 21/03/2016
3485 0000C444 BE00000800        <1>       mov    esi, Directory_Buffer
3486 0000C449 C1E205            <1>       shl    edx, 5 ; 32 * index number
3487 0000C44C 01D6              <1>       add    esi, edx
3488 0000C44E BF[3E950100]      <1>       mov    edi, DestinationFile_DirEntry
3489 0000C453 B108              <1>       mov    cl, 8 ; 32 bytes
3490 0000C455 F3A5              <1>       rep    movsd
3491                            <1>
```

```
                            <1> csftdf2_set_sf_percentage:
                            <1>       ; 17/03/2016
0000C457 31C0               <1>       xor    eax, eax
0000C459 A2[BC950100]       <1>       mov    [csftdf_percentage], al ; 0, reset
                            <1>
0000C45E A3[B4950100]       <1>       mov    [csftdf_sf_rbytes], eax ; 0, reset
0000C463 A3[B8950100]       <1>       mov    [csftdf_df_wbytes], eax ; 0, reset
                            <1>
0000C468 8A25[6C940100]     <1>       mov    ah, [SourceFile_Drv]
0000C46E BE00010900         <1>       mov    esi, Logical_DOSDisks
0000C473 01C6               <1>       add    esi, eax
                            <1>
0000C475 8935[C0950100]     <1>       mov    [csftdf_sf_drv_dt], esi ; 23/03/2016
                            <1>
0000C47B 668B15[D2940100]   <1>       mov    dx, [SourceFile_DirEntry+DirEntry_FstClusHI]
0000C482 C1E210             <1>       shl    edx, 16
0000C485 668B15[D8940100]   <1>       mov    dx, [SourceFile_DirEntry+DirEntry_FstClusLO]
0000C48C 8915[A4950100]     <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>       ;
0000C492 0FB64E13           <1>       movzx  ecx, byte [esi+LD_BPB+SecPerClust]
0000C496 880D[EA940100]     <1>       mov    [SourceFile_SecPerClust], cl
                            <1>
                            <1>       ; 17/03/2016
0000C49C 386E03             <1>       cmp    [esi+LD_FATType], ch ; 0
0000C49F 7707               <1>       ja     short csftdf2_set_sf_percent_rsize1
                            <1>
0000C4A1 B800000100         <1>       mov    eax, 65536 ; read/write buffer size for Singlix FS
0000C4A6 EB06               <1>       jmp    short csftdf2_set_sf_percent_rsize2
                            <1>
                            <1> csftdf2_set_sf_percent_rsize1:
0000C4A8 668B4611           <1>       mov    ax, [esi+LD_BPB+BytesPerSec]
0000C4AC F7E1               <1>       mul    ecx
                            <1>       ;sub   edx, edx
                            <1> csftdf2_set_sf_percent_rsize2:
0000C4AE A3[AC950100]       <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>
0000C4B3 8B3D[C4950100]     <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]
0000C4B9 8A4F13             <1>       mov    cl, [edi+LD_BPB+SecPerClust]
0000C4BC 880D[6A950100]     <1>       mov    [DestinationFile_SecPerClust], cl
                            <1>
                            <1>       ; 17/03/2016
0000C4C2 386F03             <1>       cmp    [edi+LD_FATType], ch ; 0
0000C4C5 7707               <1>       ja     short csftdf2_set_df_percent_wsize1
                            <1>
0000C4C7 B800000100         <1>       mov    eax, 65536 ; read/write buffer size for Singlix FS
0000C4CC EB06               <1>       jmp    short csftdf2_set_df_percent_wsize2
                            <1>
                            <1> csftdf2_set_df_percent_wsize1:
0000C4CE 0FB74711           <1>       movzx  eax, word [edi+LD_BPB+BytesPerSec]
0000C4D2 F7E1               <1>       mul    ecx
                            <1>       ;sub   edx, edx
                            <1> csftdf2_set_df_percent_wsize2:
0000C4D4 A3[B0950100]       <1>       mov    [csftdf_w_size], eax
                            <1>
0000C4D9 A1[98950100]       <1>       mov    eax, [csftdf_filesize]
                            <1>
0000C4DE 3D00000100         <1>       cmp    eax, 65536 ; 64KB   ; small file
0000C4E3 721F               <1>       jb     short csftdf2_load_file ; do not display percentage
                            <1>
                            <1> csftdf2_reset_wf_percent_ptr_chk_64k:
0000C4E5 B201               <1>       mov    dl, 1 ; 25/03/2016
                            <1>
0000C4E7 3D00000400         <1>       cmp    eax, 65536*4 ; 256KB
0000C4EC 7310               <1>       jnb    short csftdf2_enable_percentage_display ; big file
                            <1>
                            <1>       ; 64-128KB file size for floppy disks
0000C4EE 3815[6C940100]     <1>       cmp    byte [SourceFile_Drv], dl ; 1 ; read from floppy disk ?
0000C4F4 7608               <1>       jna    short csftdf2_enable_percentage_display
                            <1>
0000C4F6 3815[EC940100]     <1>       cmp    byte [DestinationFile_Drv], dl ; 1 ; write to floppy disk ?
0000C4FC 7706               <1>       ja     short csftdf2_load_file
                            <1>
                            <1> csftdf2_enable_percentage_display:
0000C4FE 8815[BC950100]     <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
0000C504 B40F               <1>       mov    ah, 0Fh
0000C506 E85E52FFFF         <1>       call   _int10h
                            <1>       ; 13/05/2016
0000C50B 883D[BD950100]     <1>       mov    [csftdf_videopage], bh ; active video page
```

```
3597 0000C511 B403                <1>       mov   ah, 03h
3598 0000C513 E85152FFFF          <1>       call  _int10h
3599 0000C518 668915[BE950100]    <1>       mov   [csftdf_cursorpos], dx
3600                              <1>
3601 0000C51F 29C0                <1>       sub   eax, eax
3602 0000C521 A2[95950100]        <1>       mov   [csftdf_rw_err], al ; 0
3603                              <1>
3604                              <1> ; ///
3605                              <1> csftdf_sf_amb: ; 15/03/2016
3606 0000C526 8B0D[98950100]      <1>       mov   ecx, [csftdf_filesize]    ; 23/03/2016
3607                              <1>
3608                              <1>       ; TRDOS 386 (TRDOS v2.0)
3609                              <1>       ; Allocate contiguous memory block for loading the file
3610                              <1>
3611                              <1>       ;mov   ecx, [SourceFile_DirEntry+DirEntry_FileSize]
3612                              <1>
3613                              <1>       ;sub   eax, eax ; First free memory aperture
3614                              <1>
3615                              <1>       ; eax = 0 (Allocate memory from the beginning)
3616                              <1>       ; ecx = File (Allocation) size in bytes
3617                              <1>
3618 0000C52C E84C9FFFFF          <1>       call  allocate_memory_block
3619 0000C531 7304                <1>       jnc   short loc_check_sf_save_loading_parms
3620                              <1>
3621 0000C533 29C0                <1>       sub   eax, eax
3622 0000C535 29C9                <1>       sub   ecx, ecx
3623                              <1>
3624                              <1> loc_check_sf_save_loading_parms:
3625 0000C537 A3[9C950100]        <1>       mov   [csftdf_sf_mem_addr], eax ; loading address
3626 0000C53C 890D[A0950100]      <1>       mov   [csftdf_sf_mem_bsize], ecx ; block size
3627                              <1> ; ///
3628                              <1>       ; 19/03/2016
3629 0000C542 8B35[C0950100]      <1>       mov   esi, [csftdf_sf_drv_dt] ; logical dos drv desc. tbl.
3630                              <1>
3631                              <1>       ; 17/03/2016
3632 0000C548 09C0                <1>       or    eax, eax ; contiguous free memory block address
3633 0000C54A 0F845B010000        <1>         jz    csftdf2_read_sf_cluster
3634                              <1>
3635                              <1>       ; 18/03/2016
3636 0000C550 8B1D[9C950100]      <1>       mov   ebx, [csftdf_sf_mem_addr] ; memory block address
3637                              <1>
3638 0000C556 807E0300            <1>       cmp   byte [esi+LD_FATType], 0
3639 0000C55A 0F8605020000        <1>         jna   csftdf2_load_fs_file
3640                              <1>
3641                              <1> csftdf2_load_fat_file:
3642 0000C560 53                  <1>       push  ebx ; *
3643                              <1>
3644                              <1> csftdf2_load_fat_file_next:
3645 0000C561 BE[A1460100]        <1>       mov   esi, msg_reading
3646 0000C566 E8EAAFFFFF          <1>       call  print_msg
3647                              <1>
3648 0000C56B 803D[BC950100]00    <1>       cmp   byte [csftdf_percentage], 0
3649 0000C572 7605                <1>       jna   short csftdf2_load_fat_file_1
3650                              <1>
3651 0000C574 E87C000000          <1>       call  csftdf2_print_percentage ; 19/03/2016
3652                              <1>
3653                              <1> csftdf2_load_fat_file_1:
3654 0000C579 8B35[C0950100]      <1>       mov   esi, [csftdf_sf_drv_dt]
3655 0000C57F 5B                  <1>       pop   ebx ; *
3656                              <1>
3657                              <1> csftdf2_load_fat_file_2:
3658 0000C580 E8B8000000          <1>       call  csftdf2_read_fat_file_sectors ; 19/03/2016
3659 0000C585 0F8250040000        <1>         jc    csftdf2_rw_error ; eocc! or disk error!
3660                              <1>
3661 0000C58B 09D2                <1>       or    edx, edx ; edx > 0 -> EOF
3662 0000C58D 7520                <1>       jnz   short csftdf2_load_fat_file_ok
3663                              <1>
3664 0000C58F 803D[BC950100]00    <1>       cmp   byte [csftdf_percentage], 0
3665 0000C596 76E8                <1>       jna   short csftdf2_load_fat_file_2
3666                              <1>
3667 0000C598 53                  <1>       push  ebx ; *
3668                              <1>
3669                              <1>       ; Set cursor position
3670                              <1>       ; AH= 02h, BH= Page Number, DH= Row, DL= Column
3671 0000C599 8A3D[BD950100]      <1>       mov   bh, [csftdf_videopage]
3672 0000C59F 668B15[BE950100]    <1>       mov   dx, [csftdf_cursorpos]
3673 0000C5A6 B402                <1>       mov   ah, 2
3674 0000C5A8 E8BC51FFFF          <1>       call  _int10h
3675 0000C5AD EBB2                <1>       jmp   short csftdf2_load_fat_file_next
3676                              <1>
3677                              <1> csftdf2_load_fat_file_ok:
3678 0000C5AF 803D[BC950100]00    <1>       cmp   byte [csftdf_percentage], 0
3679 0000C5B6 0F8651020000        <1>         jna   csftdf2_save_file ; 25/03/2016
3680                              <1>
3681                              <1>       ; "Reading... 100%"
3682 0000C5BC BF[B9460100]        <1>       mov   edi, percentagestr
3683 0000C5C1 B031                <1>       mov   al, '1'
3684 0000C5C3 AA                  <1>       stosb
3685 0000C5C4 B030                <1>       mov   al, '0'
3686 0000C5C6 AA                  <1>       stosb
3687 0000C5C7 AA                  <1>       stosb
3688                              <1>
3689 0000C5C8 8A3D[BD950100]      <1>       mov   bh, [csftdf_videopage]
3690 0000C5CE 668B15[BE950100]    <1>       mov   dx, [csftdf_cursorpos]
3691 0000C5D5 B402                <1>       mov   ah, 2
3692 0000C5D7 E88D51FFFF          <1>       call  _int10h
3693                              <1>
3694 0000C5DC BE[A1460100]        <1>       mov   esi, msg_reading
3695 0000C5E1 E86FAFFFFF          <1>       call  print_msg
3696                              <1>
3697 0000C5E6 BE[B9460100]        <1>       mov   esi, percentagestr
3698 0000C5EB E865AFFFFF          <1>       call  print_msg
3699                              <1>
3700 0000C5F0 E918020000          <1>         jmp   csftdf2_save_file ; 25/03/2016
3701                              <1>
```

```
                            <1> csftdf2_print_percentage:
                            <1>       ; 09/12/2017
                            <1>       ; 19/03/2016
                            <1>       ; 18/03/2016
0000C5F5 B020               <1>       mov   al, 20h
0000C5F7 BF[B9460100]       <1>       mov   edi, percentagestr
0000C5FC AA                 <1>       stosb
0000C5FD AA                 <1>       stosb
0000C5FE A1[B4950100]       <1>       mov   eax, [csftdf_sf_rbytes]
0000C603 BA64000000         <1>       mov   edx, 100
0000C608 F7E2               <1>       mul   edx
0000C60A 8B0D[98950100]     <1>       mov   ecx, [csftdf_filesize]
0000C610 F7F1               <1>       div   ecx
0000C612 B10A               <1>       mov   cl, 10
0000C614 F6F1               <1>       div   cl
0000C616 80C430             <1>       add   ah, '0'
0000C619 8827               <1>       mov   [edi], ah
0000C61B 20C0               <1>       and   al, al
0000C61D 740A               <1>       jz    short csftdf2_print_percent_1
0000C61F 4F                 <1>       dec   edi
                            <1>       ;cbw
0000C620 28E4               <1>       sub   ah, ah ; 09/12/2017
0000C622 F6F1               <1>       div   cl
0000C624 80C430             <1>       add   ah, '0'
0000C627 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:
0000C629 BE[B9460100]       <1>       mov   esi, percentagestr
                            <1>       ;call print_msg
                            <1>       ;retn
0000C62E E922AFFFFF         <1>       jmp   print_msg
                            <1>
                            <1> csftdf2_read_file_sectors:
                            <1>       ; 19/03/2016
0000C633 807E0300           <1>       cmp   byte [esi+LD_FATType], 0
0000C637 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:
0000C63D 8B15[98950100]     <1>       mov   edx, [csftdf_filesize]
0000C643 2B15[B4950100]     <1>       sub   edx, [csftdf_sf_rbytes]
0000C649 3B15[AC950100]     <1>       cmp   edx, [csftdf_r_size]
0000C64F 7306               <1>       jnb   short csftdf2_read_fat_file_secs_1
0000C651 8915[AC950100]     <1>       mov   [csftdf_r_size], edx
                            <1>
                            <1> csftdf2_read_fat_file_secs_1:
0000C657 A1[AC950100]       <1>       mov   eax, [csftdf_r_size]
0000C65C 29D2               <1>       sub   edx, edx
0000C65E 0FB74E11           <1>       movzx ecx, word [esi+LD_BPB+BytesPerSec]
0000C662 01C8               <1>       add   eax, ecx
0000C664 48                 <1>       dec   eax
0000C665 F7F1               <1>       div   ecx
0000C667 89C1               <1>       mov   ecx, eax ; sector count
0000C669 A1[A4950100]       <1>       mov   eax, [csftdf_sf_cluster]
                            <1>
                            <1>       ; EBX = memory block address (current)
                            <1>
0000C66E E821090000         <1>       call  read_fat_file_sectors
0000C673 7235               <1>       jc    short csftdf2_read_fat_file_secs_3
                            <1>
                            <1>       ; EBX = next memory address
                            <1>
0000C675 A1[B4950100]       <1>       mov   eax, [csftdf_sf_rbytes]
0000C67A 0305[AC950100]     <1>       add   eax, [csftdf_r_size]
0000C680 8B15[98950100]     <1>       mov   edx, [csftdf_filesize]
0000C686 39D0               <1>       cmp   eax, edx
0000C688 7320               <1>       jnb   short csftdf2_read_fat_file_secs_3 ; edx > 0
0000C68A A3[B4950100]       <1>       mov   [csftdf_sf_rbytes], eax
                            <1>
0000C68F 53                 <1>       push  ebx ; *
                            <1>       ; get next cluster (csftdf_r_size! bytes)
0000C690 A1[A4950100]       <1>       mov   eax, [csftdf_sf_cluster]
0000C695 E8CC060000         <1>       call  get_next_cluster
0000C69A 5B                 <1>       pop   ebx ; *
0000C69B 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
0000C69D B811000000         <1>       mov   eax, 17 ; Read error !
0000C6A2 C3                 <1>       retn
                            <1>
                            <1> csftdf2_read_fat_file_secs_2:
0000C6A3 29D2               <1>       sub   edx, edx ; 0
0000C6A5 A3[A4950100]       <1>       mov   [csftdf_sf_cluster], eax ; next cluster
                            <1>
                            <1> csftdf2_read_fat_file_secs_3:
0000C6AA C3                 <1>       retn
                            <1>
                            <1> csftdf2_read_sf_cluster:
                            <1>       ; 19/03/2016
0000C6AB BB00000700         <1>       mov   ebx, Cluster_Buffer ; buffer address (64KB)
                            <1>
0000C6B0 803D[BC950100]00   <1>       cmp   byte [csftdf_percentage], 0
0000C6B7 760D               <1>       jna   short csftdf2_read_sf_clust_2
```

```
                                <1>
0000C6B9 53                     <1>       push   ebx ; *
                                <1>
                                <1> csftdf2_read_sf_clust_next:
0000C6BA E836FFFFFF             <1>       call   csftdf2_print_percentage
                                <1>
                                <1> csftdf2_read_sf_clust_0:
0000C6BF 8B35[C0950100]         <1>       mov    esi, [csftdf_sf_drv_dt]
                                <1> csftdf2_read_sf_clust_1:
0000C6C5 5B                     <1>       pop    ebx ; *
                                <1>
                                <1> csftdf2_read_sf_clust_2:
0000C6C6 89DA                   <1>       mov    edx, ebx
0000C6C8 0315[AC950100]         <1>       add    edx, [csftdf_r_size]
0000C6CE 81FA00000800           <1>       cmp    edx, Cluster_Buffer + 65536
0000C6D4 772F                   <1>       ja     short csftdf2_write_df_cluster
                                <1>
0000C6D6 E858FFFFFF             <1>       call   csftdf2_read_file_sectors ; 19/03/2016
0000C6DB 0F8280020000           <1>         jc     csftdf2_save_fat_file_err2 ; eocc! or disk error!
                                <1>
0000C6E1 09D2                   <1>       or     edx, edx ; edx > 0 -> EOF
0000C6E3 7520                   <1>       jnz    short csftdf2_write_df_cluster
                                <1>
0000C6E5 803D[BC950100]00       <1>       cmp    byte [csftdf_percentage], 0
0000C6EC 76D8                   <1>       jna    short csftdf2_read_sf_clust_2
                                <1>
0000C6EE 53                     <1>       push   ebx ; *
                                <1>
                                <1>       ; Set cursor position
                                <1>       ; AH= 02h, BH= Page Number, DH= Row, DL= Column
0000C6EF 8A3D[BD950100]         <1>       mov    bh, [csftdf_videopage]
0000C6F5 668B15[BE950100]       <1>       mov    dx, [csftdf_cursorpos]
0000C6FC B402                   <1>       mov    ah, 2
0000C6FE E86650FFFF             <1>       call   _int10h
0000C703 EBB5                   <1>       jmp    short csftdf2_read_sf_clust_next
                                <1>
                                <1> csftdf2_write_df_cluster:
                                <1>       ; 19/03/2016
0000C705 8B35[C4950100]         <1>       mov    esi, [csftdf_df_drv_dt]
0000C70B BB00000700             <1>       mov    ebx, Cluster_Buffer ; buffer address (64KB)
                                <1>
                                <1> csftdf2_write_df_clust_next:
0000C710 E855000000             <1>       call   csftdf2_write_file_sectors ; 19/03/2016
0000C715 0F8246020000           <1>         jc     csftdf2_save_fat_file_err2 ; eocc! or disk error!
                                <1>
0000C71B 09D2                   <1>       or     edx, edx ; edx > 0 -> EOF
0000C71D 750A                   <1>       jnz    short csftdf2_rw_f_clust_ok
                                <1>
0000C71F 81FB00000800           <1>       cmp    ebx, Cluster_Buffer + 65536
0000C725 72E9                   <1>       jb     short csftdf2_write_df_clust_next
                                <1>
0000C727 EB82                   <1>       jmp    short csftdf2_read_sf_cluster
                                <1>
                                <1> csftdf2_rw_f_clust_ok:
0000C729 803D[BC950100]00       <1>       cmp    byte [csftdf_percentage], 0
0000C730 0F86B2010000           <1>         jna    csftdf2_save_fat_file_4 ; 25/03/2016
                                <1>
                                <1>       ; "100%"
0000C736 BF[B9460100]           <1>       mov    edi, percentagestr
0000C73B B031                   <1>       mov    al, '1'
0000C73D AA                     <1>       stosb
0000C73E B030                   <1>       mov    al, '0'
0000C740 AA                     <1>       stosb
0000C741 AA                     <1>       stosb
                                <1>
0000C742 8A3D[BD950100]         <1>       mov    bh, [csftdf_videopage]
0000C748 668B15[BE950100]       <1>       mov    dx, [csftdf_cursorpos]
0000C74F B402                   <1>       mov    ah, 2
0000C751 E81350FFFF             <1>       call   _int10h
                                <1>
0000C756 BE[B9460100]           <1>       mov    esi, percentagestr
0000C75B E8F5ADFFFF             <1>       call   print_msg
                                <1>
0000C760 E983010000             <1>         jmp    csftdf2_save_fat_file_4
                                <1>
                                <1> csftdf2_load_fs_file:
                                <1>       ; temporary - 18/03/2016
0000C765 E96F020000             <1>         jmp    csftdf2_read_error
                                <1>
                                <1> csftdf2_write_file_sectors:
                                <1>       ; 19/03/2016
0000C76A 807E0300               <1>       cmp    byte [esi+LD_FATType], 0
0000C76E 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:
0000C774 8B15[98950100]         <1>       mov    edx, [csftdf_filesize]
0000C77A 2B15[B8950100]         <1>       sub    edx, [csftdf_df_wbytes]
0000C780 3B15[B0950100]         <1>       cmp    edx, [csftdf_w_size]
0000C786 7306                   <1>       jnb    short csftdf2_write_fat_file_secs_1
0000C788 8915[B0950100]         <1>       mov    [csftdf_w_size], edx
                                <1>
                                <1> csftdf2_write_fat_file_secs_1:
0000C78E A1[B0950100]           <1>       mov    eax, [csftdf_w_size]
0000C793 29D2                   <1>       sub    edx, edx
0000C795 0FB74E11               <1>       movzx  ecx, word [esi+LD_BPB+BytesPerSec]
0000C799 01C8                   <1>       add    eax, ecx
0000C79B 48                     <1>       dec    eax
```

```
3912 0000C79C F7F1               <1>       div    ecx
3913 0000C79E 89C1               <1>       mov    ecx, eax ; sector count
3914 0000C7A0 A1[A8950100]       <1>       mov    eax, [csftdf_df_cluster]
3915                             <1>
3916                             <1>       ; EBX = memory block address (current)
3917                             <1>
3918 0000C7A5 E8A20F0000         <1>       call   write_fat_file_sectors
3919 0000C7AA 7259               <1>       jc     short csftdf2_write_fat_file_secs_4
3920                             <1>
3921                             <1>       ; EBX = next memory address
3922                             <1>
3923 0000C7AC A1[B8950100]       <1>       mov    eax, [csftdf_df_wbytes]
3924 0000C7B1 0305[B0950100]     <1>       add    eax, [csftdf_w_size]
3925 0000C7B7 8B15[98950100]     <1>       mov    edx, [csftdf_filesize]
3926 0000C7BD 39D0               <1>       cmp    eax, edx
3927 0000C7BF 7344               <1>       jnb    short csftdf2_write_fat_file_secs_4
3928 0000C7C1 A3[B8950100]       <1>       mov    [csftdf_df_wbytes], eax
3929                             <1>       ;
3930 0000C7C6 A3[5A950100]       <1>       mov    [DestinationFile_DirEntry+DirEntry_FileSize], eax
3931                             <1>
3932 0000C7CB 53                 <1>       push   ebx ; *
3933                             <1>
3934 0000C7CC 803D[96950100]01   <1>       cmp    byte [DestinationFileFound], 1
3935 0000C7D3 7210               <1>       jb     short csftdf2_write_fat_file_secs_2
3936                             <1>
3937                             <1>       ; get next cluster (csftdf_w_size! bytes)
3938 0000C7D5 A1[A8950100]       <1>       mov    eax, [csftdf_df_cluster]
3939 0000C7DA E887050000         <1>       call   get_next_cluster
3940 0000C7DF 731C               <1>       jnc    short csftdf2_write_fat_file_secs_3
3941                             <1>
3942 0000C7E1 21C0               <1>       and    eax, eax ; end of cluster chain!?
3943 0000C7E3 7521               <1>       jnz    short csftdf2_write_fat_file_secs_5 ; disk error !
3944                             <1>
3945                             <1> csftdf2_write_fat_file_secs_2:
3946 0000C7E5 A1[A8950100]       <1>       mov    eax, [csftdf_df_cluster] ; last cluster
3947 0000C7EA E8800E0000         <1>       call   add_new_cluster
3948 0000C7EF 7215               <1>       jc     short csftdf2_write_fat_file_secs_5
3949                             <1>
3950                             <1>       ; NOTE: Destination file size may be bigger than
3951                             <1>       ; source file size when the last reading fails after here.
3952                             <1>       ; (The last -empty- cluster of destination file must be
3953                             <1>       ; truncated and LMDT must be current date&time for partial
3954                             <1>       ; copy result!)
3955 0000C7F1 8B15[B0950100]     <1>       mov    edx, [csftdf_w_size] ; bytes per cluster
3956 0000C7F7 0115[5A950100]     <1>       add    [DestinationFile_DirEntry+DirEntry_FileSize], edx
3957                             <1>
3958                             <1> csftdf2_write_fat_file_secs_3:
3959 0000C7FD 5B                 <1>       pop    ebx ; *
3960 0000C7FE 29D2               <1>       sub    edx, edx ; 0
3961 0000C800 A3[A8950100]       <1>       mov    [csftdf_df_cluster], eax ; next cluster
3962                             <1>
3963                             <1> csftdf2_write_fat_file_secs_4:
3964 0000C805 C3                 <1>       retn
3965                             <1>
3966                             <1> csftdf2_write_fat_file_secs_5:
3967 0000C806 5B                 <1>       pop    ebx ; *
3968                             <1>       ; 16/10/2016 (1Dh -> 18)
3969 0000C807 B812000000         <1>       mov    eax, 18 ; Write error !
3970 0000C80C C3                 <1>       retn
3971                             <1>
3972                             <1> csftdf2_save_file:
3973                             <1>       ; 09/12/2017
3974                             <1>       ; 25/03/2016
3975                             <1>       ; 19/03/2016
3976                             <1>       ; 18/03/2016
3977 0000C80D 8B35[C4950100]     <1>       mov    esi, [csftdf_df_drv_dt] ; logical dos drv desc. tbl.
3978                             <1>
3979 0000C813 8B1D[9C950100]     <1>       mov    ebx, [csftdf_sf_mem_addr] ; memory block address
3980                             <1>
3981 0000C819 807E0300           <1>       cmp    byte [esi+LD_FATType], 0
3982 0000C81D 0F86F4010000       <1>         jna     csftdf2_save_fs_file
3983                             <1>
3984                             <1> csftdf2_save_fat_file:
3985 0000C823 53                 <1>       push   ebx; *
3986                             <1>
3987 0000C824 803D[BC950100]00   <1>       cmp    byte [csftdf_percentage], 0
3988 0000C82B 7724               <1>       ja     short csftdf2_save_fat_file_0
3989                             <1>
3990                             <1>       ; Set cursor position
3991                             <1>       ; AH= 02h, BH= Page Number, DH= Row, DL= Column
3992 0000C82D 8A3D[BD950100]     <1>       mov    bh, [csftdf_videopage]
3993 0000C833 668B15[BE950100]   <1>       mov    dx, [csftdf_cursorpos]
3994 0000C83A B402               <1>       mov    ah, 2
3995 0000C83C E8284FFFFF         <1>       call   _int10h
3996                             <1>
3997 0000C841 BE[AD460100]       <1>       mov    esi, msg_writing
3998 0000C846 E80AADFFFF         <1>       call   print_msg
3999                             <1>
4000                             <1> csftdf2_save_fat_file_next:
4001 0000C84B 8B35[C4950100]     <1>       mov    esi, [csftdf_df_drv_dt] ; 25/03/2016
4002                             <1>
4003                             <1> csftdf2_save_fat_file_0:
4004 0000C851 5B                 <1>       pop    ebx ; *
4005                             <1>
4006                             <1> csftdf2_save_fat_file_1:
4007 0000C852 E813FFFFFF         <1>       call   csftdf2_write_file_sectors ; 19/03/2016
4008 0000C857 0F827E010000       <1>         jc      csftdf2_rw_error ; eocc! or disk error!
4009                             <1>
4010 0000C85D 09D2               <1>       or     edx, edx ; edx > 0 -> EOF
4011 0000C85F 756D               <1>         jnz     short csftdf2_save_fat_file_3 ; 25/03/2016
4012                             <1>
4013 0000C861 803D[BC950100]00   <1>       cmp    byte [csftdf_percentage], 0
4014 0000C868 76E8               <1>       jna    short csftdf2_save_fat_file_1
4015                             <1>
4016 0000C86A B020               <1>       mov    al, 20h
```

```
4017 0000C86C BF[B9460100]        <1>       mov   edi, percentagestr
4018 0000C871 AA                  <1>       stosb
4019 0000C872 AA                  <1>       stosb
4020 0000C873 A1[B8950100]        <1>       mov   eax, [csftdf_df_wbytes]
4021 0000C878 BA64000000          <1>       mov   edx, 100
4022 0000C87D F7E2                <1>       mul   edx
4023 0000C87F 8B0D[98950100]      <1>       mov   ecx, [csftdf_filesize]
4024 0000C885 F7F1                <1>       div   ecx
4025 0000C887 B10A                <1>       mov   cl, 10
4026 0000C889 F6F1                <1>       div   cl
4027 0000C88B 80C430              <1>       add   ah, '0'
4028 0000C88E 8827                <1>       mov   [edi], ah
4029 0000C890 20C0                <1>       and   al, al
4030 0000C892 740A                <1>       jz    short csftdf2_save_fat_file_2
4031 0000C894 4F                  <1>       dec   edi
4032                              <1>       ;cbw
4033 0000C895 30E4                <1>       xor   ah, ah ; 09/12/2017
4034 0000C897 F6F1                <1>       div   cl
4035 0000C899 80C430              <1>       add   ah, '0'
4036 0000C89C 8827                <1>       mov   [edi], ah
4037                              <1>       ;and  al, al
4038                              <1>       ;jz   short csftdf2_save_fat_file_2
4039                              <1>       ;dec  edi
4040                              <1>       ;mov  [edi], '1' ; 100%
4041                              <1>
4042                              <1> csftdf2_save_fat_file_2:
4043 0000C89E 53                  <1>       push  ebx ; *
4044                              <1>
4045 0000C89F E802000000          <1>       call  csftdf2_print_wr_percentage ; 25/03/2016
4046                              <1>
4047 0000C8A4 EBA5                <1>         jmp     csftdf2_save_fat_file_next
4048                              <1>
4049                              <1> csftdf2_print_wr_percentage:
4050                              <1>       ; Set cursor position
4051                              <1>       ; AH= 02h, BH= Page Number, DH= Row, DL= Column
4052 0000C8A6 8A3D[BD950100]      <1>       mov   bh, [csftdf_videopage]
4053 0000C8AC 668B15[BE950100]    <1>       mov   dx, [csftdf_cursorpos]
4054 0000C8B3 B402                <1>       mov   ah, 2
4055 0000C8B5 E8AF4EFFFF          <1>       call  _int10h
4056                              <1>
4057 0000C8BA BE[AD460100]        <1>       mov   esi, msg_writing
4058 0000C8BF E891ACFFFF          <1>       call  print_msg
4059                              <1>
4060 0000C8C4 BE[B9460100]        <1>       mov   esi, percentagestr
4061                              <1>       ;call print_msg
4062                              <1>       ;retn
4063 0000C8C9 E987ACFFFF          <1>       jmp   print_msg
4064                              <1>
4065                              <1> csftdf2_save_fat_file_3:
4066 0000C8CE 803D[BC950100]00    <1>       cmp   byte [csftdf_percentage], 0
4067 0000C8D5 7611                <1>         jna     csftdf2_save_fat_file_4 ; 25/03/2016
4068                              <1>
4069                              <1>       ; "100%"
4070 0000C8D7 BF[B9460100]        <1>       mov   edi, percentagestr
4071 0000C8DC B031                <1>       mov   al, '1'
4072 0000C8DE AA                  <1>       stosb
4073 0000C8DF B030                <1>       mov   al, '0'
4074 0000C8E1 AA                  <1>       stosb
4075 0000C8E2 AA                  <1>       stosb
4076                              <1>
4077 0000C8E3 E8BEFFFFFF          <1>       call  csftdf2_print_wr_percentage
4078                              <1>
4079                              <1> csftdf2_save_fat_file_4:
4080 0000C8E8 803D[96950100]00    <1>       cmp   byte [DestinationFileFound], 0
4081 0000C8EF 7647                <1>       jna   short csftdf2_save_fat_file_6
4082                              <1>
4083 0000C8F1 8B35[C4950100]      <1>       mov   esi, [csftdf_df_drv_dt] ; 31/03/2016
4084                              <1>
4085 0000C8F7 A1[A8950100]        <1>       mov   eax, [csftdf_df_cluster] ; last cluster
4086 0000C8FC E865040000          <1>       call  get_next_cluster
4087 0000C901 7235                <1>       jc    short csftdf2_save_fat_file_6 ; eocc! or disk error!
4088                              <1>
4089 0000C903 A1[A8950100]        <1>       mov   eax, [csftdf_df_cluster] ; last cluster
4090                              <1>       ;xor  ecx, ecx
4091                              <1>       ;mov  [FAT_ClusterCounter], ecx ; 0 ; reset
4092                              <1>       ;dec  ecx ; 0FFFFFFFFh
4093                              <1>       ;shr  ecx, 4 ; 28 bit ; 0FFFFFFFh
4094 0000C908 B9FFFFFF0F          <1>       mov   ecx, 0FFFFFFFh
4095 0000C90D E87E070000          <1>       call  update_cluster
4096 0000C912 7224                <1>       jc    short csftdf2_save_fat_file_6 ; really last cluster!?
4097                              <1>
4098 0000C914 A3[A8950100]        <1>       mov   [csftdf_df_cluster], eax ; next cluster
4099                              <1>
4100                              <1>       ; byte [FAT_BuffValidData] = 2
4101 0000C919 E82F0A0000          <1>       call  save_fat_buffer
4102 0000C91E 730E                <1>       jnc   short csftdf2_save_fat_file_5
4103                              <1>
4104 0000C920 8B15[98950100]      <1>       mov   edx, [csftdf_filesize]
4105 0000C926 8915[5A950100]      <1>       mov   [DestinationFile_DirEntry+DirEntry_FileSize], edx
4106 0000C92C EB58                <1>       jmp   short csftdf2_save_fat_file_err3
4107                              <1>
4108                              <1> csftdf2_save_fat_file_5:
4109 0000C92E A1[A8950100]        <1>       mov   eax, [csftdf_df_cluster]
4110                              <1>
4111                              <1>       ; EAX = First cluster to be truncated/unlinked
4112                              <1>       ; ESI = Logical dos drive description table address
4113 0000C933 E8580C0000          <1>       call  truncate_cluster_chain
4114                              <1>
4115                              <1> csftdf2_save_fat_file_6:
4116                              <1>       ; 28/03/2016
4117 0000C938 BE[C9940100]        <1>       mov   esi, SourceFile_DirEntry+DirEntry_Attr ; +11 to + 18
4118 0000C93D BF[49950100]        <1>       mov   edi, DestinationFile_DirEntry+DirEntry_Attr ; +11 to + 18
4119 0000C942 A4                  <1>       movsb ; +11
4120 0000C943 A5                  <1>       movsd ; +12 .. +15
4121 0000C944 66A5                <1>       movsw ; +16 .. +17
```

```
4122                                <1>             ; + 18
4123 0000C946 83C604                <1>       add   esi, 4
4124 0000C949 83C704                <1>       add   edi, 4
4125 0000C94C A5                    <1>       movsd ; DirEntry_WrtTime ; +22 .. +25
4126                                <1>
4127 0000C94D 8B15[98950100]        <1>       mov   edx, [csftdf_filesize]
4128 0000C953 8915[5A950100]        <1>       mov   [DestinationFile_DirEntry+DirEntry_FileSize], edx
4129                                <1>
4130 0000C959 E8BAF0FFFF            <1>       call  convert_current_date_time
4131                                <1>       ; DX = Date in dos dir entry format
4132                                <1>       ; AX = Time in dos dir entry format
4133 0000C95E EB4D                  <1>       jmp   short csftdf2_save_fat_file_7
4134                                <1>
4135                                <1> csftdf2_save_fat_file_err1:
4136 0000C960 5B                    <1>       pop   ebx ; *
4137                                <1> csftdf2_save_fat_file_err2:
4138 0000C961 A1[B8950100]          <1>       mov   eax, [csftdf_df_wbytes]
4139 0000C966 8B15[5A950100]        <1>       mov   edx, [DestinationFile_DirEntry+DirEntry_FileSize]
4140 0000C96C 39C2                  <1>       cmp   edx, eax
4141 0000C96E 7616                  <1>       jna   short csftdf2_save_fat_file_err3
4142 0000C970 A1[A8950100]          <1>       mov   eax, [csftdf_df_cluster] ; last (empty) cluster
4143                                <1>       ; ESI = Logical dos drive description table address
4144 0000C975 E8160C0000            <1>       call  truncate_cluster_chain
4145 0000C97A 720A                  <1>       jc    short csftdf2_save_fat_file_err3
4146 0000C97C A1[B8950100]          <1>       mov   eax, [csftdf_df_wbytes]
4147 0000C981 A3[5A950100]          <1>       mov   [DestinationFile_DirEntry+DirEntry_FileSize], eax
4148                                <1> csftdf2_save_fat_file_err3:
4149 0000C986 E88DF0FFFF            <1>       call  convert_current_date_time
4150                                <1>       ; DX = Date in dos dir entry format
4151                                <1>       ; AX = Time in dos dir entry format
4152 0000C98B C605[4B950100]00      <1>       mov   byte [DestinationFile_DirEntry+DirEntry_CrtTimeTenth], 0
4153 0000C992 66A3[4C950100]        <1>       mov   [DestinationFile_DirEntry+DirEntry_CrtTime], ax
4154 0000C998 668915[4E950100]      <1>       mov   [DestinationFile_DirEntry+DirEntry_CrtDate], dx
4155 0000C99F 66A3[54950100]        <1>       mov   [DestinationFile_DirEntry+DirEntry_WrtTime], ax
4156 0000C9A5 668915[56950100]      <1>       mov   [DestinationFile_DirEntry+DirEntry_WrtDate], dx
4157 0000C9AC F9                    <1>       stc
4158                                <1> csftdf2_save_fat_file_7:
4159 0000C9AD 9C                    <1>       pushf
4160 0000C9AE 668915[50950100]      <1>       mov   [DestinationFile_DirEntry+DirEntry_LastAccDate], dx
4161 0000C9B5 BE[3E950100]          <1>       mov   esi, DestinationFile_DirEntry
4162 0000C9BA BF00000800            <1>       mov   edi, Directory_Buffer
4163 0000C9BF 0FB70D[66950100]      <1>       movzx ecx, word [DestinationFile_DirEntryNumber] ; (<2048)
4164 0000C9C6 66C1E105              <1>       shl   cx, 5 ; 32 * directory entry number
4165 0000C9CA 01CF                  <1>       add   edi, ecx
4166                                <1>       ;mov  ecx, 8
4167 0000C9CC 66B90800              <1>       mov   cx, 8
4168 0000C9D0 F3A5                  <1>       rep   movsd
4169 0000C9D2 9D                    <1>       popf
4170 0000C9D3 730B                  <1>       jnc   short csftdf2_write_file_OK
4171                                <1>
4172                                <1> csftdf2_write_error:
4173                                <1>       ; 18/03/2016
4174 0000C9D5 B01D                  <1>       mov   al, 1Dh ; write error
4175 0000C9D7 EB02                  <1>       jmp   short csftdf2_rw_error
4176                                <1>
4177                                <1>       ; 16/03/2016
4178                                <1> csftdf2_read_error:
4179 0000C9D9 B011                  <1>       mov   al, 17 ; ; Drive not ready or read error!
4180                                <1> csftdf2_rw_error:
4181 0000C9DB A2[95950100]          <1>       mov   [csftdf_rw_err], al
4182                                <1>
4183                                <1> csftdf2_write_file_OK:
4184                                <1>       ; 18/03/2016
4185 0000C9E0 C605[AC910100]02      <1>       mov   byte [DirBuff_ValidData], 2
4186 0000C9E7 E8CAF0FFFF            <1>       call  save_directory_buffer
4187                                <1>
4188                                <1>       ; Update last modification date&time of destination
4189                                <1>       ; file's (parent) directory
4190 0000C9EC E860F1FFFF            <1>       call  update_parent_dir_lmdt
4191                                <1>       ;
4192 0000C9F1 A1[9C950100]          <1>       mov   eax, [csftdf_sf_mem_addr] ; start address
4193                                <1>
4194 0000C9F6 21C0                  <1>       and   eax, eax
4195 0000C9F8 750E                  <1>       jnz   short csftdf2_dealloc_mblock
4196                                <1>
4197 0000C9FA 88C5                  <1>       mov   ch, al ; 0 (Cluster r/w, not full loading)
4198                                <1> csftdf2_dealloc_retn:
4199 0000C9FC 8A0D[95950100]        <1>       mov   cl, [csftdf_rw_err]
4200 0000CA02 A1[A8950100]          <1>       mov   eax, [csftdf_df_cluster]
4201 0000CA07 C3                    <1>       retn
4202                                <1>
4203                                <1> csftdf2_dealloc_mblock:
4204 0000CA08 8B0D[A0950100]        <1>       mov   ecx, [csftdf_sf_mem_bsize] ; block size
4205 0000CA0E E8779CFFFF            <1>       call  deallocate_memory_block
4206 0000CA13 B5FF                  <1>         mov     ch, 0FFh ; (File was full loaded at memory)
4207 0000CA15 EBE5                  <1>       jmp   short csftdf2_dealloc_retn
4208                                <1>
4209                                <1> csftdf2_save_fs_file:
4210                                <1>       ; 16/10/2016 (1Dh -> 18)
4211                                <1>       ; temporary - (21/03/2016)
4212 0000CA17 B812000000            <1>       mov   eax, 18 ; write error
4213 0000CA1C F9                    <1>       stc
4214 0000CA1D C3                    <1>       retn
4215                                <1>
4216                                <1> create_file:
4217                                <1>       ; 16/10/2016
4218                                <1>       ; 24/03/2016, 31/03/2016
4219                                <1>       ; 20/03/2016, 21/03/2016, 23/03/2016
4220                                <1>       ; 19/03/2016 (TRDOS 396 = TRDOS v2.0)
4221                                <1>       ; 03/09/2011 (FILE.ASM, 'proc_create_file')
4222                                <1>       ; 09/08/2010
4223                                <1>       ;
4224                                <1>       ; INPUT ->
4225                                <1>       ;     EAX = File Size
4226                                <1>       ;     ESI = ASCIIZ File Name
```

```
4227                            <1>       ;      CL = File Attributes
4228                            <1>       ;      EBX = FFFFFFFFh -> create empty file
4229                            <1>       ;                   (only for FAT fs)
4230                            <1>       ; OUTPUT ->
4231                            <1>       ;     CF = 0 ->
4232                            <1>       ;      EAX = New file's first cluster
4233                            <1>       ;      ESI = Logical Dos Drv Descr. Table Addr.
4234                            <1>       ;      EBX = offset CreateFile_Size
4235                            <1>       ;      ECX = Sectors per cluster (<256)
4236                            <1>       ;      EDX = Directory entry index/number (<65536)
4237                            <1>       ;     CF = 1 -> error code in AL
4238                            <1>
4239                            <1> ;     test   cl, 18h (directory or volume name)
4240                            <1> ;     jnz    short loc_createfile_access_denied
4241 0000CA1E 80E107           <1>       and    cl, 07h ; S, H, R
4242 0000CA21 880D[E4950100]   <1>         mov     [createfile_attrib], cl
4243                            <1>
4244 0000CA27 89D9             <1>       mov    ecx, ebx
4245 0000CA29 89F3             <1>       mov    ebx, esi ; ASCIIZ File Name address
4246 0000CA2B 29D2             <1>       sub    edx, edx
4247 0000CA2D 8A35[868A0100]   <1>         mov     dh, [Current_Drv]
4248 0000CA33 BE00010900       <1>         mov     esi, Logical_DOSDisks
4249 0000CA38 01D6             <1>       add    esi, edx
4250                            <1>
4251 0000CA3A 8815[EF950100]   <1>       mov    [createfile_UpdatePDir], dl ; 0 ; 31/03/2016
4252                            <1>
4253                            <1>       ; LD_DiskType = 0 for write protection (read only)
4254 0000CA40 807E0101         <1>       cmp    byte [esi+LD_DiskType], 1 ; 0 = Invalid
4255 0000CA44 730A             <1>       jnb    short loc_createfile_check_file_sytem
4256                            <1>       ; 16/10/2016 (TRDOS Error code: 30, disk write protected)
4257 0000CA46 B81E000000       <1>       mov    eax, 30 ; 13h, MSDOS err : Disk write-protected
4258 0000CA4B 66BA0000         <1>       mov    dx, 0
4259                            <1>       ; err retn: EDX = 0, EBX = File name offset
4260                            <1>       ; ESI -> Dos drive description table address
4261 0000CA4F C3               <1>       retn
4262                            <1>
4263                            <1> ;loc_createfile_access_denied:
4264                            <1> ;     mov    eax, 05h ; access denied (invalid attributes input)
4265                            <1> ;     stc
4266                            <1> ;     retn
4267                            <1>
4268                            <1> loc_createfile_check_file_sytem:
4269 0000CA50 807E0301         <1>       cmp    byte [esi+LD_FATType], 1
4270 0000CA54 730A             <1>       jnb    short loc_createfile_chk_empty_FAT_file_sign1
4271                            <1>
4272 0000CA56 A3[D0950100]     <1>       mov    [createfile_size], eax
4273                            <1>       ; ESI = Logical Dos Drive Description Table address
4274                            <1>       ; EBX = ASCIIZ File Name address
4275 0000CA5B E9FE020000       <1>       jmp    create_fs_file
4276                            <1>
4277                            <1> loc_createfile_chk_empty_FAT_file_sign1:
4278                            <1>       ; ECX = FFFFFFFFh -> create empty file if drive has FAT fs
4279 0000CA60 41               <1>       inc    ecx
4280 0000CA61 7506             <1>       jnz    short loc_createfile_chk_empty_FAT_file_sign2
4281 0000CA63 890D[D0950100]   <1>       mov    [createfile_size], ecx ; 0 ; empty file
4282                            <1>
4283                            <1> loc_createfile_chk_empty_FAT_file_sign2:
4284                            <1>       ; 23/03/2016
4285 0000CA69 668B4E11         <1>       mov    cx, [esi+LD_BPB+BytesPerSec]
4286 0000CA6D 66890D[EC950100] <1>       mov    [createfile_BytesPerSec], cx
4287                            <1>
4288                            <1>       ; EBX = ASCIIZ File Name address
4289 0000CA74 0FB65613         <1>       movzx  edx, byte [esi+LD_BPB+SecPerClust]
4290 0000CA78 8815[E5950100]   <1>       mov    [createfile_SecPerClust], dl
4291 0000CA7E 8B4E74           <1>       mov    ecx, [esi+LD_FreeSectors]
4292 0000CA81 39D1             <1>       cmp    ecx, edx ; byte [createfile_SecPerClust]
4293 0000CA83 7306             <1>       jnb    short loc_create_fat_file
4294                            <1>
4295                            <1> loc_createfile_insufficient_disk_space:
4296 0000CA85 B827000000       <1>       mov    eax, 27h
4297                            <1> loc_createfile_gffc_retn:
4298 0000CA8A C3               <1>       retn
4299                            <1>
4300                            <1> loc_create_fat_file:
4301 0000CA8B 891D[C8950100]   <1>       mov    [createfile_Name_Offset], ebx
4302 0000CA91 890D[CC950100]   <1>       mov    [createfile_FreeSectors], ecx
4303                            <1>
4304                            <1> loc_createfile_gffc_1:
4305 0000CA97 E821050000       <1>       call   get_first_free_cluster
4306 0000CA9C 72EC             <1>       jc     short loc_createfile_gffc_retn
4307                            <1>
4308 0000CA9E A3[D4950100]     <1>       mov    [createfile_FFCluster], eax
4309                            <1>
4310                            <1> loc_createfile_locate_ffe_on_directory:
4311                            <1>       ; Current directory fcluster <> Directory buffer cluster
4312                            <1>       ; Current directory will be reloaded by
4313                            <1>       ; 'locate_current_dir_file' procedure
4314                            <1>       ;
4315                            <1>       ; ESI = Logical Dos Drv Desc. Table Adress
4316 0000CAA3 56               <1>       push   esi ; *
4317 0000CAA4 31C0             <1>       xor    eax, eax
4318                            <1>
4319 0000CAA6 A3[A2910100]     <1>       mov    dword [FAT_ClusterCounter], eax ; 0
4320                            <1>       ; 21/03/2016
4321 0000CAAB A2[EE950100]     <1>       mov    byte [createfile_wfc], al ; 0
4322                            <1>
4323 0000CAB0 89C1             <1>       mov    ecx, eax
4324 0000CAB2 6649             <1>       dec    cx ; FFFFh
4325                            <1>       ; CX = FFFFh -> find first deleted or free entry
4326                            <1>       ; ESI would be ASCIIZ filename address if the call
4327                            <1>       ; would not be for first free or deleted dir entry
4328 0000CAB4 E8D7E7FFFF       <1>       call   locate_current_dir_file
4329 0000CAB9 0F83EE000000     <1>       jnc    loc_createfile_set_ff_dir_entry
4330 0000CABF 5E               <1>       pop    esi ; *
4331                            <1>        ; ESI = Logical DOS Drv. Description Table Address
```

```
4332 0000CAC0 83F802              <1>       cmp    eax, 2
4333 0000CAC3 7402                <1>       je     short loc_createfile_add_new_cluster
4334                              <1> loc_createfile_locate_file_stc_retn:
4335 0000CAC5 F9                  <1>       stc
4336 0000CAC6 C3                  <1>       retn
4337                              <1>
4338                              <1> loc_createfile_add_new_cluster:
4339 0000CAC7 803D[858A0100]02    <1>       cmp    byte [Current_FATType], 2
4340                              <1>       ;cmp   byte [esi+LD_FATType], 2
4341 0000CACE 770C                <1>       ja     short loc_createfile_add_new_cluster_check_fsc
4342 0000CAD0 803D[848A0100]01    <1>       cmp    byte [Current_Dir_Level], 1
4343                              <1>       ;cmp   byte [esi+LD_CDirLevel], 1
4344 0000CAD7 7303                <1>       jnb    short loc_createfile_add_new_cluster_check_fsc
4345                              <1>
4346                              <1>       ;mov   eax, 12
4347 0000CAD9 B00C                <1>       mov    al, 12 ; No more files
4348                              <1>
4349                              <1> loc_createfile_anc_retn:
4350 0000CADB C3                  <1>       retn
4351                              <1>
4352                              <1> loc_createfile_add_new_cluster_check_fsc:
4353 0000CADC 8B0D[CC950100]      <1>       mov    ecx, [createfile_FreeSectors]
4354 0000CAE2 0FB605[E5950100]    <1>       movzx  eax, byte [createfile_SecPerClust]
4355 0000CAE9 66D1E0              <1>       shl    ax, 1 ; AX = 2 * AX
4356 0000CAEC 39C1                <1>       cmp    ecx, eax
4357 0000CAEE 7295                <1>         jb   short loc_createfile_insufficient_disk_space
4358                              <1>
4359                              <1> loc_createfile_add_new_subdir_cluster:
4360 0000CAF0 8B15[B1910100]      <1>       mov    edx, [DirBuff_Cluster]
4361 0000CAF6 8915[D8950100]      <1>       mov    [createfile_LastDirCluster], edx
4362                              <1>
4363 0000CAFC A1[D4950100]        <1>       mov    eax, [createfile_FFCluster]
4364 0000CB01 E846040000          <1>       call   load_FAT_sub_directory
4365 0000CB06 72D3                <1>       jc     short loc_createfile_anc_retn
4366                              <1>
4367                              <1> pass_createfile_add_new_subdir_cluster:
4368                              <1>       ;movzx eax, word [esi+LD_BPB+BytesPerSec]
4369 0000CB08 0FB705[EC950100]    <1>       movzx  eax, word [createfile_BytesPerSec] ; 23/03/2016
4370 0000CB0F F7E1                <1>       mul    ecx ; ecx = directory buffer sector count
4371 0000CB11 89C1                <1>       mov    ecx, eax
4372 0000CB13 C1E902              <1>       shr    ecx, 2 ; dword count
4373 0000CB16 29C0                <1>       sub    eax, eax ; 0
4374 0000CB18 F3AB                <1>       rep    stosd
4375                              <1>       ;
4376 0000CB1A C605[AC910100]02    <1>       mov    byte [DirBuff_ValidData], 2
4377 0000CB21 E890EFFFFF          <1>       call   save_directory_buffer
4378 0000CB26 72B3                <1>       jc     short loc_createfile_anc_retn
4379                              <1>
4380                              <1> loc_createfile_save_added_subdir_cluster:
4381 0000CB28 A1[D8950100]        <1>       mov    eax, [createfile_LastDirCluster]
4382 0000CB2D 8B0D[D4950100]      <1>       mov    ecx, [createfile_FFCluster]
4383 0000CB33 E858050000          <1>       call   update_cluster
4384 0000CB38 7304                <1>       jnc    short loc_createfile_save_fat_buffer_0
4385 0000CB3A 09C0                <1>       or     eax, eax ; EAX = 0 -> cluster value is 0 or eocc
4386 0000CB3C 751A                <1>       jnz    short loc_createfile_save_fat_buffer_stc_retn
4387                              <1>
4388                              <1> loc_createfile_save_fat_buffer_0:
4389 0000CB3E A1[D4950100]        <1>       mov    eax, [createfile_FFCluster]
4390 0000CB43 A3[D8950100]        <1>       mov    [createfile_LastDirCluster], eax
4391 0000CB48 B9FFFFFF0F          <1>       mov    ecx, 0FFFFFFFh ; 28 bit
4392 0000CB4D E83E050000          <1>       call   update_cluster
4393 0000CB52 7306                <1>       jnc    short loc_createfile_save_fat_buffer_1
4394 0000CB54 09C0                <1>       or     eax, eax ; Was it free cluster
4395 0000CB56 7402                <1>       jz     short loc_createfile_save_fat_buffer_1
4396                              <1>
4397                              <1> loc_createfile_save_fat_buffer_stc_retn:
4398 0000CB58 F9                  <1>       stc
4399                              <1> loc_createfile_save_fat_buffer_retn:
4400                              <1> loc_createfile_gffc_2_stc_retn:
4401 0000CB59 C3                  <1>       retn
4402                              <1>
4403                              <1> loc_createfile_save_fat_buffer_1:
4404                              <1>       ; byte [FAT_BuffValidData] = 2
4405 0000CB5A E8EE070000          <1>       call   save_fat_buffer
4406 0000CB5F 72F8                <1>       jc     short loc_createfile_save_fat_buffer_retn
4407                              <1>
4408 0000CB61 803D[A2910100]01    <1>       cmp    byte [FAT_ClusterCounter], 1
4409 0000CB68 7222                <1>       jb     short loc_createfile_save_fat_buffer_2
4410                              <1>
4411                              <1>       ; ESI = Logical DOS Drive Description Table address
4412 0000CB6A A1[A2910100]        <1>       mov    eax, [FAT_ClusterCounter]
4413                              <1>
4414 0000CB6F C605[A2910100]00    <1>       mov    byte [FAT_ClusterCounter], 0 ; 21/03/2016
4415                              <1>
4416 0000CB76 66BB01FF            <1>       mov    bx, 0FF01h ; add free clusters
4417 0000CB7A E863080000          <1>       call   calculate_fat_freespace
4418                              <1>
4419                              <1>       ;inc   eax ; 0FFFFFFFFh -> 0 ; recalculation is needed!
4420                              <1>       ;jnz   short loc_createfile_save_fat_buffer_2
4421                              <1>
4422                              <1>       ; ecx > 0 -> Recalculation is needed
4423 0000CB7F 09C9                <1>       or     ecx, ecx
4424 0000CB81 7409                <1>       jz     short loc_createfile_save_fat_buffer_2
4425                              <1>
4426 0000CB83 66BB00FF            <1>       mov    bx, 0FF00h ; ; recalculate free space
4427 0000CB87 E856080000          <1>       call   calculate_fat_freespace
4428                              <1>
4429                              <1> loc_createfile_save_fat_buffer_2:
4430                              <1>       ;call  update_parent_dir_lmdt
4431                              <1>
4432                              <1> loc_createfile_gffc_2:
4433 0000CB8C E82C040000          <1>       call   get_first_free_cluster
4434 0000CB91 72C6                <1>       jc     short loc_createfile_gffc_2_stc_retn
4435                              <1>
4436 0000CB93 A3[D4950100]        <1>       mov    [createfile_FFCluster], eax
```

```
4437                                    <1>
4438 0000CB98 A1[D8950100]              <1>       mov   eax, [createfile_LastDirCluster]
4439                                    <1>
4440 0000CB9D E8AA030000                <1>       call  load_FAT_sub_directory
4441 0000CBA2 72B5                      <1>       jc    short loc_createfile_gffc_2_stc_retn
4442                                    <1>
4443 0000CBA4 BF00000800                <1>       mov   edi, Directory_Buffer
4444                                    <1>
4445 0000CBA9 6629DB                    <1>       sub   bx, bx ; directory entry index/number = 0
4446                                    <1>
4447 0000CBAC 56                        <1>       push  esi ; * ; 23/03/2016
4448                                    <1>
4449                                    <1> loc_createfile_set_ff_dir_entry:
4450 0000CBAD 66891D[E6950100]          <1>       mov   [createfile_DirIndex], bx
4451                                    <1>
4452                                    <1>         ; EDI = Directory entry address
4453 0000CBB4 8B35[C8950100]            <1>       mov   esi, [createfile_Name_Offset]
4454 0000CBBA A1[D4950100]              <1>       mov   eax, [createfile_FFCluster]
4455 0000CBBF A3[DC950100]              <1>       mov   [createfile_Cluster], eax ; 24/03/2016
4456 0000CBC4 B5FF                      <1>       mov   ch, 0FFh
4457 0000CBC6 8A0D[E4950100]            <1>         mov cl, [createfile_attrib] ; file attributes
4458                                    <1>       ; CH > 0 -> File size is in [EBX]
4459 0000CBCC BB[D0950100]              <1>       mov   ebx, createfile_size
4460                                    <1>
4461 0000CBD1 E803EEFFFF                <1>       call  make_directory_entry
4462                                    <1>
4463 0000CBD6 5E                        <1>       pop   esi ; * ; ESI = Logical Dos Drv Desc. Table address
4464                                    <1>
4465 0000CBD7 C605[AC910100]02          <1>       mov   byte [DirBuff_ValidData], 2
4466 0000CBDE E8D3EEFFFF                <1>       call  save_directory_buffer
4467 0000CBE3 7221                      <1>       jc    short loc_createfile_set_ff_dir_entry_retn
4468                                    <1>
4469 0000CBE5 C605[EF950100]01          <1>       mov   byte [createfile_UpdatePDir], 1 ; 31/03/2016
4470                                    <1>
4471                                    <1> loc_createfile_get_set_write_file_cluster:
4472 0000CBEC A1[D0950100]              <1>       mov   eax, [createfile_size]
4473 0000CBF1 09C0                      <1>       or    eax, eax
4474 0000CBF3 7570                      <1>       jnz   short loc_createfile_get_set_wfc_cont
4475 0000CBF5 40                        <1>       inc   eax
4476                                    <1>       ; 23/03/2016
4477 0000CBF6 0FB61D[E5950100]          <1>       movzx ebx, byte [createfile_SecPerClust]
4478                                    <1>       ;movzx ecx, word [esi+LD_BPB+BytesPerSec] ; 512
4479 0000CBFD 0FB70D[EC950100]          <1>         movzx   ecx, word [createfile_BytesPerSec] ; 512
4480 0000CC04 EB7C                      <1>       jmp   loc_createfile_set_cluster_count
4481                                    <1>
4482                                    <1> loc_createfile_set_ff_dir_entry_retn:
4483 0000CC06 C3                        <1>       retn
4484                                    <1>
4485                                    <1> loc_createfile_write_fcluster_to_disk:
4486 0000CC07 034668                    <1>       add   eax, [esi+LD_DATABegin] ; convert to physical address
4487 0000CC0A BB00000700                <1>       mov   ebx, Cluster_Buffer
4488                                    <1>       ; ESI = Logical DOS Drv. Desc. Tbl. address
4489                                    <1>       ; EAX = Disk address
4490                                    <1>       ; EBX = Sector Buffer
4491                                    <1>       ; ECX = sectors per cluster
4492 0000CC0F E8285F0000                <1>       call  disk_write
4493 0000CC14 7211                      <1>       jc    short loc_createfile_dsk_wr_err
4494                                    <1>
4495                                    <1> loc_createfile_update_fat_cluster:
4496                                    <1>       ; 21/03/2016
4497 0000CC16 803D[EE950100]00          <1>       cmp   byte [createfile_wfc], 0
4498 0000CC1D 7712                      <1>       ja    short loc_createfile_update_fat_cluster_n1
4499                                    <1>
4500 0000CC1F FE05[EE950100]            <1>       inc   byte [createfile_wfc] ; 1
4501 0000CC25 EB24                      <1>       jmp   short loc_createfile_update_fat_cluster_n2
4502                                    <1>
4503                                    <1> loc_createfile_dsk_wr_err:
4504                                    <1>       ; 16/10/2016 (1Dh -> 18)
4505                                    <1>       ; 23/03/2016
4506 0000CC27 B812000000                <1>       mov   eax, 18 ; Drive not ready or write error !
4507 0000CC2C E9BD000000                <1>       jmp   loc_createfile_stc_retn
4508                                    <1>
4509                                    <1> loc_createfile_update_fat_cluster_n1:
4510 0000CC31 A1[E0950100]              <1>       mov   eax, [createfile_PCluster]
4511 0000CC36 8B0D[DC950100]            <1>       mov   ecx, [createfile_Cluster]
4512 0000CC3C E84F040000                <1>       call  update_cluster
4513 0000CC41 7308                      <1>       jnc   short loc_createfile_update_fat_cluster_n2
4514 0000CC43 09C0                      <1>       or    eax, eax ; EAX = 0 -> cluster value is 0 or eocc
4515 0000CC45 0F85A3000000              <1>       jnz   loc_createfile_stc_retn
4516                                    <1>
4517                                    <1> loc_createfile_update_fat_cluster_n2:
4518 0000CC4B A1[DC950100]              <1>         mov eax, [createfile_Cluster]
4519 0000CC50 B9FFFFFF0F                <1>       mov   ecx, 0FFFFFFFh
4520 0000CC55 E836040000                <1>       call  update_cluster
4521 0000CC5A 734E                      <1>       jnc   short loc_createfile_save_fat_buffer_3
4522 0000CC5C 09C0                      <1>       or    eax, eax ; EAX = 0 -> cluster value is 0 or eocc
4523 0000CC5E 744A                      <1>       jz    short loc_createfile_save_fat_buffer_3
4524                                    <1>
4525                                    <1> loc_createfile_upd_fat_fcluster_stc_retn:
4526 0000CC60 E989000000                <1>       jmp   loc_createfile_stc_retn
4527                                    <1>
4528                                    <1> loc_createfile_get_set_wfc_cont:
4529                                    <1>       ;movzx ecx, word [esi+LD_BPB+BytesPerSec] ; 512
4530 0000CC65 0FB70D[EC950100]          <1>       movzx ecx, word [createfile_BytesPerSec] ; 512
4531 0000CC6C 01C8                      <1>       add   eax, ecx
4532 0000CC6E 48                        <1>       dec   eax  ; add eax, 511
4533 0000CC6F 29D2                      <1>       sub   edx, edx
4534 0000CC71 F7F1                      <1>       div   ecx
4535 0000CC73 0FB61D[E5950100]          <1>       movzx ebx, byte [createfile_SecPerClust]
4536 0000CC7A 01D8                      <1>       add   eax, ebx
4537 0000CC7C 48                        <1>       dec   eax  ; add eax, SecPerClust - 1
4538 0000CC7D 6631D2                    <1>       xor   dx, dx
4539 0000CC80 F7F3                      <1>       div   ebx
4540                                    <1>
4541                                    <1> loc_createfile_set_cluster_count:
```

```
4542 0000CC82 A3[E8950100]        <1>       mov    [createfile_CCount], eax
4543                              <1>
4544 0000CC87 BF00000700          <1>       mov    edi, Cluster_Buffer
4545 0000CC8C 89C8                <1>       mov    eax, ecx ; Bytes per Sector
4546 0000CC8E F7E3                <1>       mul    ebx ; Sectors per Cluster
4547                              <1>       ; EAX = Bytes per Cluster
4548 0000CC90 89C1                <1>       mov    ecx, eax
4549 0000CC92 C1E902              <1>       shr    ecx, 2 ; dword count
4550 0000CC95 31C0                <1>       xor    eax, eax
4551 0000CC97 F3AB                <1>       rep    stosd ; clear cluster buffer
4552                              <1>
4553 0000CC99 A1[DC950100]        <1>       mov    eax, [createfile_Cluster] ; 24/03/2016
4554                              <1>
4555 0000CC9E 89D9                <1>       mov    ecx, ebx
4556                              <1>
4557                              <1> loc_createfile_get_set_wf_fclust_cont:
4558 0000CCA0 83E802              <1>       sub    eax, 2
4559 0000CCA3 F7E1                <1>       mul    ecx
4560                              <1>       ; EAX = Logical DOS disk address (offset)
4561 0000CCA5 E95DFFFFFF          <1>         jmp     loc_createfile_write_fcluster_to_disk
4562                              <1>
4563                              <1> loc_createfile_save_fat_buffer_3:
4564                              <1>       ; byte [FAT_BuffValidData] = 2
4565 0000CCAA E89E060000          <1>       call   save_fat_buffer
4566 0000CCAF 723D                <1>       jc     loc_createfile_stc_retn
4567                              <1>
4568                              <1>       ; 21/03/2016
4569 0000CCB1 803D[A2910100]01    <1>       cmp    byte [FAT_ClusterCounter], 1
4570 0000CCB8 721B                <1>       jb     short loc_createfile_save_fat_buffer_4
4571                              <1>
4572                              <1>       ; ESI = Logical DOS Drive Description Table address
4573 0000CCBA A1[A2910100]        <1>       mov    eax, [FAT_ClusterCounter]
4574 0000CCBF 66BB01FF            <1>       mov    bx, 0FF01h ; add free clusters
4575 0000CCC3 E81A070000          <1>       call   calculate_fat_freespace
4576                              <1>
4577                              <1>       ;inc   eax ; 0FFFFFFFFh -> 0 ; recalculation is needed!
4578                              <1>       ;jnz   short loc_createfile_save_fat_buffer_4
4579                              <1>
4580                              <1>       ; ecx > 0 -> Recalculation is needed
4581 0000CCC8 09C9                <1>       or     ecx, ecx
4582 0000CCCA 7409                <1>       jz     short loc_createfile_save_fat_buffer_4
4583                              <1>
4584 0000CCCC 66BB00FF            <1>       mov    bx, 0FF00h ; ; recalculate free space
4585 0000CCD0 E80D070000          <1>       call   calculate_fat_freespace
4586                              <1>
4587                              <1> loc_createfile_save_fat_buffer_4:
4588 0000CCD5 FF0D[E8950100]      <1>       dec    dword [createfile_CCount]
4589                              <1>       ;jz    short loc_createfile_upd_dir_modif_date_time
4590 0000CCDB 743F                <1>       jz     short loc_createfile_stc_retn_cc ; 31/03/2016
4591                              <1>
4592                              <1> loc_createfile_get_set_write_next_cluster:
4593 0000CCDD E8DB020000          <1>       call   get_first_free_cluster
4594 0000CCE2 720A                <1>       jc     short loc_createfile_stc_retn
4595                              <1>
4596                              <1> loc_createfile_get_set_write_next_cluster_1:
4597 0000CCE4 83F8FF              <1>       cmp    eax, 0FFFFFFFFh
4598 0000CCE7 7213                <1>       jb     short loc_createfile_get_set_write_next_cluster_2
4599                              <1>
4600                              <1> loc_createfile_wnc_insufficient_disk_space:
4601 0000CCE9 B827000000          <1>       mov    eax, 27h ; Insufficient disk space
4602                              <1>
4603                              <1> loc_createfile_stc_retn:
4604 0000CCEE 803D[EE950100]01    <1>       cmp    byte [createfile_wfc], 1
4605 0000CCF5 7324                <1>       jnb    short loc_createfile_err_retn
4606 0000CCF7 C3                  <1>       retn
4607                              <1>
4608                              <1> loc_createfile_wnc_inv_format_retn:
4609                              <1>       ;mov   eax, 28
4610 0000CCF8 B01C                <1>       mov    al, 28 ; Invalid format
4611 0000CCFA EBF2                <1>       jmp    short loc_createfile_stc_retn
4612                              <1>
4613                              <1> loc_createfile_get_set_write_next_cluster_2:
4614 0000CCFC 83F802              <1>       cmp    eax, 2
4615 0000CCFF 72F7                <1>       jb     short loc_createfile_wnc_inv_format_retn
4616                              <1>
4617                              <1> loc_createfile_get_set_write_next_cluster_3:
4618 0000CD01 8B0D[DC950100]      <1>       mov    ecx, [createfile_Cluster]
4619 0000CD07 A3[DC950100]        <1>       mov    [createfile_Cluster], eax
4620 0000CD0C 890D[E0950100]      <1>       mov    [createfile_PCluster], ecx
4621 0000CD12 0FB60D[E5950100]    <1>       movzx  ecx, byte [createfile_SecPerClust]
4622 0000CD19 EB85                <1>       jmp    short loc_createfile_get_set_wf_fclust_cont
4623                              <1>
4624                              <1> loc_createfile_err_retn:
4625 0000CD1B F9                  <1>       stc
4626                              <1>
4627                              <1> ;loc_createfile_upd_dir_modif_date_time:
4628                              <1> loc_createfile_stc_retn_cc: ; 31/03/2016
4629 0000CD1C 9C                  <1>       pushf  ; cpu is here for an error return or completion
4630 0000CD1D 50                  <1>       push   eax ; error code if cf = 1
4631                              <1>
4632                              <1>       ;call  update_parent_dir_lmdt
4633                              <1>
4634                              <1> ;loc_createfile_stc_retn_cc:
4635 0000CD1E A1[A2910100]        <1>       mov    eax, [FAT_ClusterCounter]
4636 0000CD23 09C0                <1>       or     eax, eax
4637 0000CD25 741A                <1>       jz     short loc_createfile_stc_retn_pop_eax
4638 0000CD27 8A3D[868A0100]      <1>       mov    bh, [Current_Drv]
4639 0000CD2D B301                <1>       mov    bl, 01h ; BL = 1 -> add clusters
4640                              <1>       ; NOTE: EAX value will be added to Free Cluster Count
4641                              <1>       ; (If EAX value is negative, Free Cluster Count will be decreased)
4642 0000CD2F E8AE060000          <1>       call   calculate_fat_freespace
4643                              <1>         ; ESI = Logical DOS Drive Description Table Address
4644                              <1>         ;jc  short loc_createfile_stc_retn_pop_eax_cf
4645 0000CD34 21C9                <1>       and    ecx, ecx ; cx = 0 -> valid free sector count
4646 0000CD36 7409                <1>       jz     short loc_createfile_stc_retn_pop_eax
```

```
4647                                <1>
4648                                <1> loc_createfile_stc_retn_recalc_FAT_freespace:
4649 0000CD38 66BB00FF             <1>         mov    bx, 0FF00h ; bh = 0FFh ->
4650                                <1>         ; ESI = Logical DOS Drv DT Addr
4651                                <1>         ; BL = 0 -> Recalculate
4652 0000CD3C E8A1060000           <1>         call   calculate_fat_freespace
4653                                <1>
4654                                <1> loc_createfile_stc_retn_pop_eax:
4655 0000CD41 58                   <1>         pop    eax
4656 0000CD42 9D                   <1>         popf
4657 0000CD43 7218                 <1>         jc     short loc_createfile_retn
4658                                <1>
4659                                <1> loc_createfile_retn_fcluster:
4660 0000CD45 A1[D4950100]         <1>         mov    eax, [createfile_FFCluster]
4661 0000CD4A BB[D0950100]         <1>         mov    ebx, createfile_size
4662                                <1>         ;movzx ecx, byte [esi+LD_BPB+SecPerClust]
4663 0000CD4F 0FB60D[E5950100]     <1>         movzx  ecx, byte [createfile_SecPerClust] ; 23/03/2016
4664 0000CD56 0FB715[E6950100]     <1>         movzx  edx, word [createfile_DirIndex]
4665                                <1>
4666                                <1> loc_createfile_retn:
4667 0000CD5D C3                   <1>         retn
4668                                <1>
4669                                <1> create_fs_file:
4670                                <1>         ; temporary (21/03/2016)
4671 0000CD5E C3                   <1>         retn
4672                                <1>
4673                                <1> delete_fs_file:
4674                                <1>         ; temporary (28/02/2016)
4675 0000CD5F C3                   <1>         retn
4676                                <1>
4677                                <1> rename_fs_file_or_directory:
4678 0000CD60 C3                   <1>         retn
4679                                <1>
4680                                <1> make_fs_directory:
4681                                <1>         ; temporary (21/02/2016)
4682 0000CD61 C3                   <1>         retn
4683                                <1>
4684                                <1> add_new_fs_section:
4685                                <1>         ; temporary (11/03/2016)
4686 0000CD62 C3                   <1>         retn
4687                                <1>
4688                                <1> delete_fs_directory_entry:
4689                                <1>         ; temporary (11/03/2016)
4690 0000CD63 C3                   <1>         retn
4691                                <1>
4692                                <1> csftdf2_read_fs_file_sectors:
4693                                <1>         ; temporary (19/03/2016)
4694 0000CD64 C3                   <1>         retn
4695                                <1>
4696                                <1> csftdf2_write_fs_file_sectors:
4697                                <1>         ; temporary (19/03/2016)
4698 0000CD65 C3                   <1>         retn
3083                                    %include 'trdosk5.s' ; 24/01/2016
   1                                <1> ; ****************************************************************************
   2                                <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - File System Procedures : trdosk5s
   3                                <1> ; ----------------------------------------------------------------------------
   4                                <1> ; Last Update: 23/10/2016
   5                                <1> ; ----------------------------------------------------------------------------
   6                                <1> ; Beginning: 24/01/2016
   7                                <1> ; ----------------------------------------------------------------------------
   8                                <1> ; Assembler: NASM version 2.11 (trdos386.s)
   9                                <1> ; ----------------------------------------------------------------------------
  10                                <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
  11                                <1> ; DRV_FAT.ASM (21/08/2011)
  12                                <1> ; ****************************************************************************
  13                                <1> ; DRV_FAT.ASM (c) 2005-2011 Erdogan TAN [ 07/07/2009 ] Last Update: 21/08/2011
  14                                <1>
  15                                <1> get_next_cluster:
  16                                <1>         ; 15/10/2016
  17                                <1>         ; 23/03/2016
  18                                <1>         ; 01/02/2016 (TRDOS 386 =  TRDOS v2.0)
  19                                <1>         ; 05/07/2011
  20                                <1>         ; 07/07/2009
  21                                <1>         ; 2005
  22                                <1>         ; INPUT ->
  23                                <1>         ;      EAX = Cluster Number (32 bit)
  24                                <1>         ;      ESI = Logical DOS Drive Parameters Table
  25                                <1>         ; OUTPUT ->
  26                                <1>         ;      cf = 0 -> No Error, EAX valid
  27                                <1>         ;      cf = 1 & EAX = 0 -> End Of Cluster Chain
  28                                <1>         ;      cf = 1 & EAX > 0 -> Error
  29                                <1>         ;      ECX = Current/Previous cluster (if CF = 0)
  30                                <1>         ;      EAX = Next Cluster Number (32 bit)
  31                                <1>         ;
  32                                <1>         ; (Modified registers: EAX, ECX, EBX, EDX)
  33                                <1>
  34 0000CD66 A3[96910100]          <1>         mov    [FAT_CurrentCluster], eax
  35                                <1> check_next_cluster_fat_type:
  36 0000CD6B 29D2                  <1>         sub    edx, edx ; 0
  37 0000CD6D 807E0302              <1>           cmp     byte [esi+LD_FATType], 2
  38 0000CD71 7250                  <1>         jb     short get_FAT12_next_cluster
  39 0000CD73 0F87AF000000          <1>           ja      get_FAT32_next_cluster
  40                                <1> get_FAT16_next_cluster:
  41 0000CD79 BB00030000            <1>         mov    ebx, 300h ;768
  42 0000CD7E F7F3                  <1>         div    ebx
  43                                <1>         ; EAX = Count of 3 FAT sectors
  44                                <1>         ; EDX = Cluster Offset (< 768)
  45 0000CD80 66D1E2                <1>         shl    dx, 1 ; Multiply by 2
  46 0000CD83 89D3                  <1>         mov    ebx, edx ; Byte Offset
  47 0000CD85 81C3001C0900          <1>         add    ebx, FAT_Buffer
  48 0000CD8B 66BA0300              <1>         mov    dx, 3
  49 0000CD8F F7E2                  <1>         mul    edx
  50                                <1>         ; EAX = FAT Sector (<= 256)
  51                                <1>         ; EDX = 0
  52 0000CD91 8A0E                  <1>         mov    cl, [esi+LD_Name]
```

```
0000CD93 803D[9A910100]00    <1>       cmp    byte [FAT_BuffValidData], 0
0000CD9A 0F86CC000000        <1>         jna     load_FAT_sectors0
0000CDA0 3A0D[9B910100]      <1>       cmp    cl, [FAT_BuffDrvName]
0000CDA6 0F85C0000000        <1>         jne     load_FAT_sectors0
0000CDAC 3B05[9E910100]      <1>       cmp    eax, [FAT_BuffSector]
0000CDB2 0F85BA000000        <1>         jne     load_FAT_sectors1
                             <1>       ;movzx eax, word [ebx]
0000CDB8 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
0000CDBB 6683F8F7            <1>       cmp    ax, 0FFF7h
0000CDBF 725A                <1>       jb     short loc_pass_gnc_FAT16_eoc_check
                             <1>       ; ax >= FFF7h (cluster 0002h to FFF6h is valid, in use)
0000CDC1 EB56                <1>       jmp    short loc_pass_gnc_FAT16_eoc_check_xor_eax
                             <1>
                             <1> get_FAT12_next_cluster:
0000CDC3 BB00040000          <1>       mov    ebx, 400h ;1024
0000CDC8 F7F3                <1>       div    ebx
                             <1>       ; EAX = Count of 3 FAT sectors
                             <1>       ; EDX = Cluster Offset (< 1024)
0000CDCA 6650                <1>       push   ax
0000CDCC 66B80300            <1>       mov    ax, 3
0000CDD0 66F7E2              <1>       mul    dx     ; Multiply by 3
0000CDD3 66D1E8              <1>       shr    ax, 1  ; Divide by 2
0000CDD6 6689C3              <1>         mov  bx, ax       ; Byte Offset
0000CDD9 81C3001C0900        <1>       add    ebx, FAT_Buffer
0000CDDF 6658                <1>       pop    ax
0000CDE1 66BA0300            <1>       mov    dx, 3
0000CDE5 F7E2                <1>       mul    edx
                             <1>       ; EAX = FAT Sector (<= 12)
                             <1>       ; EDX = 0
0000CDE7 8A0E                <1>       mov    cl, [esi+LD_Name]
0000CDE9 803D[9A910100]00    <1>       cmp    byte [FAT_BuffValidData], 0
0000CDF0 767A                <1>       jna    short load_FAT_sectors0
0000CDF2 3A0D[9B910100]      <1>       cmp    cl, [FAT_BuffDrvName]
0000CDF8 7572                <1>       jne    short load_FAT_sectors0
0000CDFA 3B05[9E910100]      <1>       cmp    eax, [FAT_BuffSector]
0000CE00 7570                <1>       jne    short load_FAT_sectors1
0000CE02 A1[96910100]        <1>       mov    eax, [FAT_CurrentCluster]
0000CE07 66D1E8              <1>       shr    ax, 1
                             <1>       ;movzx eax, word [ebx]
0000CE0A 668B03              <1>       mov    ax, [ebx]
0000CE0D 7314                <1>       jnc    short get_FAT12_nc_even
0000CE0F 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
0000CE13 663DF70F            <1>       cmp    ax, 0FF7h
0000CE17 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:
0000CE19 31C0                <1>       xor    eax, eax ; 0
                             <1> loc_pass_gnc_FAT16_eoc_check:
                             <1> loc_pass_gnc_FAT32_eoc_check:
0000CE1B 8B0D[96910100]      <1>       mov    ecx, [FAT_CurrentCluster]
0000CE21 F5                  <1>       cmc
0000CE22 C3                  <1>       retn
                             <1>
                             <1> get_FAT12_nc_even:
0000CE23 80E40F              <1>       and    ah, 0Fh
0000CE26 EBEB                <1>       jmp    short loc_gnc_fat12_eoc_check
                             <1>
                             <1> get_FAT32_next_cluster:
0000CE28 BB80010000          <1>       mov    ebx, 180h ;384
0000CE2D F7F3                <1>       div    ebx
                             <1>       ; EAX = Count of 3 FAT sectors
                             <1>       ; EDX = Cluster Offset (< 384)
0000CE2F 66C1E202            <1>       shl    dx, 2  ; Multiply by 4
0000CE33 89D3                <1>       mov    ebx, edx ; Byte Offset
0000CE35 81C3001C0900        <1>       add    ebx, FAT_Buffer
0000CE3B 66BA0300            <1>       mov    dx, 3
0000CE3F 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
0000CE41 8A0E                <1>       mov    cl, [esi+LD_Name]
0000CE43 803D[9A910100]00    <1>       cmp    byte [FAT_BuffValidData], 0
0000CE4A 7620                <1>       jna    short load_FAT_sectors0
0000CE4C 3A0D[9B910100]      <1>       cmp    cl, [FAT_BuffDrvName]
0000CE52 7518                <1>       jne    short load_FAT_sectors0
0000CE54 3B05[9E910100]      <1>       cmp    eax, [FAT_BuffSector] ; 0, 3, 6, 9 ...
0000CE5A 7516                <1>       jne    short load_FAT_sectors1
0000CE5C 8B03                <1>       mov    eax, [ebx]
0000CE5E 25FFFFFF0F          <1>       and    eax, 0FFFFFFFh ; 28 bit Cluster
0000CE63 3DF7FFFF0F          <1>       cmp    eax, 0FFFFFF7h
0000CE68 72B1                <1>       jb     short loc_pass_gnc_FAT32_eoc_check
                             <1>       ; eax >= FFFFFF7h (cluster 0002h to FFFFFF6h is valid)
0000CE6A EBAD                <1>       jmp    short loc_pass_gnc_FAT16_eoc_check_xor_eax
                             <1>
                             <1> load_FAT_sectors0:
0000CE6C 880D[9B910100]      <1>       mov    [FAT_BuffDrvName], cl
                             <1> load_FAT_sectors1:
0000CE72 A3[9E910100]        <1>       mov    [FAT_BuffSector], eax
0000CE77 89C3                <1>       mov    ebx, eax
0000CE79 034660              <1>         add     eax, [esi+LD_FATBegin]
```

```
0000CE7C 807E0302            <1>       cmp    byte [esi+LD_FATType], 2
0000CE80 7706                <1>         ja      short load_FAT_sectors3
0000CE82 0FB74E1C            <1>       movzx  ecx, word [esi+LD_BPB+BPB_FATSz16]
0000CE86 EB03                <1>       jmp    short load_FAT_sectors4
                             <1> load_FAT_sectors3:
0000CE88 8B4E2A              <1>       mov    ecx, [esi+LD_BPB+BPB_FATSz32]
                             <1> load_FAT_sectors4:
0000CE8B 29D9                <1>       sub    ecx, ebx ; [FAT_BuffSector]
0000CE8D 83F903              <1>         cmp     ecx, 3
0000CE90 7605                <1>         jna     short load_FAT_sectors5
0000CE92 B903000000          <1>       mov    ecx, 3
                             <1> load_FAT_sectors5:
0000CE97 BB001C0900          <1>       mov    ebx, FAT_Buffer
0000CE9C E8AA5C0000          <1>       call   disk_read
0000CEA1 730D                <1>       jnc    short load_FAT_sectors_ok
                             <1>       ; 15/10/2016 (15h -> 17)
                             <1>       ; 23/03/2016 (15h)
0000CEA3 B811000000          <1>       mov    eax, 17 ; Drive not ready or read error
0000CEA8 C605[9A910100]00    <1>       mov    byte [FAT_BuffValidData], 0
0000CEAF C3                  <1>       retn
                             <1> load_FAT_sectors_ok:
0000CEB0 C605[9A910100]01    <1>       mov    byte [FAT_BuffValidData], 1
0000CEB7 A1[96910100]        <1>       mov    eax, [FAT_CurrentCluster]
0000CEBC 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>
0000CEC1 8A1E                <1>       mov    bl, [esi+LD_Name]
0000CEC3 8A7E03              <1>       mov    bh, [esi+LD_FATType]
                             <1>
                             <1>       ;mov   [DirBuff_DRV], bl
                             <1>       ;mov   [DirBuff_FATType], bh
0000CEC6 66891D[AA910100]    <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
0000CECD 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>
0000CED1 6681FA0002          <1>       cmp    dx, 512 ; Number of Root Dir Entries
0000CED6 7414                <1>       je     short lrd_mov_ecx_32
0000CED8 89D0                <1>       mov    eax, edx
                             <1>       ; 23/10/2016
0000CEDA 89C1                <1>       mov    ecx, eax
0000CEDC 6683C10F            <1>       add    cx, 15 ; round up
0000CEE0 66C1E904            <1>       shr    cx, 4  ; 16 entries per sector (512/32)
                             <1>       ; ecx = Root directory size in sectors
0000CEE4 66C1E005            <1>       shl    ax, 5 ; Root directory size in bytes
0000CEE8 664A                <1>       dec    dx    ; Last entry number of root dir
                             <1>       ; cx = Dir Buffer sector count
0000CEEA EB0B                <1>       jmp    short lrd_check_dir_buffer
                             <1>
                             <1> lrd_mov_ecx_32:
0000CEEC B920000000          <1>       mov    ecx, 32
0000CEF1 664A                <1>       dec    dx ; 511
0000CEF3 66B80040            <1>       mov    ax, 32*512
                             <1>
                             <1> lrd_check_dir_buffer:
0000CEF7 29DB                <1>       sub    ebx, ebx ; 0
0000CEF9 881D[AC910100]      <1>       mov    [DirBuff_ValidData], bl ; 0
0000CEFF 668915[AF910100]    <1>       mov    [DirBuff_LastEntry], dx
0000CF06 891D[B1910100]      <1>       mov    [DirBuff_Cluster], ebx ; 0
0000CF0C 66A3[B5910100]      <1>       mov    [DirBuffer_Size], ax
                             <1>
0000CF12 8B4664              <1>       mov    eax, [esi+LD_ROOTBegin]
                             <1> read_directory:
0000CF15 BB00000800          <1>       mov    ebx, Directory_Buffer
0000CF1A 51                  <1>       push   ecx ; Directory buffer sector count
0000CF1B 53                  <1>       push   ebx
0000CF1C E82A5C0000          <1>       call   disk_read
0000CF21 5B                  <1>       pop    ebx
0000CF22 720B                <1>       jc     short load_DirBuff_error
                             <1>
                             <1> validate_DirBuff_and_return:
0000CF24 59                  <1>       pop    ecx ; Number of loaded sectors
0000CF25 C605[AC910100]01    <1>       mov    byte [DirBuff_ValidData], 1
0000CF2C 31C0                <1>       xor    eax, eax ; 0 = no error
0000CF2E C3                  <1>       retn
                             <1>
```

```
                            <1> load_DirBuff_error:
0000CF2F 89C8               <1>       mov    eax, ecx ; remaining sectors
0000CF31 59                 <1>       pop    ecx ; sector count
0000CF32 29C1               <1>       sub    ecx, eax ; Number of loaded sectors
                            <1>       ; 15/10/2016 (15h -> 17)
0000CF34 B811000000         <1>       mov    eax, 17 ; DRV NOT READY OR READ ERROR !
0000CF39 F9                 <1>       stc
0000CF3A 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>
0000CF3B 8A1E               <1>       mov    bl, [esi+LD_Name]
0000CF3D 8A7E03             <1>       mov    bh, [esi+LD_FATType]
                            <1>
                            <1>       ;mov   [DirBuff_DRV], bl
                            <1>       ;mov   [DirBuff_FATType], bh
0000CF40 66891D[AA910100]   <1>       mov    [DirBuff_DRV], bx
                            <1>
                            <1> load_FAT_sub_dir0:
0000CF47 8B4632             <1>       mov    eax, [esi+LD_BPB+FAT32_RootFClust]
0000CF4A 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>
0000CF4C 8A1E               <1>       mov    bl, [esi+LD_Name]
0000CF4E 8A7E03             <1>       mov    bh, [esi+LD_FATType]
                            <1>
                            <1>       ;mov   [DirBuff_DRV], bl
                            <1>       ;mov   [DirBuff_FATType], bh
0000CF51 66891D[AA910100]   <1>       mov    [DirBuff_DRV], bx
                            <1>
                            <1> load_FAT_sub_dir0:
0000CF58 0FB64E13           <1>       movzx  ecx, byte [esi+LD_BPB+SecPerClust]
                            <1>
0000CF5C 882D[AC910100]     <1>       mov    [DirBuff_ValidData], ch ; 0
0000CF62 A3[B1910100]       <1>       mov    [DirBuff_Cluster], eax
                            <1>
0000CF67 0FB74611           <1>       movzx  eax, word [esi+LD_BPB+BytesPerSec]
0000CF6B F7E1               <1>       mul    ecx
0000CF6D C1E805             <1>       shr    eax, 5 ; directory entry count (dir size / 32)
0000CF70 6648               <1>       dec    ax ; last entry
0000CF72 66A3[AF910100]     <1>       mov    [DirBuff_LastEntry], ax
                            <1>
0000CF78 A1[B1910100]       <1>       mov    eax, [DirBuff_Cluster]
0000CF7D 83E802             <1>       sub    eax, 2
0000CF80 F7E1               <1>       mul    ecx
0000CF82 034668             <1>       add    eax, [esi+LD_DATABegin]
                            <1>       ; ecx = sector per cluster (dir buffer size = 32 sectors)
0000CF85 EB8E               <1>       jmp    short read_directory
                            <1>
                            <1> ; DRV_FS.ASM
                            <1>
                            <1> load_current_FS_directory:
0000CF87 C3                 <1>       retn
                            <1> load_FS_root_directory:
0000CF88 C3                 <1>       retn
                            <1> load_FS_sub_directory:
0000CF89 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>
0000CF8A 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
0000CF8E 807E0300               <1>       cmp    byte [esi+LD_FATType], 0
0000CF92 761C                   <1>       jna    short read_fs_cluster
                                <1>
                                <1> read_fat_file_sectors: ; 18/03/2016
0000CF94 83E802                 <1>       sub    eax, 2 ; Beginning cluster number is always 2
0000CF97 0FB65613               <1>       movzx  edx, byte [esi+LD_BPB+BPB_SecPerClust] ; 18/03/2016
0000CF9B F7E2                   <1>       mul    edx
0000CF9D 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>
0000CFA0 E8A65B0000             <1>       call   disk_read
0000CFA5 7306                   <1>       jnc    short rclust_retn
                                <1>
                                <1>       ; 15/10/2016 (15h -> 17)
0000CFA7 B811000000             <1>       mov    eax, 17 ; Drive not ready or read error !
0000CFAC C3                     <1>       retn
                                <1>
                                <1> rclust_retn:
0000CFAD 29C0                   <1>       sub    eax, eax ; 0
0000CFAF 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)
0000CFB0 B980000000             <1>       mov    ecx, 128 ; maximum count of sectors (before eof)
0000CFB5 E801000000             <1>       call   read_fs_sectors
0000CFBA C3                     <1>       retn
                                <1>
                                <1> read_fs_sectors:
                                <1>       ; 15/02/2016 (TRDOS 386 =  TRDOS v2.0)
0000CFBB F9                     <1>       stc
0000CFBC 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>
0000CFBD 8B4678                 <1>       mov    eax, [esi+LD_Clusters]
0000CFC0 40                     <1>       inc    eax ; add eax, 1
0000CFC1 A3[34940100]           <1>       mov    [gffc_last_free_cluster], eax
                                <1>
0000CFC6 31DB                   <1>       xor    ebx, ebx ; 0 ; 02/03/2016
                                <1>
0000CFC8 807E0302               <1>       cmp    byte [esi+LD_FATType], 2
0000CFCC 760E                   <1>       jna    short loc_gffc_get_first_fat_free_cluster0
                                <1>
                                <1> loc_gffc_get_first_fat32_free_cluster:
                                <1>       ; 02/03/2016
0000CFCE E844060000             <1>       call   get_fat32_fsinfo_sector_parms
0000CFD3 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)
0000CFD5 89D0                   <1>       mov    eax, edx ; FSI_Next_Free (First Free Cluster)
0000CFD7 83F8FF                 <1>       cmp    eax, 0FFFFFFFFh ; invalid (unknown) !
0000CFDA 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:
0000CFDC B802000000             <1>       mov    eax, 2
                                <1>       ;xor   edx, edx
                                <1>
                                <1> loc_gffc_get_first_fat_free_cluster1:
0000CFE1 53                     <1>       push   ebx ; 02/03/2016
```

```
                                <1>
                                <1> loc_gffc_get_first_fat_free_cluster2:
0000CFE2 A3[30940100]           <1>         mov     [gffc_first_free_cluster], eax
0000CFE7 A3[2C940100]           <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:
0000CFEC E875FDFFFF             <1>         call    get_next_cluster
0000CFF1 7307                   <1>         jnc     short loc_gffc_get_first_fat_free_cluster4
0000CFF3 09C0                   <1>         or      eax, eax
0000CFF5 740B                   <1>         jz      short loc_gffc_first_free_fat_cluster_next
0000CFF7 5B                     <1>         pop     ebx ; 02/03/2016
0000CFF8 F5                     <1>         cmc     ; stc
0000CFF9 C3                     <1>         retn
                                <1>
                                <1> loc_gffc_get_first_fat_free_cluster4:
0000CFFA 21C0                   <1>         and     eax, eax ; next cluster value
0000CFFC 7504                   <1>         jnz     short loc_gffc_first_free_fat_cluster_next
0000CFFE 89C8                   <1>         mov     eax, ecx ; current (previous cluster) value
0000D000 EB22                   <1>         jmp     short loc_gffc_check_for_set
                                <1>
                                <1> loc_gffc_first_free_fat_cluster_next:
0000D002 A1[2C940100]           <1>         mov     eax, [gffc_next_free_cluster]
0000D007 3B05[34940100]         <1>         cmp     eax, [gffc_last_free_cluster]
0000D00D 7308                   <1>         jnb     short retn_stc_from_get_first_free_cluster
                                <1> pass_gffc_last_cluster_eax_check:
0000D00F 40                     <1>         inc     eax ; add eax, 1
0000D010 A3[2C940100]           <1>         mov     [gffc_next_free_cluster], eax
0000D015 EBD5                   <1>         jmp     short loc_gffc_get_first_fat_free_cluster3
                                <1>
                                <1> retn_stc_from_get_first_free_cluster:
0000D017 A1[30940100]           <1>         mov     eax, [gffc_first_free_cluster]
0000D01C 83F802                 <1>         cmp     eax, 2
0000D01F 7709                   <1>         ja      short loc_gffc_check_previous_clusters
0000D021 29C0                   <1>         sub     eax, eax
0000D023 48                     <1>         dec     eax ; FFFFFFFFh
                                <1>
                                <1> loc_gffc_check_for_set:
                                <1>         ; 02/03/2016
0000D024 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>
0000D025 09DB                   <1>         or      ebx, ebx
0000D027 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:
0000D029 C3                     <1>         retn
                                <1>
                                <1> loc_gffc_check_previous_clusters:
0000D02A 48                     <1>         dec     eax ; sub eax, 1
0000D02B A3[34940100]           <1>         mov     [gffc_last_free_cluster], eax
0000D030 B802000000             <1>         mov     eax, 2
                                <1>         ;xor   edx, edx
0000D035 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'
0000D037 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>
0000D03A 813B52526141           <1>         cmp     dword [ebx], 41615252h
0000D040 7540                   <1>         jne     short loc_sffc_read_fsinfo_sector
0000D042 81BBE4010000727241-    <1>         cmp     dword [ebx+484], 61417272h
```

```
0000D04B 61                <1>
0000D04C 7534              <1>       jne    short loc_sffc_read_fsinfo_sector
                           <1>
0000D04E 3B83EC010000      <1>       cmp    eax, [ebx+492]  ; FSI_Next_Free
0000D054 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'
0000D056 8983EC010000      <1>       mov    [ebx+492], eax
0000D05C A1[44940100]      <1>       mov    eax, [CFS_FAT32FSINFOSEC]
0000D061 B901000000        <1>       mov    ecx, 1
0000D066 53                <1>       push   ebx
0000D067 E8D05A0000        <1>       call   disk_write
0000D06C 7208              <1>       jc      short loc_sffc_read_fsinfo_sector_err1
0000D06E 5B                <1>       pop    ebx
                           <1>
0000D06F 8B83EC010000      <1>       mov    eax, [ebx+492] ; First (Next) Free Cluster
                           <1>
                           <1> loc_sffc_retn:
0000D075 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:
0000D076 BB00000000        <1>       mov    ebx, 0
                           <1>       ; 15/10/2016 (1Dh -> 18)
                           <1>       ; 23/03/2016 (1Dh)
0000D07B B812000000        <1>       mov    eax, 18 ; Drive not ready or write error
                           <1>
                           <1> loc_sffc_read_fsinfo_sector_err2:
0000D080 5A                <1>       pop    edx
0000D081 C3                <1>       retn
                           <1>
                           <1> loc_sffc_read_fsinfo_sector:
0000D082 50                <1>       push   eax
                           <1>
0000D083 E88F050000        <1>       call   get_fat32_fsinfo_sector_parms
0000D088 72F6              <1>       jc     short loc_sffc_read_fsinfo_sector_err2
                           <1>
0000D08A 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)
0000D08B 39D0              <1>       cmp    eax, edx ; First free Cluster (eax = new value)
0000D08D 75C7              <1>       jne    short loc_sffc_write_fsinfo_sector
                           <1>
0000D08F 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>
0000D090 A3[96910100]      <1>       mov    [FAT_CurrentCluster], eax
0000D095 890D[38940100]    <1>       mov    [ClusterValue], ecx
                           <1>
                           <1> loc_update_cluster_check_fat_buffer:
0000D09B 8A1E              <1>       mov    bl, [esi+LD_Name]
0000D09D 381D[9B910100]    <1>       cmp    [FAT_BuffDrvName], bl
0000D0A3 741A              <1>       je     short loc_update_cluster_check_fat_type
0000D0A5 803D[9A910100]02  <1>       cmp    byte [FAT_BuffValidData], 2
0000D0AC 0F84C2000000      <1>         je      loc_uc_save_fat_buffer
                           <1>
                           <1> loc_uc_reset_fat_buffer_validation:
0000D0B2 C605[9A910100]00  <1>       mov    byte [FAT_BuffValidData], 0
                           <1>
                           <1> loc_uc_check_fat_type_reset_drvname:
0000D0B9 881D[9B910100]    <1>       mov    [FAT_BuffDrvName], bl
                           <1>
                           <1> loc_update_cluster_check_fat_type:
0000D0BF 29D2              <1>       sub    edx, edx ; 26/02/2016
0000D0C1 8A5E03            <1>       mov    bl, [esi+LD_FATType]
0000D0C4 83F802            <1>       cmp    eax, 2
0000D0C7 0F82BE000000      <1>         jb      update_cluster_inv_data
```

```
0000D0CD 80FB02             <1>       cmp    bl, 2
0000D0D0 0F877A010000       <1>         ja      update_fat32_cluster
                            <1>       ;cmp   bl, 1
                            <1>       ;jb    short update_cluster_inv_data
0000D0D6 8B4E78             <1>       mov    ecx, [esi+LD_Clusters]
0000D0D9 41                 <1>       inc    ecx
0000D0DA 890D[A6910100]     <1>       mov    [LastCluster], ecx
0000D0E0 39C8               <1>       cmp    eax, ecx ; dword [LastCluster]
0000D0E2 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:')
0000D0E8 80FB01             <1>       cmp    bl, 1 ; correct comparison is this !
0000D0EB 0F86A2000000       <1>         jna     update_fat12_cluster
                            <1>
                            <1> update_fat16_cluster:
                            <1> pass_uc_fat16_errc:
                            <1>       ;sub   edx, edx
0000D0F1 BB00030000         <1>       mov    ebx, 300h ;768
0000D0F6 F7F3               <1>       div    ebx
                            <1>       ; EAX = Count of 3 FAT sectors
                            <1>       ; DX = Cluster offset in FAT buffer
0000D0F8 6689D3             <1>       mov    bx, dx
0000D0FB 66D1E3             <1>       shl    bx, 1 ; Multiply by 2
0000D0FE 66BA0300           <1>       mov    dx, 3
0000D102 F7E2               <1>       mul    edx
                            <1>       ; EAX = FAT Sector
                            <1>       ; EDX = 0
                            <1>       ; EBX = Byte offset in FAT buffer
0000D104 8A0D[9A910100]     <1>       mov    cl, [FAT_BuffValidData]
0000D10A 80F902             <1>       cmp    cl, 2
0000D10D 750A               <1>       jne    short loc_uc_check_fat16_buff_sector_load
                            <1>
                            <1> loc_uc_check_fat16_buff_sector_save:
0000D10F 3B05[9E910100]     <1>       cmp    eax, [FAT_BuffSector]
0000D115 755D               <1>       jne    short loc_uc_save_fat_buffer
0000D117 EB15               <1>       jmp    short loc_update_fat16_cell
                            <1>
                            <1> loc_uc_check_fat16_buff_sector_load:
0000D119 80F901             <1>       cmp    cl, 1 ; byte [FAT_BuffValidData]
0000D11C 0F85FB010000       <1>         jne     loc_uc_load_fat_sectors
0000D122 3B05[9E910100]     <1>       cmp    eax, [FAT_BuffSector]
0000D128 0F85EF010000       <1>         jne     loc_uc_load_fat_sectors
                            <1>
                            <1> loc_update_fat16_cell:
                            <1> loc_update_fat16_buffer:
0000D12E 81C3001C0900       <1>       add    ebx, FAT_Buffer ; 26/02/2016
                            <1>       ;movzx eax, word [ebx]
0000D134 668B03             <1>       mov    ax, [ebx]
                            <1>       ; 01/03/2016
0000D137 89C2               <1>       mov    edx, eax ; old value of the cluster
0000D139 A3[96910100]       <1>       mov    [FAT_CurrentCluster], eax
0000D13E 8B0D[38940100]     <1>       mov    ecx, [ClusterValue] ; 32 bits
0000D144 66890B             <1>       mov    [ebx], cx ; 16 bits !
                            <1>
0000D147 C605[9A910100]02   <1>       mov    byte [FAT_BuffValidData], 2
                            <1>
0000D14E 6683F802           <1>       cmp    ax, 2
0000D152 723A               <1>       jb     short return_uc_fat_stc
0000D154 3B05[A6910100]     <1>       cmp    eax, [LastCluster]
0000D15A 7732               <1>       ja     short return_uc_fat_stc
                            <1>
                            <1> loc_fat_buffer_updated:
                            <1>       ; 01/03/2016
0000D15C F8                 <1>       clc
                            <1> loc_fat_buffer_stc_1:
0000D15D 9C                 <1>       pushf
0000D15E 21C9               <1>       and    ecx, ecx
0000D160 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
0000D162 FF05[A2910100]     <1>       inc    dword [FAT_ClusterCounter]
                            <1>
                            <1> loc_fat_buffer_updated_1: ; new value of the cluster > 0
0000D168 09D2               <1>       or     edx, edx ; 02/03/2016
0000D16A 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
0000D16C FF0D[A2910100]     <1>       dec    dword [FAT_ClusterCounter] ; it may be negative number
                            <1>
                            <1> loc_fat_buffer_updated_2:
0000D172 9D                 <1>       popf
0000D173 C3                 <1>       retn
                            <1>
                            <1> loc_uc_save_fat_buffer:
                            <1>       ; byte [FAT_BuffValidData] = 2
0000D174 E8D4010000         <1>       call   save_fat_buffer
0000D179 0F8297010000       <1>         jc      loc_fat_sectors_rw_error2
                            <1>       ;mov   byte [FAT_BuffValidData], 1
0000D17F A1[96910100]       <1>       mov    eax, [FAT_CurrentCluster]
                            <1>       ;mov   ecx, [ClusterValue]
                            <1>       ;jmp   short loc_update_cluster_check_fat_buffer
0000D184 8A1E               <1>       mov    bl, [esi+LD_Name] ; 01/03/2016
0000D186 E927FFFFFF         <1>         jmp     loc_uc_reset_fat_buffer_validation
                            <1>
                            <1> update_cluster_inv_data:
                            <1>       ;mov   eax, 0Dh
0000D18B B00D               <1>       mov    al, 0Dh  ; Invalid Data
0000D18D C3                 <1>       retn
```

```
                                <1>
                                <1> return_uc_fat_stc:
                                <1>       ; 01/03/2016
0000D18E 31C0                   <1>       xor    eax, eax
0000D190 F9                     <1>       stc
0000D191 EBCA                   <1>       jmp    short loc_fat_buffer_stc_1
                                <1>
                                <1> update_fat12_cluster:
                                <1> pass_uc_fat12_errc:
                                <1>       ;sub   edx, edx
0000D193 BB00040000             <1>       mov    ebx, 400h ;1024
0000D198 F7F3                   <1>       div    ebx
                                <1>       ; EAX = Count of 3 FAT sectors
                                <1>       ; DX = Cluster offset in FAT buffer
0000D19A 66B90300               <1>       mov    cx, 3
0000D19E 6689C3                 <1>       mov    bx, ax
0000D1A1 6689C8                 <1>       mov    ax, cx ; 3
0000D1A4 66F7E2                 <1>       mul    dx     ; Multiply by 3
0000D1A7 66D1E8                 <1>       shr    ax, 1  ; Divide by 2
0000D1AA 6693                   <1>       xchg   bx, ax
                                <1>       ; EAX = Count of 3 FAT sectors
                                <1>       ; EBX = Byte Offset in FAT buffer
0000D1AC 66F7E1                 <1>       mul    cx  ; 3 * AX
                                <1>       ; EAX = FAT Beginning Sector
                                <1>       ; EDX = 0
0000D1AF 8A0D[9A910100]         <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 !)
0000D1B5 80F902                 <1>       cmp    cl, 2 ; 2 = dirty buffer (must be written to disk)
0000D1B8 750A                   <1>       jne    short loc_uc_check_fat12_buff_sector_load
                                <1>
                                <1> loc_uc_check_fat12_buff_sector_save:
0000D1BA 3B05[9E910100]         <1>       cmp    eax, [FAT_BuffSector]
0000D1C0 75B2                   <1>         jne     short loc_uc_save_fat_buffer
0000D1C2 EB15                   <1>       jmp    short loc_update_fat12_cell
                                <1>
                                <1> loc_uc_check_fat12_buff_sector_load:
0000D1C4 80F901                 <1>       cmp    cl, 1 ; byte ptr [FAT_BuffValidData]
0000D1C7 0F8550010000           <1>         jne     loc_uc_load_fat_sectors
0000D1CD 3B05[9E910100]         <1>       cmp    eax, [FAT_BuffSector]
0000D1D3 0F8544010000           <1>         jne     loc_uc_load_fat_sectors
                                <1>
                                <1> loc_update_fat12_cell:
0000D1D9 81C3001C0900           <1>       add    ebx, FAT_Buffer ; 26/02/2016
0000D1DF 668B0D[96910100]       <1>       mov    cx, [FAT_CurrentCluster]
0000D1E6 66D1E9                 <1>       shr    cx, 1
0000D1E9 668B03                 <1>       mov    ax, [ebx]
0000D1EC 6689C2                 <1>       mov    dx, ax
0000D1EF 7344                   <1>       jnc    short uc_fat12_nc_even
                                <1>
0000D1F1 6683E00F               <1>       and    ax, 0Fh
0000D1F5 8B0D[38940100]         <1>       mov    ecx, [ClusterValue] ; 32 bits
0000D1FB 66C1E104               <1>       shl    cx, 4
0000D1FF 6609C1                 <1>       or     cx, ax
0000D202 6689D0                 <1>       mov    ax, dx
0000D205 66890B                 <1>       mov    [ebx], cx  ; 16 bits !
0000D208 66C1E804               <1>       shr    ax, 4 ; al(bit4..7)+ah(bit0..7)
                                <1>
                                <1> update_fat12_buffer:
0000D20C A3[96910100]           <1>       mov    [FAT_CurrentCluster], eax
0000D211 89C2                   <1>       mov    edx, eax ; 01/03/2016
0000D213 C605[9A910100]02       <1>       mov    byte [FAT_BuffValidData], 2
0000D21A 6683F802               <1>       cmp    ax, 2
0000D21E 0F826AFFFFFF           <1>         jb      return_uc_fat_stc
0000D224 3B05[A6910100]         <1>       cmp    eax, [LastCluster]
0000D22A 0F875EFFFFFF           <1>         ja      return_uc_fat_stc
0000D230 E927FFFFFF             <1>         jmp     loc_fat_buffer_updated
                                <1>
                                <1> uc_fat12_nc_even:
0000D235 662500F0               <1>       and    ax, 0F000h
0000D239 8B0D[38940100]         <1>       mov    ecx, [ClusterValue] ; 32 bits
0000D23F 80E50F                 <1>       and    ch, 0Fh
0000D242 6609C1                 <1>       or     cx, ax
0000D245 6689D0                 <1>       mov    ax, dx
0000D248 66890B                 <1>       mov    [ebx], cx ; 16 bits !
0000D24B 80E40F                 <1>       and    ah, 0Fh ; al(bit0..7)+ah(bit0..3)
0000D24E EBBC                   <1>       jmp    short update_fat12_buffer
                                <1>
                                <1> update_fat32_cluster:
0000D250 8B4E78                 <1>       mov    ecx, [esi+LD_Clusters]
0000D253 41                     <1>       inc    ecx
0000D254 890D[A6910100]         <1>       mov    [LastCluster], ecx
                                <1>
0000D25A 39C8                   <1>       cmp    eax, ecx
0000D25C 0F872CFFFFFF           <1>         ja      return_uc_fat_stc
                                <1>
                                <1> pass_uc_fat32_errc:
                                <1>       ;sub   edx, edx
0000D262 BB80010000             <1>       mov    ebx, 180h ;384
0000D267 F7F3                   <1>       div    ebx
                                <1>       ; EAX = Count of 3 FAT sectors
                                <1>       ; DX = Cluster offset in FAT buffer
0000D269 89D3                   <1>       mov    ebx, edx
0000D26B C1E302                 <1>       shl    ebx, 2 ; Multiply by 4
0000D26E BA03000000             <1>       mov    edx, 3
0000D273 F7E2                   <1>       mul    edx
                                <1>       ; EBX = Cluster Offset in FAT buffer
                                <1>       ; EAX = FAT Sector
                                <1>       ; EDX = 0
0000D275 8A0D[9A910100]         <1>       mov    cl, [FAT_BuffValidData]
0000D27B 80F902                 <1>       cmp    cl, 2
0000D27E 750E                   <1>       jne    short loc_uc_check_fat32_buff_sector_load
                                <1>
                                <1> loc_uc_check_fat32_buff_sector_save:
```

```
0000D280 3B05[9E910100]      <1>       cmp    eax, [FAT_BuffSector]
0000D286 0F85E8FEFFFF        <1>         jne     loc_uc_save_fat_buffer
0000D28C EB11                <1>       jmp    short loc_update_fat32_cell
                             <1>
                             <1> loc_uc_check_fat32_buff_sector_load:
0000D28E 80F901              <1>       cmp    cl, 1 ; byte [FAT_BuffValidData]
0000D291 0F8586000000        <1>         jne     loc_uc_load_fat_sectors
0000D297 3B05[9E910100]      <1>       cmp    eax, [FAT_BuffSector]
0000D29D 757E                <1>         jne     loc_uc_load_fat_sectors
                             <1>
                             <1> loc_update_fat32_cell:
                             <1> loc_update_fat32_buffer:
0000D29F 81C3001C0900        <1>       add    ebx, FAT_Buffer ; 26/02/2016
0000D2A5 8B03                <1>       mov    eax, [ebx]
0000D2A7 25FFFFFF0F          <1>       and    eax, 0FFFFFFFh ; 28 bit cluster value
                             <1>
0000D2AC 8B15[96910100]      <1>       mov    edx, [FAT_CurrentCluster] ; 01/03/2016
                             <1>
0000D2B2 A3[96910100]        <1>       mov    [FAT_CurrentCluster], eax
0000D2B7 8B0D[38940100]      <1>       mov    ecx, [ClusterValue]
0000D2BD 890B                <1>       mov    [ebx], ecx ; 29/02/2016
                             <1>
0000D2BF C605[9A910100]02    <1>       mov    byte [FAT_BuffValidData], 2
                             <1>
                             <1>       ; 01/03/2016
0000D2C6 21C0                <1>       and    eax, eax ; was it free cluster ?
0000D2C8 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>
0000D2CA 3B563E              <1>       cmp    edx, [esi+LD_BPB+BPB_Reserved+4] ; First free cluster
0000D2CD 7520                <1>       jne    short loc_upd_fat32_c3
                             <1>
0000D2CF 3B15[A6910100]      <1>       cmp    edx, [LastCluster]
0000D2D5 7207                <1>       jb     short loc_upd_fat32_c0
                             <1>
0000D2D7 BA02000000          <1>       mov    edx, 2 ; rewind !
0000D2DC EB0E                <1>       jmp    short loc_upd_fat32_c2
                             <1>
                             <1> loc_upd_fat32_c0:
0000D2DE FF463E              <1>       inc    dword [esi+LD_BPB+BPB_Reserved+4] ; set it to next cluster
0000D2E1 EB0C                <1>       jmp    short loc_upd_fat32_c3
                             <1>
                             <1> loc_upd_fat32_c1:
0000D2E3 09C9                <1>       or     ecx, ecx ; will it be free cluster ?
0000D2E5 7508                <1>       jnz    short loc_upd_fat32_c3
                             <1>
0000D2E7 3B563E              <1>       cmp    edx, [esi+LD_BPB+BPB_Reserved+4] ; First free cluster
0000D2EA 7303                <1>       jnb    short loc_upd_fat32_c3
                             <1>
                             <1> loc_upd_fat32_c2:
0000D2EC 89563E              <1>       mov    [esi+LD_BPB+BPB_Reserved+4], edx
                             <1>
                             <1> loc_upd_fat32_c3:
0000D2EF 89C2                <1>       mov    edx, eax
                             <1>
                             <1> loc_upd_fat32_c4:
0000D2F1 83F802              <1>       cmp    eax, 2
0000D2F4 0F8294FEFFFF        <1>         jb      return_uc_fat_stc
                             <1>
                             <1> pass_uc_fat32_c_zero_check_2:
0000D2FA 3B05[A6910100]      <1>       cmp    eax, [LastCluster]
0000D300 0F8788FEFFFF        <1>         ja      return_uc_fat_stc
                             <1>
0000D306 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
0000D30B B811000000          <1>       mov    eax, 17 ; Drive not ready or read error
0000D310 8825[9A910100]      <1>       mov    [FAT_BuffValidData], ah ; 0
                             <1>
                             <1> loc_fat_sectors_rw_error2:
                             <1>       ;mov   eax, error code
                             <1>       ;mov   edx, 0
0000D316 8B0D[38940100]      <1>       mov    ecx, [ClusterValue]
0000D31C C3                  <1>       retn
                             <1>
                             <1> loc_uc_load_fat_sectors:
0000D31D A3[9E910100]        <1>       mov    [FAT_BuffSector], eax
                             <1>
                             <1> load_uc_fat_sectors_zero:
0000D322 034660              <1>       add    eax, [esi+LD_FATBegin]
0000D325 BB001C0900          <1>       mov    ebx, FAT_Buffer
0000D32A B903000000          <1>       mov    ecx, 3
0000D32F E817580000          <1>       call   disk_read
0000D334 72D5                <1>       jc     short loc_fat_sectors_rw_error1
                             <1>
0000D336 C605[9A910100]01    <1>         mov     byte [FAT_BuffValidData], 1
0000D33D A1[96910100]        <1>       mov    eax, [FAT_CurrentCluster]
0000D342 8B0D[38940100]      <1>       mov    ecx, [ClusterValue]
0000D348 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.
```

```
 997                                    <1>       ;      cf = 1 -> error code in AL (EAX)
 998                                    <1>       ;
 999                                    <1>       ;      EBX = FAT_Buffer address
1000                                    <1>       ;
1001                                    <1>       ; (EAX, EDX, ECX will be modified)
1002                                    <1>
1003                                    <1>       ;cmp   byte [FAT_BuffValidData], 2
1004                                    <1>       ;je    short loc_save_fat_buff
1005                                    <1>
1006                                    <1> ;loc_save_fat_buffer_retn:
1007                                    <1> ;     xor    eax, eax
1008                                    <1> ;     retn
1009                                    <1>
1010                                    <1> loc_save_fat_buff:
1011 0000D34D 31D2                     <1>       xor    edx, edx
1012 0000D34F 8A35[9B910100]           <1>       mov    dh, [FAT_BuffDrvName]
1013 0000D355 80FE41                   <1>       cmp    dh, 'A'
1014 0000D358 722E                     <1>       jb     short loc_save_fat_buffer_inv_data_retn
1015 0000D35A 80EE41                   <1>       sub    dh, 'A'
1016 0000D35D 56                       <1>       push   esi ; *
1017 0000D35E BE00010900               <1>         mov     esi, Logical_DOSDisks
1018 0000D363 01D6                     <1>       add    esi, edx
1019                                    <1>
1020 0000D365 8A5603                   <1>       mov    dl, [esi+LD_FATType]
1021 0000D368 20D2                     <1>       and    dl, dl
1022 0000D36A 741B                     <1>       jz     short loc_save_fat_buffer_inv_data_pop_retn
1023                                    <1>
1024 0000D36C A1[9E910100]             <1>       mov    eax, [FAT_BuffSector]
1025 0000D371 80FA02                   <1>       cmp    dl, 2
1026 0000D374 770A                     <1>       ja     short loc_save_fat32_buff
1027                                    <1>
1028                                    <1> loc_save_fat_12_16_buff:
1029                                    <1>       ; 01/03/2016
1030                                    <1>       ; TRDOS v1 has a FATal bug here!
1031                                    <1>       ; Correct code: mov dx, word ptr [FAT_BuffSector]+2
1032                                    <1>       ; (DX:AX in TRDOS v1 -> EAX in TRDOS v2)
1033                                    <1>       ;
1034 0000D376 0FB74E1C                 <1>       movzx  ecx, word [esi+LD_BPB+FATSecs]
1035 0000D37A 29C1                     <1>       sub    ecx, eax
1036                                    <1>       ; TRDOS v1 has a bug here... ('pop esi' was forgotten!)
1037                                    <1>       ;jna   short loc_save_fat_buffer_inv_data_retn ; wrong addr!
1038 0000D37C 7609                     <1>       jna    short loc_save_fat_buffer_inv_data_pop_retn ; correct addr.
1039 0000D37E EB15                     <1>       jmp    short loc_save_fat_buffer_check_rs3
1040                                    <1>
1041                                    <1> loc_save_fat32_buff:
1042 0000D380 8B4E2A                   <1>       mov    ecx, [esi+LD_BPB+FAT32_FAT_Size]
1043 0000D383 29C1                     <1>       sub    ecx, eax
1044 0000D385 770E                     <1>       ja     short loc_save_fat_buffer_check_rs3
1045                                    <1>
1046                                    <1> loc_save_fat_buffer_inv_data_pop_retn:
1047 0000D387 5E                       <1>       pop    esi ; *
1048                                    <1> loc_save_fat_buffer_inv_data_retn:
1049 0000D388 B80D000000               <1>       mov    eax, 0Dh ; Invalid DATA
1050 0000D38D C3                       <1>       retn
1051                                    <1>
1052                                    <1> loc_save_fat_buff_remain_sectors_3:
1053 0000D38E B903000000               <1>       mov    ecx, 3
1054 0000D393 EB05                     <1>       jmp    short loc_save_fat_buff_continue
1055                                    <1>
1056                                    <1> loc_save_fat_buffer_check_rs3:
1057 0000D395 83F903                   <1>       cmp    ecx, 3
1058 0000D398 77F4                     <1>       ja     short loc_save_fat_buff_remain_sectors_3
1059                                    <1>
1060                                    <1> loc_save_fat_buff_continue:
1061 0000D39A BB001C0900               <1>       mov    ebx, FAT_Buffer
1062 0000D39F 034660                   <1>       add    eax, [esi+LD_FATBegin]
1063 0000D3A2 51                       <1>       push   ecx
1064 0000D3A3 E894570000               <1>       call   disk_write
1065 0000D3A8 59                       <1>       pop    ecx
1066 0000D3A9 722B                     <1>       jc     short loc_save_FAT_buff_write_err
1067                                    <1>
1068 0000D3AB 807E0302                 <1>       cmp    byte [esi+LD_FATType], 2
1069 0000D3AF 7605                     <1>       jna    short loc_calc_2nd_fat12_16_addr
1070                                    <1>
1071                                    <1> loc_calc_2nd_fat32_addr:
1072 0000D3B1 8B462A                   <1>       mov    eax, [esi+LD_BPB+FAT32_FAT_Size]
1073 0000D3B4 EB04                     <1>       jmp    short loc_calc_2nd_fat_addr
1074                                    <1>
1075                                    <1> loc_calc_2nd_fat12_16_addr:
1076 0000D3B6 0FB7461C                 <1>       movzx  eax, word [esi+LD_BPB+FATSecs]
1077                                    <1>
1078                                    <1> loc_calc_2nd_fat_addr:
1079 0000D3BA 034660                   <1>       add    eax, [esi+LD_FATBegin]
1080 0000D3BD 0305[9E910100]           <1>       add    eax, [FAT_BuffSector]
1081 0000D3C3 BB001C0900               <1>       mov    ebx, FAT_Buffer
1082                                    <1>       ; ecx = 1 to 3
1083 0000D3C8 E86F570000               <1>       call   disk_write
1084 0000D3CD 7207                     <1>       jc     short loc_save_FAT_buff_write_err
1085                                    <1>       ; Valid  buffer (1 = valid but do not save)
1086 0000D3CF C605[9A910100]01         <1>       mov    byte [FAT_BuffValidData], 1
1087                                    <1>
1088                                    <1> loc_save_FAT_buff_write_err:
1089 0000D3D6 5E                       <1>       pop    esi ; *
1090 0000D3D7 BB001C0900               <1>       mov    ebx, FAT_Buffer
1091                                    <1>       ; 15/10/2016 (1Dh -> 18)
1092                                    <1>       ; 23/03/2016 (1Dh)
1093 0000D3DC B812000000               <1>       mov    eax, 18 ; Drive not ready or write error
1094 0000D3E1 C3                       <1>       retn
1095                                    <1>
1096                                    <1> calculate_fat_freespace:
1097                                    <1>       ; 23/03/2016
1098                                    <1>       ; 02/03/2016
1099                                    <1>       ; 01/03/2016
1100                                    <1>       ; 29/02/2016
1101                                    <1>       ; 22/02/2016 (TRDOS 386 =  TRDOS v2.0)
```

```
1102                                    <1>       ; 30/04/2011
1103                                    <1>       ; 03/04/2010
1104                                    <1>       ; 2005
1105                                    <1>       ; INPUT ->
1106                                    <1>       ;      EAX = Cluster count to be added or subtracted
1107                                    <1>       ;      If BH = FFh, ESI = TR-DOS Logical Drive Description Table
1108                                    <1>       ;      If BH < FFh, BH = TR-DOS Logical Drive Number
1109                                    <1>       ;      BL:
1110                                    <1>       ;      0 = Calculate, 1 = Add, 2 = Subtract, 3 = Get (Not Set/Calc)
1111                                    <1>       ; OUTPUT ->
1112                                    <1>       ;      EAX = Free Space in sectors
1113                                    <1>       ;      ESI = Logical Dos Drive Description Table address
1114                                    <1>       ;      BH = Logical Dos Drive Number (same with input value of BH)
1115                                    <1>       ;      BL = Type of operation (same with input value of BL)
1116                                    <1>       ;      ECX = 0 -> valid
1117                                    <1>       ;      ECX > 0 -> error or invalid
1118                                    <1>       ;      If EAX = FFFFFFFFh, it is 're-calculation needed'
1119                                    <1>       ;                             sign due to r/w error
1120                                    <1>
1121 0000D3E2 66891D[3E940100]         <1>       mov    [CFS_OPType], bx
1122 0000D3E9 A3[40940100]             <1>       mov    [CFS_CC], eax
1123                                    <1>
1124 0000D3EE 80FFFF                   <1>       cmp    bh, 0FFh
1125 0000D3F1 740B                     <1>       je     short pass_calculate_freespace_get_drive_dt_offset
1126                                    <1>
1127                                    <1> loc_calculate_freespace_get_drive_dt_offset:
1128 0000D3F3 31C0                     <1>       xor    eax, eax
1129 0000D3F5 88FC                     <1>         mov     ah, bh
1130 0000D3F7 BE00010900               <1>       mov    esi, Logical_DOSDisks
1131 0000D3FC 01C6                     <1>         add     esi, eax
1132                                    <1>
1133                                    <1> pass_calculate_freespace_get_drive_dt_offset:
1134 0000D3FE 08DB                     <1>       or     bl, bl
1135 0000D400 7435                     <1>       jz     short loc_reset_fcc
1136                                    <1>
1137                                    <1> loc_get_free_sectors:
1138 0000D402 8B4674                   <1>       mov    eax, [esi+LD_FreeSectors]
1139                                    <1>
1140                                    <1>       ;xor   ecx, ecx
1141                                    <1>       ;dec   ecx ; 0FFFFFFFFh
1142                                    <1>       ;cmp   eax, ecx ; 29/02/2016
1143                                    <1>       ;je    short loc_get_free_sectors_retn ; recalculation is needed!
1144                                    <1>
1145                                    <1>       ; 23/03/2016
1146 0000D405 8B4E70                   <1>       mov    ecx, [esi+LD_TotalSectors]
1147 0000D408 39C1                     <1>       cmp    ecx, eax ; Total sectors must be greater than Free sectors !
1148 0000D40A 7707                     <1>       ja     short loc_get_free_sectors_check_optype
1149                                    <1>
1150 0000D40C 31C0                     <1>       xor    eax, eax
1151 0000D40E 48                       <1>       dec    eax ; 0FFFFFFFFh  ; recalculation is needed!
1152 0000D40F 894674                   <1>       mov    [esi+LD_FreeSectors], eax ; reset (for recalculation)
1153                                    <1>
1154                                    <1> loc_get_free_sectors_retn:
1155 0000D412 C3                       <1>       retn
1156                                    <1>
1157                                    <1> loc_get_free_sectors_check_optype:
1158 0000D413 80FB03                   <1>       cmp    bl, 3
1159 0000D416 7203                     <1>       jb     short loc_set_fcc
1160                                    <1>
1161 0000D418 29C9                     <1>       sub    ecx, ecx ; 0
1162                                    <1>
1163 0000D41A C3                       <1>       retn
1164                                    <1>
1165                                    <1> loc_set_fcc:
1166 0000D41B 807E0302                 <1>       cmp    byte [esi+LD_FATType], 2
1167 0000D41F 0F87DF000000             <1>         ja      loc_update_FAT32_fs_info_fcc
1168                                    <1>
1169                                    <1>       ;mov   eax, [esi+LD_FreeSectors]
1170 0000D425 0FB64E13                 <1>       movzx  ecx, byte [esi+LD_BPB+SecPerClust]
1171 0000D429 29D2                     <1>       sub    edx, edx
1172 0000D42B F7F1                     <1>       div    ecx
1173                                    <1>       ;or    dx, dx
1174                                    <1>       ;      ; DX -> Remain sectors < SecPerClust
1175                                    <1>       ;      ; DX > 0 -> invalid free sector count
1176                                    <1>       ;jnz   short loc_reset_fcc
1177                                    <1>
1178                                    <1> ;pass_set_fcc_div32:
1179 0000D42D A3[B7910100]             <1>       mov    [FreeClusterCount], eax
1180 0000D432 E988000000               <1>         jmp     loc_set_free_sectors_FAT12_FAT16
1181                                    <1>
1182                                    <1> loc_reset_fcc:
1183 0000D437 31C0                     <1>       xor    eax, eax
1184 0000D439 A3[B7910100]             <1>       mov    [FreeClusterCount], eax ; 0
1185 0000D43E 8B5678                   <1>       mov    edx, [esi+LD_Clusters]
1186 0000D441 42                       <1>       inc    edx
1187 0000D442 8915[A6910100]           <1>       mov    [LastCluster], edx
1188                                    <1>
1189 0000D448 807E0302                 <1>       cmp    byte [esi+LD_FATType], 2
1190 0000D44C 7647                     <1>       jna    short loc_count_free_fat_clusters_0
1191                                    <1>
1192 0000D44E 48                       <1>       dec    eax ; FFFFFFFFh
1193 0000D44F A3[48940100]             <1>       mov    [CFS_FAT32FC], eax
1194                                    <1>
1195                                    <1>       ; 29/02/2016
1196 0000D454 89463A                   <1>       mov    [esi+LD_BPB+BPB_Reserved], eax ; reset
1197 0000D457 89463E                   <1>       mov    [esi+LD_BPB+BPB_Reserved+4], eax ; reset
1198                                    <1>
1199 0000D45A B802000000               <1>       mov    eax, 2
1200                                    <1>
1201                                    <1> loc_count_fc_next_cluster_0:
1202 0000D45F 50                       <1>       push   eax
1203 0000D460 E801F9FFFF               <1>       call   get_next_cluster
1204 0000D465 7310                     <1>       jnc    short loc_check_fat32_ff_cluster
1205 0000D467 09C0                     <1>       or     eax, eax
1206 0000D469 741E                     <1>       jz     short pass_inc_cfs_fcc_0
```

```
                                <1>
                                <1> loc_put_fcc_unknown_sign:
0000D46B 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!
0000D46C 8B15[48940100]         <1>       mov    edx, [CFS_FAT32FC] ; First Free Cluster
                                <1>       ; Save First Free Cluster value in FAT32 'BPB_Reserved+4' area
0000D472 89563E                 <1>       mov    [esi+LD_BPB+BPB_Reserved+4], edx
                                <1>
0000D475 EB7D                   <1>         jmp     loc_put_fcc_invalid_sign
                                <1>
                                <1> loc_check_fat32_ff_cluster:
0000D477 09C0                   <1>       or     eax, eax
0000D479 750E                   <1>       jnz    short pass_inc_cfs_fcc_0
0000D47B 58                     <1>       pop    eax
0000D47C A3[48940100]           <1>       mov    [CFS_FAT32FC], eax
                                <1>       ;mov   dword [FreeClusterCount], 1
0000D481 FF05[B7910100]         <1>       inc    dword [FreeClusterCount]
0000D487 EB27                   <1>       jmp    short pass_inc_cfs_fcc_1
                                <1>
                                <1> pass_inc_cfs_fcc_0:
0000D489 58                     <1>       pop    eax
                                <1>
                                <1> pass_inc_cfs_fcc_0c:
0000D48A 40                     <1>       inc    eax ; add eax, 1
0000D48B 3B05[A6910100]         <1>       cmp    eax, [LastCluster]
0000D491 76CC                   <1>       jna    short loc_count_fc_next_cluster_0
0000D493 EB6F                   <1>       jmp    short loc_update_FAT32_fs_info_fcc
                                <1>
                                <1> loc_count_free_fat_clusters_0:
                                <1>       ;mov   eax, 2
0000D495 B002                   <1>       mov    al, 2
                                <1>
                                <1> loc_count_fc_next_cluster:
0000D497 50                     <1>       push   eax
0000D498 E8C9F8FFFF             <1>       call   get_next_cluster
0000D49D 720C                   <1>       jc     short loc_count_fcc_stc
                                <1>
                                <1> loc_count_free_clusters_1:
0000D49F 21C0                   <1>       and    eax, eax
0000D4A1 750C                   <1>       jnz    short pass_inc_cfs_fcc
                                <1>
0000D4A3 FF05[B7910100]         <1>       inc    dword [FreeClusterCount]
0000D4A9 EB04                   <1>       jmp    short pass_inc_cfs_fcc
                                <1>
                                <1> loc_count_fcc_stc:
0000D4AB 09C0                   <1>       or     eax, eax
0000D4AD 75BC                   <1>       jnz    short loc_put_fcc_unknown_sign ; 29/02/2016
                                <1>
                                <1> pass_inc_cfs_fcc:
0000D4AF 58                     <1>       pop    eax
                                <1>
                                <1> pass_inc_cfs_fcc_1:
0000D4B0 40                     <1>       inc    eax ; add eax, 1
0000D4B1 3B05[A6910100]         <1>       cmp    eax, [LastCluster]
0000D4B7 76DE                   <1>       jna    short loc_count_fc_next_cluster
                                <1>
                                <1> loc_set_free_sectors:
0000D4B9 807E0302               <1>       cmp    byte [esi+LD_FATType], 2
0000D4BD 7745                   <1>       ja     short loc_update_FAT32_fs_info_fcc
                                <1>
                                <1> loc_set_free_sectors_FAT12_FAT16:
0000D4BF 803D[3E940100]00       <1>       cmp    byte [CFS_OPType], 0
0000D4C6 761C                   <1>       jna    short pass_FAT_add_sub_fcc
0000D4C8 A1[40940100]           <1>       mov    eax, [CFS_CC]
0000D4CD 803D[3E940100]01       <1>       cmp    byte [CFS_OPType], 1
0000D4D4 7708                   <1>       ja     short pass_FAT_add_fcc
0000D4D6 0105[B7910100]         <1>       add    [FreeClusterCount], eax
0000D4DC EB06                   <1>       jmp    short pass_FAT_add_sub_fcc
                                <1>
                                <1> pass_FAT_add_fcc:
0000D4DE 2905[B7910100]         <1>       sub    [FreeClusterCount], eax
                                <1>
                                <1> pass_FAT_add_sub_fcc:
0000D4E4 0FB64613               <1>       movzx  eax, byte [esi+LD_BPB+SecPerClust]
0000D4E8 8B15[B7910100]         <1>       mov    edx, [FreeClusterCount]
0000D4EE F7E2                   <1>       mul    edx
                                <1>
0000D4F0 31C9                   <1>       xor    ecx, ecx
0000D4F2 EB05                   <1>       jmp    short loc_cfs_retn_params
                                <1>
                                <1> loc_put_fcc_invalid_sign:
0000D4F4 29C0                   <1>             sub    eax, eax ; 0
0000D4F6 48                     <1>       dec    eax ; FFFFFFFFh
                                <1> loc_fat32_ffc_recalc_needed:
0000D4F7 89C1                   <1>       mov    ecx, eax
                                <1>
                                <1> loc_cfs_retn_params:
0000D4F9 894674                 <1>       mov    [esi+LD_FreeSectors], eax
0000D4FC 0FB71D[3E940100]       <1>       movzx  ebx, word [CFS_OPType]
0000D503 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
0000D504 803D[3E940100]01       <1>       cmp    byte [CFS_OPType], 1
0000D50B 7221                   <1>       jb     short loc_cfs_FAT32_get_rcalc_parms ; 0 = recalculated
0000D50D 7406                   <1>       je     short loc_check_fcc_FSINFO_op1 ; 1 = add
                                <1> loc_check_fcc_FSINFO_op2: ; subtract
```

```
1312 0000D50F F71D[40940100]      <1>       neg    dword [CFS_CC] ; prepare to subtract ; 2 = sub (add negative)
1313                              <1> loc_check_fcc_FSINFO_op1:
1314                              <1>       ; 01/03/2016
1315 0000D515 31D2                <1>       xor    edx, edx ; 0
1316 0000D517 4A                  <1>       dec    edx ; 0FFFFFFFFh
1317 0000D518 8B463A              <1>       mov    eax, [esi+LD_BPB+BPB_Reserved]
1318 0000D51B 39D0                <1>       cmp    eax, edx
1319 0000D51D 73D5                <1>       jnb    short loc_put_fcc_invalid_sign
1320 0000D51F 0305[40940100]      <1>         add     eax, [CFS_CC] ; free cluster count on disk + current count
1321 0000D525 72CD                <1>       jc     short loc_put_fcc_invalid_sign
1322                              <1>
1323 0000D527 A3[B7910100]        <1>       mov    [FreeClusterCount], eax
1324 0000D52C EB0E                <1>       jmp    short loc_cfs_write_FSINFO_sector
1325                              <1>
1326                              <1> loc_cfs_FAT32_get_rcalc_parms:
1327 0000D52E 8B15[48940100]      <1>       mov    edx, [CFS_FAT32FC]
1328 0000D534 A1[B7910100]        <1>       mov    eax, [FreeClusterCount]
1329 0000D539 89563E              <1>       mov    [esi+LD_BPB+BPB_Reserved+4], edx ; First Free Cluster
1330                              <1> loc_cfs_write_FSINFO_sector:
1331 0000D53C 89463A              <1>       mov    [esi+LD_BPB+BPB_Reserved], eax ; Free cluster count
1332                              <1>       ; 01/03/2016
1333 0000D53F E8AA000000          <1>       call   set_fat32_fsinfo_sector_parms
1334 0000D544 72AE                <1>         jc      short loc_put_fcc_invalid_sign
1335                              <1>
1336                              <1> loc_set_FAT32_free_sectors:
1337                              <1>       ; 29/02/2016
1338                              <1>       ;mov   eax, [FreeClusterCount]
1339                              <1>       ;mov   ecx, eax
1340                              <1>       ;cmp   eax, 0FFFFFFFFh ; Invalid !
1341                              <1>       ;je    short loc_cfs_retn_params
1342                              <1>       ;
1343 0000D546 8B0D[B7910100]      <1>       mov    ecx, [FreeClusterCount]
1344 0000D54C 0FB64613            <1>       movzx  eax, byte [esi+LD_BPB+SecPerClust]
1345 0000D550 F7E1                <1>       mul    ecx
1346                              <1>       ; 29/02/2016
1347 0000D552 31C9                <1>       xor    ecx, ecx ; 0
1348 0000D554 09D2                <1>       or     edx, edx ; 0 ?
1349 0000D556 759C                <1>         jnz     loc_put_fcc_invalid_sign
1350 0000D558 394670              <1>       cmp    [esi+LD_TotalSectors], eax ; Volume size in sectors
1351 0000D55B 7697                <1>         jna     short loc_put_fcc_invalid_sign
1352                              <1>       ;
1353                              <1> loc_set_FAT32_free_sectors_ok:
1354 0000D55D 31D2                <1>       xor    edx, edx ; 0
1355 0000D55F EB98                <1>         jmp     short loc_cfs_retn_params
1356                              <1>       ;
1357                              <1>
1358                              <1> get_last_cluster:
1359                              <1>       ; 22/10/2016
1360                              <1>       ; 27/02/2016 (TRDOS 386 =  TRDOS v2.0)
1361                              <1>       ; 12/06/2010 (DRV_FAT.ASM, 'proc_get_last_custer')
1362                              <1>       ; 06/06/2010
1363                              <1>       ; INPUT ->
1364                              <1>       ;      EAX = First Cluster Number
1365                              <1>       ;      ESI = Logical Dos Drive Parameters Table
1366                              <1>       ; OUTPUT ->
1367                              <1>       ;      cf = 0 -> No Error, EAX is valid
1368                              <1>       ;      cf = 1 -> EAX > 0 -> Error
1369                              <1>       ;      EAX = Last Cluster Number
1370                              <1>       ;        ECX = Previous Cluster -just before the last cluster-
1371                              <1>       ;        ; 22/10/2016
1372                              <1>       ;      [glc_index] = cluster index number of the last cluster
1373                              <1>       ;
1374                              <1>       ; (Modified registers: EAX, ECX, EBX, EDX)
1375                              <1>
1376 0000D561 89C1                <1>       mov    ecx, eax
1377                              <1>
1378 0000D563 C705[50940100]FFFF- <1>       mov    dword [glc_index], 0FFFFFFFFh ; 22/10/2016
1378 0000D56B FFFF                <1>
1379                              <1>
1380                              <1> loc_glc_get_next_cluster_1:
1381 0000D56D 890D[4C940100]      <1>       mov    [glc_prevcluster], ecx
1382                              <1>       ; 22/10/2016
1383 0000D573 FF05[50940100]      <1>       inc    dword [glc_index]
1384                              <1>
1385                              <1> loc_glc_get_next_cluster_2:
1386 0000D579 E8E8F7FFFF          <1>       call   get_next_cluster
1387                              <1>       ; ecx = current/previous cluster
1388                              <1>       ; eax = next/last cluster
1389 0000D57E 73ED                <1>       jnc    short loc_glc_get_next_cluster_1
1390                              <1>
1391 0000D580 09C0                <1>       or     eax, eax
1392 0000D582 7509                <1>       jnz    short loc_glc_stc_retn
1393                              <1>
1394                              <1>       ; ecx = previous cluster
1395 0000D584 89C8                <1>         mov  eax, ecx
1396                              <1>
1397                              <1>       ; previous cluster becomes last cluster (ecx -> eax)
1398                              <1>       ; previous of previous cluster becomes previous cluster (ecx)
1399                              <1>
1400                              <1> loc_glc_prev_cluster_retn:
1401 0000D586 8B0D[4C940100]      <1>       mov    ecx, [glc_prevcluster]
1402 0000D58C C3                  <1>       retn
1403                              <1>
1404                              <1> loc_glc_stc_retn:
1405 0000D58D F5                  <1>       cmc    ;stc
1406 0000D58E EBF6                <1>         jmp  short loc_glc_prev_cluster_retn
1407                              <1>
1408                              <1> truncate_cluster_chain:
1409                              <1>       ; 01/03/2016
1410                              <1>       ; 28/02/2016 (TRDOS 386 =  TRDOS v2.0)
1411                              <1>       ; 22/01/2011 (DRV_FAT.ASM, 'proc_truncate_cluster_chain')
1412                              <1>       ; 11/09/2010
1413                              <1>       ; INPUT ->
1414                              <1>       ;      ESI = Logical dos drive description table address
1415                              <1>       ;      EAX = First cluster to be truncated/unlinked
```

```
1416                            <1>       ; OUTPUT ->
1417                            <1>       ;      ESI = Logical dos drive description table address
1418                            <1>       ;      ECX = Count of truncated/removed clusters
1419                            <1>       ;      CF = 0 -> EAX = Free sectors
1420                            <1>       ;      CF = 1 -> Error code in EAX (AL)
1421                            <1>
1422                            <1>       ; NOTE: This procedure does not update lm date&time !
1423                            <1>
1424                            <1> loc_truncate_cc:
1425 0000D590 31C9             <1>       xor    ecx, ecx ; mov ecx, 0
1426                            <1>       ;mov   byte [FAT_BuffValidData], 0
1427 0000D592 890D[A2910100]   <1>       mov    [FAT_ClusterCounter], ecx ; 0 ; reset
1428                            <1>
1429                            <1> loc_tcc_unlink_clusters:
1430 0000D598 E8F3FAFFFF       <1>       call   update_cluster
1431                            <1>       ; EAX = Next Cluster
1432                            <1>       ; ECX = Cluster Value
1433                            <1>       ; Note:
1434                            <1>       ; Returns count of unlinked clusters in
1435                            <1>       ; dword ptr FAT_ClusterCounter
1436 0000D59D 73F9               <1>       jnc short loc_tcc_unlink_clusters
1437                            <1>
1438                            <1> pass_tcc_unlink_clusters:
1439 0000D59F A2[57940100]     <1>       mov    byte [TCC_FATErr], al
1440 0000D5A4 803D[9A910100]02 <1>       cmp    byte [FAT_BuffValidData], 2
1441 0000D5AB 750E             <1>       jne    short loc_tcc_calculate_FAT_freespace
1442 0000D5AD E89BFDFFFF       <1>       call   save_fat_buffer
1443 0000D5B2 7307             <1>       jnc    short loc_tcc_calculate_FAT_freespace
1444 0000D5B4 A2[57940100]     <1>       mov    byte [TCC_FATErr], al ; Error
1445                            <1>       ;mov   byte [FAT_BuffValidData], 0
1446                            <1>
1447                            <1>       ; 01/03/2016
1448 0000D5B9 EB12             <1>       jmp    short loc_tcc_recalculate_FAT_freespace
1449                            <1>
1450                            <1> loc_tcc_calculate_FAT_freespace:
1451 0000D5BB A1[A2910100]     <1>       mov    eax, [FAT_ClusterCounter] ; signed (+-) number
1452 0000D5C0 66BB01FF         <1>       mov    bx, 0FF01h ; BH = FFh -> ESI = Dos drv desc. table
1453                            <1>                         ; BL = 1 -> add cluster
1454 0000D5C4 E819FEFFFF       <1>       call   calculate_fat_freespace
1455 0000D5C9 21C9             <1>       and    ecx, ecx ; cx = 0 -> valid free sector count
1456 0000D5CB 7409             <1>       jz     short pass_truncate_cc_recalc_FAT_freespace
1457                            <1>
1458                            <1> loc_tcc_recalculate_FAT_freespace:
1459 0000D5CD 66BB00FF         <1>       mov    bx, 0FF00h ; recalculate !
1460 0000D5D1 E80CFEFFFF       <1>       call   calculate_fat_freespace
1461                            <1>
1462                            <1> loc_tcc_calculate_FAT_freespace_err:
1463                            <1> pass_truncate_cc_recalc_FAT_freespace:
1464 0000D5D6 8B0D[A2910100]   <1>       mov    ecx, [FAT_ClusterCounter]
1465                            <1>
1466 0000D5DC 803D[57940100]00 <1>       cmp    byte [TCC_FATErr], 0
1467 0000D5E3 7608             <1>       jna    short loc_tcc_unlink_clusters_retn
1468                            <1>
1469                            <1> loc_tcc_unlink_clusters_error:
1470 0000D5E5 0FB605[57940100] <1>       movzx  eax, byte [TCC_FATErr]
1471 0000D5EC F9               <1>       stc
1472                            <1> loc_tcc_unlink_clusters_retn:
1473 0000D5ED C3               <1>       retn
1474                            <1>
1475                            <1> set_fat32_fsinfo_sector_parms:
1476                            <1>       ; 15/10/2016
1477                            <1>       ; 23/03/2016
1478                            <1>       ; 29/02/2016 (TRDOS 386 =  TRDOS v2.0)
1479                            <1>       ; INPUT ->
1480                            <1>       ;      ESI = Logical dos drive description table address
1481                            <1>       ;      [esi+LD_BPB+BPB_Reserved] = Free Cluster Count
1482                            <1>       ;      [esi+LD_BPB+BPB_Reserved+4] = First Free Cluster
1483                            <1>       ; OUTPUT ->
1484                            <1>       ;      ESI = Logical dos drive description table address
1485                            <1>       ;      CF = 0 -> OK..
1486                            <1>       ;      CF = 1 -> Error code in EAX (AL)
1487                            <1>       ;
1488                            <1>       ; (Modified registers: EAX, EBX, ECX, EDX)
1489                            <1>
1490 0000D5EE E824000000       <1>       call   get_fat32_fsinfo_sector_parms
1491 0000D5F3 7221             <1>       jc     short update_fat32_fsinfo_sector_retn
1492                            <1>
1493 0000D5F5 8B463A           <1>       mov    eax, [esi+LD_BPB+BPB_Reserved] ; Free Cluster Count
1494 0000D5F8 8B563E           <1>       mov    edx, [esi+LD_BPB+BPB_Reserved+4] ; First free Cluster
1495                            <1>
1496                            <1>         ;mov ebx, DOSBootSectorBuff
1497 0000D5FB 8983E8010000     <1>       mov    [ebx+488], eax
1498 0000D601 8993EC010000     <1>       mov    [ebx+492], edx
1499                            <1>
1500 0000D607 A1[44940100]     <1>       mov    eax, [CFS_FAT32FSINFOSEC]
1501 0000D60C B901000000       <1>       mov    ecx, 1
1502 0000D611 E826550000       <1>       call   disk_write
1503                            <1>       ;jnc     short update_fat32_fsinfo_sector_retn
1504                            <1>
1505                            <1>       ; 15/10/2016 (1Dh -> 18)
1506                            <1>       ; 23/03/2016 (1Dh)
1507                            <1>       ;mov   eax, 18 ; Drive not ready or write error
1508                            <1>
1509                            <1> update_fat32_fsinfo_sector_retn:
1510 0000D616 C3               <1>       retn
1511                            <1>
1512                            <1> get_fat32_fsinfo_sector_parms:
1513                            <1>       ; 15/10/2016
1514                            <1>       ; 23/03/2016
1515                            <1>       ; 01/03/2016
1516                            <1>       ; 29/02/2016 (TRDOS 386 =  TRDOS v2.0)
1517                            <1>       ; INPUT ->
1518                            <1>       ;      ESI = Logical dos drive description table address
1519                            <1>       ; OUTPUT ->
1520                            <1>       ;      ESI = Logical dos drive description table address
```

```
1521                                    <1>      ;      EBX = FSINFO sector buffer address (DOSBootSectorBuff)
1522                                    <1>      ;      CF = 0 -> OK..
1523                                    <1>      ;         EAX = FsInfo sector address
1524                                    <1>      ;         ECX = Free cluster count
1525                                    <1>      ;         EDX = First free cluster
1526                                    <1>      ;      CF = 1 -> Error code in AL (EAX)
1527                                    <1>      ;         EBX = 0
1528                                    <1>      ;
1529                                    <1>      ;      [CFS_FAT32FSINFOSEC] = FAT32 FSINFO sector address
1530                                    <1>        ;
1531                                    <1>      ; (Modified registers: EAX, EBX, ECX, EDX)
1532                                    <1>
1533 0000D617 0FB74636                  <1>      movzx  eax, word [esi+LD_BPB+FAT32_FSInfoSec]
1534 0000D61B 03466C                    <1>      add    eax, [esi+LD_StartSector]
1535 0000D61E A3[44940100]              <1>      mov    [CFS_FAT32FSINFOSEC], eax
1536                                    <1>
1537 0000D623 BB[968F0100]              <1>        mov     ebx, DOSBootSectorBuff
1538 0000D628 B901000000                <1>      mov    ecx, 1
1539 0000D62D E819550000                <1>      call   disk_read
1540 0000D632 7232                      <1>      jc     short loc_read_FAT32_fsinfo_sec_err
1541                                    <1>
1542 0000D634 BB[968F0100]              <1>      mov    ebx, DOSBootSectorBuff
1543                                    <1>
1544 0000D639 813B52526141              <1>      cmp    dword [ebx], 41615252h
1545 0000D63F 751E                      <1>      jne    short loc_read_FAT32_fsinfo_sec_stc
1546                                    <1>
1547 0000D641 81BBE4010000727241-       <1>      cmp    dword [ebx+484], 61417272h
1547 0000D64A 61                        <1>
1548 0000D64B 7512                      <1>      jne    short loc_read_FAT32_fsinfo_sec_stc
1549                                    <1>
1550 0000D64D A1[44940100]              <1>      mov    eax, [CFS_FAT32FSINFOSEC]
1551 0000D652 8B8BE8010000              <1>      mov    ecx, [ebx+488] ; free cluster count
1552 0000D658 8B93EC010000              <1>      mov    edx, [ebx+492] ; first (next) free cluster
1553                                    <1>
1554 0000D65E C3                        <1>      retn
1555                                    <1>
1556                                    <1> loc_read_FAT32_fsinfo_sec_stc:
1557                                    <1>      ; 15/10/2016 (0Bh -> 28)
1558 0000D65F B81C000000                <1>      mov    eax, 28 ; Invalid format!
1559 0000D664 EB05                      <1>      jmp    short loc_read_FAT32_fsinfo_sec_stc_retn
1560                                    <1>
1561                                    <1> loc_read_FAT32_fsinfo_sec_err:
1562                                    <1>      ; 15/10/2016 (15h -> 17)
1563                                    <1>      ; 23/03/2016 (15h)
1564 0000D666 B811000000                <1>      mov    eax, 17 ; Drive not ready or read error
1565                                    <1>
1566                                    <1> loc_read_FAT32_fsinfo_sec_stc_retn:
1567 0000D66B 29DB                      <1>      sub    ebx, ebx ; 0
1568 0000D66D F9                        <1>      stc
1569 0000D66E C3                        <1>      retn
1570                                    <1>
1571                                    <1> add_new_cluster:
1572                                    <1>      ; 15/10/2016
1573                                    <1>      ; 16/05/2016
1574                                    <1>      ; 18/03/2016, 24/03/2016
1575                                    <1>      ; 11/03/2016 (TRDOS 386 =  TRDOS v2.0)
1576                                    <1>      ; 30/07/2011 (DRV_FAT.ASM)
1577                                    <1>      ; 11/09/2010
1578                                    <1>      ; INPUT ->
1579                                    <1>      ;      ESI = Logical dos drv desc. table address
1580                                    <1>      ;      EAX = Last cluster
1581                                    <1>      ; OUTPUT ->
1582                                    <1>      ;      ESI = Logical dos drv desc. table address
1583                                    <1>      ;      EAX = New Last cluster (next cluster)
1584                                    <1>      ;      cf = 1 -> error code in EAX (AL)
1585                                    <1>      ;      cf = 1 -> DX = sectors per cluster
1586                                    <1>      ;      ECX = Free sectors
1587                                    <1>      ; NOTE:
1588                                    <1>      ; This procedure does not update lm date&time !
1589                                    <1>      ;
1590                                    <1>      ; (Modified registers: EAX, EBX, ECX, EDX, EDI)
1591                                    <1>      ;
1592                                    <1>
1593 0000D66F A3[74950100]              <1>      mov    [FAT_anc_LCluster], eax
1594                                    <1>
1595 0000D674 E844F9FFFF                <1>      call   get_first_free_cluster
1596 0000D679 720B                      <1>      jc     short loc_add_new_cluster_retn
1597                                    <1>      ; EAX >= 2 and EAX < FFFFFFFFh is valid
1598                                    <1>
1599 0000D67B 89C2                      <1>      mov    edx, eax
1600                                    <1>
1601 0000D67D 42                        <1>      inc    edx
1602                                    <1>      ;jnz   short loc_add_new_cluster_check_ffc_eax
1603 0000D67E 7516                      <1>      jnz    short loc_add_new_cluster_save_fcc
1604                                    <1>
1605                                    <1> loc_add_new_cluster_no_disk_space_retn:
1606 0000D680 B827000000                <1>      mov    eax, 27h ; MSDOS err => insufficient disk space
1607                                    <1> loc_add_new_cluster_stc_retn:
1608 0000D685 F9                        <1>      stc
1609                                    <1> loc_add_new_cluster_retn:
1610 0000D686 0FB65E13                  <1>      movzx  ebx, byte [esi+LD_BPB+SecPerClust]
1611 0000D68A 8B4E74                    <1>      mov    ecx, [esi+LD_FreeSectors]
1612                                    <1>      ;xor   edx, edx
1613                                    <1>      ;stc
1614 0000D68D C3                        <1>      retn
1615                                    <1>
1616                                    <1> loc_anc_invalid_format_stc_retn:
1617 0000D68E F9                        <1>      stc
1618                                    <1> loc_add_new_cluster_invalid_format_retn:
1619                                    <1>      ; 15/10/2016 (0Bh -> 28)
1620 0000D68F B81C000000                <1>      mov    eax, 28 ; Invalid format
1621 0000D694 EBF0                      <1>      jmp    short loc_add_new_cluster_retn
1622                                    <1>
1623                                    <1> ;loc_add_new_cluster_check_ffc_eax:
1624                                    <1> ;     cmp    eax, 2
```

```
1625                            <1> ;       jb      short loc_add_new_cluster_invalid_format_retn
1626                            <1>
1627                            <1> loc_add_new_cluster_save_fcc:
1628 0000D696 A3[78950100]     <1>         mov     [FAT_anc_FFCluster], eax
1629                            <1>
1630 0000D69B 83E802           <1>         sub     eax, 2
1631 0000D69E 0FB65E13         <1>           movzx   ebx, byte [esi+LD_BPB+SecPerClust]
1632 0000D6A2 F7E3             <1>         mul     ebx
1633 0000D6A4 09D2             <1>         or      edx, edx
1634 0000D6A6 75E6             <1>         jnz     short loc_anc_invalid_format_stc_retn
1635                            <1>
1636                            <1> loc_add_new_cluster_allocate_cluster:
1637                            <1>         ; 18/03/2016
1638 0000D6A8 92               <1>         xchg    edx, eax ; eax = 0
1639                            <1>         ; 16/05/2016
1640                            <1>         ;cmp    [ClusterBuffer_Valid], al ; 0
1641                            <1>         ;jna    short loc_anc_clear_cluster_buffer
1642                            <1>         ;; 'copy' command,
1643                            <1>         ;; writing destination file clust after reading source file clust
1644                            <1>         ;mov    [ClusterBuffer_Valid], al ; 0 ; reset
1645                            <1>         ;jmp    short loc_add_new_cluster_write_nc_to_disk
1646                            <1>
1647                            <1> loc_anc_clear_cluster_buffer:
1648                            <1>         ; 11/03/2016
1649                            <1>         ; Clear buffer
1650 0000D6A9 BF00000700         <1>         mov     edi, Cluster_Buffer ; 70000h (for current TRDOS 386 version)
1651 0000D6AE 89D9             <1>         mov     ecx, ebx ; sector count
1652 0000D6B0 C1E107           <1>         shl     ecx, 7 ; 1 sector = 512 bytes -> 128 double words
1653                            <1>         ;xor    eax, eax ; 0
1654 0000D6B3 F3AB             <1>         rep     stosd
1655                            <1>
1656                            <1> loc_add_new_cluster_write_nc_to_disk:
1657                            <1>         ; 11/03/2016
1658                            <1>         ;xchg  eax, edx ; edx = 0, eax = sector offset
1659 0000D6B5 89D0             <1>         mov     eax, edx
1660 0000D6B7 034668           <1>           add     eax, [esi+LD_DATABegin]
1661 0000D6BA 72D3             <1>         jc      short loc_add_new_cluster_invalid_format_retn
1662                            <1>
1663 0000D6BC 89D9             <1>         mov     ecx, ebx ; ECX = sectors per cluster (<256)
1664 0000D6BE BB00000700       <1>         mov     ebx, Cluster_Buffer
1665 0000D6C3 E874540000       <1>         call    disk_write
1666 0000D6C8 7307             <1>         jnc     short loc_add_new_cluster_update_fat_nlc
1667                            <1>
1668                            <1>         ; 15/10/2016 (1Dh -> 18)
1669 0000D6CA B812000000       <1>         mov     eax, 18 ; Write Error
1670 0000D6CF EBB4             <1>         jmp     short loc_add_new_cluster_stc_retn
1671                            <1>
1672                            <1> loc_add_new_cluster_update_fat_nlc:
1673 0000D6D1 A1[78950100]     <1>         mov     eax, [FAT_anc_FFCluster]
1674 0000D6D6 31C9             <1>         xor     ecx, ecx
1675 0000D6D8 890D[A2910100]   <1>         mov     [FAT_ClusterCounter], ecx ; 0 ; reset
1676 0000D6DE 49               <1>         dec     ecx ; 0FFFFFFFFh
1677 0000D6DF E8ACF9FFFF       <1>         call    update_cluster
1678 0000D6E4 7304             <1>         jnc     short loc_add_new_cluster_update_fat_plc
1679 0000D6E6 09C0             <1>         or      eax, eax ;EAX = 0 -> cluster value is 0 or eocc
1680 0000D6E8 759B             <1>         jnz     short loc_add_new_cluster_stc_retn
1681                            <1>
1682                            <1> loc_add_new_cluster_update_fat_plc:
1683 0000D6EA A1[74950100]     <1>         mov     eax, [FAT_anc_LCluster]
1684 0000D6EF 8B0D[78950100]   <1>         mov     ecx, [FAT_anc_FFCluster]
1685 0000D6F5 E896F9FFFF       <1>         call    update_cluster
1686 0000D6FA 7314             <1>         jnc     short loc_add_new_cluster_save_fat_buffer
1687 0000D6FC 09C0             <1>         or      eax, eax ; EAX = 0 -> cluster value is 0 or eocc
1688 0000D6FE 7410             <1>         jz      short loc_add_new_cluster_save_fat_buffer
1689                            <1>
1690                            <1> loc_anc_save_fat_buffer_err_retn:
1691                            <1>         ;cmp   byte [FAT_ClusterCounter], 1
1692                            <1>         ;jb    short loc_add_new_cluster_retn
1693                            <1>
1694 0000D700 66BB00FF         <1>         mov     bx, 0FF00h ; recalculate free space (BL = 0)
1695                            <1>                            ; (BH = FFh -> Use ESI as Drv Param. Tbl.)
1696 0000D704 50               <1>         push    eax
1697 0000D705 E8D8FCFFFF       <1>         call    calculate_fat_freespace
1698 0000D70A 58               <1>         pop     eax
1699 0000D70B E975FFFFFF       <1>           jmp     loc_add_new_cluster_stc_retn
1700                            <1>
1701                            <1> loc_add_new_cluster_save_fat_buffer:
1702                            <1>         ;cmp   byte [FAT_BuffValidData], 2
1703                            <1>         ;jne   short loc_add_new_cluster_calc_FAT_freespace
1704                            <1>         ;Byte [FAT_BuffValidData] =  2
1705 0000D710 E838FCFFFF         <1>         call    save_fat_buffer
1706 0000D715 72E9             <1>         jc      short loc_anc_save_fat_buffer_err_retn
1707                            <1>
1708                            <1> loc_add_new_cluster_calc_FAT_freespace:
1709                            <1>         ;mov   eax, 1 ; Only one Cluster
1710 0000D717 A1[A2910100]     <1>         mov     eax, [FAT_ClusterCounter]
1711 0000D71C 66BB01FF         <1>         mov     bx, 0FF01h ; BH = FFh -> ESI -> Dos drv desc. table
1712                            <1>                        ; BL = 1 -> add cluster
1713 0000D720 B301             <1>         mov     bl, 01h ; BL = 1 -> add clusters
1714                            <1>         ; NOTE: EAX value will be added to Free Cluster Count
1715                            <1>         ; (Free Cluster Count is decreased when EAX value is negative)
1716 0000D722 E8BBFCFFFF       <1>           call    calculate_fat_freespace
1717                            <1>         ;ECX = 0 -> no error, ECX > 0 -> error or invalid return
1718 0000D727 21C9             <1>         and     ecx, ecx ; ECX = 0 -> valid free sector count
1719 0000D729 7409             <1>         jz      short loc_add_new_cluster_return_cluster_number
1720                            <1>
1721                            <1> loc_add_new_cluster_recalc_FAT_freespace:
1722 0000D72B 66BB00FF         <1>         mov     bx, 0FF00h  ; recalculate free space
1723 0000D72F E8AEFCFFFF       <1>           call    calculate_fat_freespace
1724                            <1>         ; cf = 0
1725                            <1> loc_add_new_cluster_return_cluster_number:
1726 0000D734 89C1             <1>         mov     ecx, eax ; Free sector count
1727 0000D736 A1[78950100]     <1>         mov     eax, [FAT_anc_FFCluster]
1728 0000D73B 0FB65E13         <1>         movzx   ebx, byte [esi+LD_BPB+SecPerClust]
1729                            <1>         ;mov   edi, Cluster_Buffer
```

```
1730 0000D73F 31D2              <1>       xor    edx, edx
1731 0000D741 C3                <1>         retn
1732                            <1>
1733                            <1> write_cluster:
1734                            <1>       ; 15/10/2016
1735                            <1>       ; 21/03/2016 (TRDOS 386 =  TRDOS v2.0)
1736                            <1>       ;
1737                            <1>       ; INPUT ->
1738                            <1>       ;      EAX = Cluster Number (Sector index for SINGLIX FS)
1739                            <1>       ;      ESI = Logical DOS Drive Description Table address
1740                            <1>       ;      EBX = Cluster (File R/W) Buffer address (max. 64KB)
1741                            <1>       ;      Only for SINGLIX FS:
1742                            <1>       ;      EDX = File Number (The 1st FDT address)
1743                            <1>       ; OUTPUT ->
1744                            <1>       ;      cf = 1 -> Cluster can not be written onto disk
1745                            <1>       ;          EAX > 0 -> Error number
1746                            <1>       ;      cf = 0 -> Cluster has been written successfully
1747                            <1>       ;
1748                            <1>       ; (Modified registers: EAX, ECX, EBX, EDX)
1749                            <1>
1750 0000D742 0FB64E13           <1>       movzx  ecx, byte [esi+LD_BPB+BPB_SecPerClust]
1751                            <1>       ; CL = 1 = [esi+LD_FS_Reserved2] ; SectPerClust for Singlix FS
1752                            <1>
1753                            <1> write_file_sectors: ; 16/03/2016
1754 0000D746 807E0300          <1>       cmp    byte [esi+LD_FATType], 0
1755 0000D74A 761C              <1>       jna    short write_fs_cluster
1756                            <1>
1757                            <1> write_fat_file_sectors:
1758 0000D74C 83E802            <1>       sub    eax, 2 ; Beginning cluster number is always 2
1759 0000D74F 0FB65613          <1>       movzx  edx, byte [esi+LD_BPB+BPB_SecPerClust] ; 18/03/2016
1760 0000D753 F7E2              <1>       mul    edx
1761 0000D755 034668            <1>       add    eax, [esi+LD_DATABegin] ; absolute address of the cluster
1762                            <1>
1763                            <1>       ; EAX = Disk sector address
1764                            <1>       ; ECX = Sector count
1765                            <1>       ; EBX = Buffer address
1766                            <1>       ; (EDX = 0)
1767                            <1>       ; ESI = Logical DOS drive description table address
1768                            <1>
1769 0000D758 E8DF530000        <1>       call   disk_write
1770 0000D75D 7306              <1>       jnc    short wclust_retn
1771                            <1>
1772                            <1>       ; 15/10/2016 (1Dh -> 18)
1773 0000D75F B812000000        <1>       mov    eax, 18 ; Drive not ready or write error !
1774 0000D764 C3                <1>       retn
1775                            <1>
1776                            <1> wclust_retn:
1777 0000D765 29C0              <1>       sub    eax, eax ; 0
1778 0000D767 C3                <1>       retn
1779                            <1>
1780                            <1> write_fs_cluster:
1781                            <1>       ; 21/03/2016 (TRDOS 386 =  TRDOS v2.0)
1782                            <1>       ; Singlix FS
1783                            <1>
1784                            <1>       ; EAX = Cluster number is sector index number of the file (eax)
1785                            <1>
1786                            <1>       ; EDX = File number is the first File Descriptor Table address
1787                            <1>       ;      of the file. (Absolute address of the FDT).
1788                            <1>
1789                            <1>       ; eax = sector index (0 for the first sector)
1790                            <1>       ; edx = FDT0 address
1791                            <1>             ; 64 KB buffer = 128 sectors (limit)
1792 0000D768 B980000000        <1>       mov    ecx, 128 ; maximum count of sectors (before eof)
1793 0000D76D E801000000        <1>       call   write_fs_sectors
1794 0000D772 C3                <1>       retn
1795                            <1>
1796                            <1> write_fs_sectors:
1797                            <1>       ; 21/03/2016 (TRDOS 386 =  TRDOS v2.0)
1798 0000D773 F9                <1>       stc
1799 0000D774 C3                <1>       retn
1800                            <1>
1801                            <1> get_cluster_by_index:
1802                            <1>       ; 29/04/2016 (TRDOS 386 =  TRDOS v2.0)
1803                            <1>       ; INPUT ->
1804                            <1>       ;      EAX = Beginning cluster
1805                            <1>       ;      EDX = Sector index in disk/file section
1806                            <1>       ;            (Only for SINGLIX file system!)
1807                            <1>       ;      ECX = Cluster sequence number after the beginning cluster
1808                            <1>       ;      ESI = Logical DOS Drive Description Table address
1809                            <1>       ; OUTPUT ->
1810                            <1>       ;      EAX = Cluster number
1811                            <1>       ;      cf = 1 -> Error code in AL (EAX)
1812                            <1>       ;
1813                            <1>       ;(Modified registers: EAX, ECX, EBX, EDX)
1814                            <1>       ;
1815 0000D775 807E0301          <1>       cmp    byte [esi+LD_FATType], 1
1816 0000D779 721E              <1>         jb      short get_fs_section_by_index
1817                            <1>
1818 0000D77B 3B4E78            <1>       cmp    ecx, [esi+LD_Clusters]
1819 0000D77E 7207              <1>       jb     short gcbi_1
1820                            <1> gcbi_0:
1821 0000D780 F9                <1>       stc
1822 0000D781 B823000000        <1>       mov    eax, 23h ; Cluster not available !
1823                            <1>                       ; MSDOS error code: FCB unavailable
1824 0000D786 C3                <1>       retn
1825                            <1> gcbi_1:
1826 0000D787 51                <1>       push   ecx
1827 0000D788 E8D9F5FFFF        <1>       call   get_next_cluster
1828 0000D78D 59                <1>       pop    ecx
1829 0000D78E 7203              <1>       jc     short gcbi_3
1830 0000D790 E2F5              <1>       loop   gcbi_1
1831                            <1> gcbi_2:
1832 0000D792 C3                <1>       retn
1833                            <1> gcbi_3:
1834 0000D793 09C0              <1>       or     eax, eax
```

```
1835 0000D795 74E9              <1>       jz     short gcbi_0
1836 0000D797 F5                <1>       cmc    ; stc
1837 0000D798 C3                <1>       retn
1838                            <1>
1839                            <1> get_fs_section_by_index:
1840                            <1>       ; 29/04/2016 (TRDOS 386 =  TRDOS v2.0)
1841                            <1>       ; INPUT ->
1842                            <1>       ;      EAX = Beginning FDT number/address
1843                            <1>       ;      EDX = Sector index in disk/file section
1844                            <1>       ;      ECX = Sector sequence number after the beginning FDT
1845                            <1>       ;      ESI = Logical DOS Drive Description Table address
1846                            <1>       ; OUTPUT ->
1847                            <1>       ;      EAX = FDT number/address
1848                            <1>       ;      EDX = Sector index of the section (0,1,2,3,4...)
1849                            <1>       ;      cf = 1 -> Error code in AL (EAX)
1850                            <1>       ;
1851                            <1>       ;(Modified registers: EAX, ECX, EBX, EDX)
1852                            <1>       ;
1853 0000D799 B8FFFFFFFF        <1>       mov    eax, 0FFFFFFFFh
1854 0000D79E C3                <1>       retn
1855                            <1>
1856                            <1> get_last_section:
1857                            <1>       ; 22/10/2016 (TRDOS 386 =  TRDOS v2.0)
1858                            <1>       ; INPUT ->
1859                            <1>       ;      EAX = (The 1st) FDT number/address
1860                            <1>       ;      ESI = Logical DOS Drive Description Table address
1861                            <1>       ; OUTPUT ->
1862                            <1>       ;      EAX = FDT number/address of the last section
1863                            <1>       ;      EDX = Last sector of the section (0,1,2,3,4...)
1864                            <1>       ;      [glc_index] = sector index number of the last sector
1865                            <1>       ;                    (for file, not for the last section)
1866                            <1>       ;
1867                            <1>       ;      cf = 1 -> Error code in AL (EAX)
1868                            <1>       ;
1869                            <1>       ;(Modified registers: EAX, ECX, EBX, EDX)
1870                            <1>       ;
1871 0000D79F B800000000        <1>       mov    eax, 0
1872 0000D7A4 BA00000000        <1>       mov    edx, 0
1873 0000D7A9 C3                <1>       retn
3084                                %include 'trdosk6.s' ; 24/01/2016
   1                            <1> ; ****************************************************************************
   2                            <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.3) - MAIN PROGRAM : trdosk6.s
   3                            <1> ; ----------------------------------------------------------------------------
   4                            <1> ; Last Update: 15/02/2021
   5                            <1> ; ----------------------------------------------------------------------------
   6                            <1> ; Beginning: 24/01/2016
   7                            <1> ; ----------------------------------------------------------------------------
   8                            <1> ; Assembler: NASM version 2.15 t(trdos386.s)
   9                            <1> ; ----------------------------------------------------------------------------
  10                            <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
  11                            <1> ; u1.s (27/17/2015), u2.s (03/01/2016)
  12                            <1> ; ****************************************************************************
  13                            <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
  14                            <1> ; TRDOS2.ASM (09/11/2011)
  15                            <1> ; ----------------------------------------------------------------------------
  16                            <1> ; INT_21H.ASM (c) 2009-2011 Erdogan TAN  [14/11/2009] Last Update: 08/11/2011
  17                            <1>
  18                            <1> sysent: ; < enter to system call >
  19                            <1>       ; 17/03/2017
  20                            <1>       ; 03/03/2017
  21                            <1>       ; 19/02/2017
  22                            <1>       ; 13/01/2017
  23                            <1>       ; 06/06/2016
  24                            <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
  25                            <1>       ; 16/04/2015 - 19/10/2015 (Retro UNIX 386 v1)
  26                            <1>       ; 10/04/2013 - 18/01/2014 (Retro UNIX 8086 v1)
  27                            <1>       ;
  28                            <1>       ; 'unkni' or 'sysent' is sytem entry from various traps.
  29                            <1>       ; The trap type is determined and an indirect jump is made to
  30                            <1>       ; the appropriate system call handler. If there is a trap inside
  31                            <1>       ; the system a jump to panic is made. All user registers are saved
  32                            <1>       ; and u.sp points to the end of the users stack. The sys (trap)
  33                            <1>       ; instructor is decoded to get the the system code part (see
  34                            <1>       ; trap instruction in the PDP-11 handbook) and from this
  35                            <1>       ; the indirect jump address is calculated. If a bad system call is
  36                            <1>       ; made, i.e., the limits of the jump table are exceeded, 'badsys'
  37                            <1>       ; is called. If the call is legitimate control passes to the
  38                            <1>       ; appropriate system routine.
  39                            <1>       ;
  40                            <1>       ; Calling sequence:
  41                            <1>       ;     Through a trap caused by any sys call outside the system.
  42                            <1>       ; Arguments:
  43                            <1>       ;     Arguments of particular system call.
  44                            <1>       ; ....................................................
  45                            <1>       ;
  46                            <1>       ; Retro UNIX 8086 v1 modification:
  47                            <1>       ;       System call number is in EAX register.
  48                            <1>       ;
  49                            <1>       ;       Other parameters are in EDX, EBX, ECX, ESI, EDI, EBP
  50                            <1>       ;     registers depending of function details.
  51                            <1>       ;
  52                            <1>       ; 16/04/2015
  53 0000D7AA 368925[5C030300]  <1>         mov     [ss:u.sp], esp ; Kernel stack points to return address
  54                            <1>
  55                            <1>       ; save user registers
  56 0000D7B1 1E                <1>       push   ds
  57 0000D7B2 06                <1>       push   es
  58 0000D7B3 0FA0              <1>       push   fs
  59 0000D7B5 0FA8              <1>       push   gs
  60 0000D7B7 60                <1>       pushad  ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi
  61                            <1>       ;
  62                            <1>       ; ESPACE = [ss:u.sp] - esp ; 4*12 = 48 ; 17/09/2015 ; 06/06/2016
  63                            <1>       ;     (ESPACE is size of space in kernel stack
  64                            <1>       ;     for saving/restoring user registers.)
  65                            <1>       ;
```

```
0000D7B8 50                  <1>       push   eax ; 01/07/2015
0000D7B9 66B81000            <1>       mov     ax, KDATA
0000D7BD 8ED8                <1>         mov     ds, ax
0000D7BF 8EC0                <1>         mov     es, ax
0000D7C1 8EE0                <1>         mov     fs, ax
0000D7C3 8EE8                <1>         mov     gs, ax
0000D7C5 A1[C0890100]        <1>       mov    eax, [k_page_dir]
0000D7CA 0F22D8              <1>       mov    cr3, eax
0000D7CD 58                  <1>       pop    eax ; 01/07/2015
                             <1>       ; 19/10/2015
0000D7CE FC                  <1>       cld
                             <1>       ;
0000D7CF FE05[5B030300]      <1>       inc    byte [sysflg]
                             <1>              ; incb sysflg / indicate a system routine is in progress
0000D7D5 FB                  <1>         sti  ; 18/01/2014
0000D7D6 0F85DF9DFFFF        <1>       jnz     panic ; 24/05/2013
                             <1>              ; beq 1f
                             <1>              ; jmp panic ; / called if trap inside system
                             <1> ;1:
                             <1>       ; 17/03/2017
0000D7DC 80642438FE          <1>       and    byte [esp+ESPACE+8], ~1 ; clear carry flag
                             <1>
                             <1>       ; 16/04/2015
0000D7E1 A3[64030300]        <1>       mov    [u.r0], eax
0000D7E6 8925[60030300]      <1>       mov    [u.usp], esp ; kernel stack points to user's registers
                             <1>
                             <1>       ; 13/01/2017 (TRDOS 386 Feaure only !)
0000D7EC 803D[D4030300]00    <1>       cmp    byte [u.t_lock], 0 ; timer interrupt lock ?
0000D7F3 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
0000D7F9 C1E002              <1>       shl    eax, 2
                             <1>             ; asl r0 / multiply by 2 to jump indirect in bytes
0000D7FC 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
0000D801 F5                  <1>       cmc
0000D802 9C                  <1>       pushf
0000D803 50                  <1>       push   eax
0000D804 8B2D[5C030300]      <1>       mov    ebp, [u.sp] ; Kernel stack at the beginning of sys call
0000D80A B0FE                <1>       mov    al, 0FEh ; 11111110b
0000D80C 1400                <1>       adc    al, 0 ; al = al + cf
0000D80E 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
0000D811 5D                  <1>       pop    ebp ; eax
0000D812 9D                  <1>       popf
0000D813 0F8208020000        <1>         jc      badsys
0000D819 A1[64030300]        <1>       mov    eax, [u.r0]
                             <1>       ; system call registers: EAX, EDX, ECX, EBX, ESI, EDI
0000D81E FFA5[24D80000]      <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)
0000D824 [361A0100]          <1>       dd sysver    ; 0 ; Get TRDOS 386 version number (v2.0)
0000D828 [83DA0000]          <1>       dd sysexit   ; 1
0000D82C [58DC0000]          <1>       dd sysfork   ; 2
0000D830 [8BE00000]          <1>       dd sysread   ; 3
0000D834 [AAE00000]          <1>       dd syswrite  ; 4
0000D838 [41DE0000]          <1>       dd sysopen   ; 5
0000D83C [62E00000]          <1>       dd sysclose  ; 6
0000D840 [DADB0000]          <1>       dd syswait   ; 7
0000D844 [70DD0000]          <1>       dd syscreat  ; 8
0000D848 [8A280100]          <1>       dd sysrename ; 9  ; TRDOS 386, Rename File (31/12/2017)
0000D84C [05240100]          <1>       dd sysdelete ; 10 ; TRDOS 386, Delete File (29/12/2017)
0000D850 [D30D0100]          <1>       dd sysexec   ; 11
0000D854 [2F250100]          <1>       dd syschdir  ; 12
0000D858 [F4260100]          <1>       dd systime   ; 13 ; TRDOS 386, Get Sys Date&Time (30/12/2017)
0000D85C [24E00000]          <1>       dd sysmkdir  ; 14
0000D860 [63250100]          <1>       dd syschmod  ; 15 ; TRDOS 386, Change Attributes (30/12/2017)
0000D864 [6C240100]          <1>       dd sysrmdir  ; 16 ; TRDOS 386, Remove Directory (29/12/2017)
0000D868 [AE100100]          <1>       dd sysbreak  ; 17
0000D86C [49260100]          <1>       dd sysdrive  ; 18 ; TRDOS 386, Get/Set Current Drv (30/12/2017)
0000D870 [EF100100]          <1>       dd sysseek   ; 19
```

```
171 0000D874 [01110100]          <1>       dd systell   ; 20
172 0000D878 [AB290100]          <1>       dd sysmem    ; 21 ; TRDOS 386, Get Total&Free Mem (31/12/2017)
173 0000D87C [E1290100]          <1>       dd sysprompt ; 22 ; TRDOS 386, Change Cmd Prompt (31/12/2017)
174 0000D880 [232A0100]          <1>       dd syspath   ; 23 ; TRDOS 386, Get/Set Run Path (31/12/2017)
175 0000D884 [902A0100]          <1>       dd sysenv    ; 24 ; TRDOS 386, Get/Set Env Vars (31/12/2017)
176 0000D888 [75270100]          <1>       dd sysstime  ; 25 ; TRDOS 386, Set Sys Date&Time (30/12/2017)
177 0000D88C [67110100]          <1>       dd sysquit   ; 26
178 0000D890 [5B110100]          <1>       dd sysintr   ; 27
179 0000D894 [98260100]          <1>       dd sysdir    ; 28 ; TRDOS 386, Get Curr Drive&Dir (30/12/2017)
180 0000D898 [41E10000]          <1>       dd sysemt    ; 29
181 0000D89C [D3260100]          <1>       dd sysldrvt  ; 30 ; TRDOS 386, Get Logical DOS DDT (30/12/2017)
182 0000D8A0 [F2E20000]          <1>       dd sysvideo  ; 31 ; TRDOS 386 Video Functions (16/05/2016)
183 0000D8A4 [5A340100]          <1>       dd sysaudio  ; 32 ; TRDOS 386 Audio Functions (16/05/2016)
184 0000D8A8 [5AE10000]          <1>       dd systimer  ; 33 ; TRDOS 386 Timer Functions (18/05/2016)
185 0000D8AC [A8110100]          <1>       dd syssleep  ; 34 ; Retro UNIX 8086 v1 feature only !
186                              <1>                            ; 11/06/2014
187 0000D8B0 [D7110100]          <1>       dd sysmsg    ; 35 ; Retro UNIX 386 v1 feature only !
188                              <1>                            ; 01/07/2015
189 0000D8B4 [F4120100]          <1>       dd sysgeterr ; 36 ; Retro UNIX 386 v1 feature only !
190                              <1>                            ; 21/09/2015 - get last error number
191 0000D8B8 [DC230100]          <1>       dd sysfpstat ; 37 ; TRDOS 386 FPU state option (28/02/2017)
192 0000D8BC [451A0100]          <1>       dd syspri    ; 38 ; change priority - TRDOS 386 (20/05/2016)
193 0000D8C0 [96D90000]          <1>       dd sysrele   ; 39 ; TRDOS 386 (19/05/2016) (0 -> 39)
194 0000D8C4 [781B0100]          <1>       dd sysfff    ; 40 ; Find First File - TRDOS 386 (15/10/2016)
195 0000D8C8 [571C0100]          <1>       dd sysfnf    ; 41 ; Find Next File - TRDOS 386 (15/10/2016)
196 0000D8CC [C7220100]          <1>       dd sysalloc  ; 42 ; Allocate contiguous memory block/pages
197                              <1>                            ; TRDOS 386 (19/02/2017) DMA buff fuctions
198 0000D8D0 [85230100]          <1>       dd sysdalloc ; 43 ; Deallocate contiguous memory block/pages
199                              <1>                            ; TRDOS 386 (19/02/2017) DMA buff fuctions
200 0000D8D4 [C0230100]          <1>       dd syscalbac ; 44 ; IRQ Callback and Signal Response Byte
201                              <1>                            ; service setup - TRDOS 386 (20/02/2017)
202                              <1>                            ; 28/08/2017 (20/08/2017)
203 0000D8D8 [EB3C0100]          <1>       dd sysdma    ; 45 ; TRDOS 386 - (ISA) DMA service
204                              <1>
205                              <1> end_of_syscalls:
206                              <1>
207                              <1> error:
208                              <1>       ; 18/05/2016
209                              <1>       ; 13/05/2016
210                              <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
211                              <1>       ; 16/04/2015 - 17/09/2015 (Retro UNIX 386 v1)
212                              <1>       ; 10/04/2013 - 07/08/2013 (Retro UNIX 8086 v1)
213                              <1>       ;
214                              <1>       ; 'error' merely sets the error bit off the processor status (c-bit)
215                              <1>       ; then falls right into the 'sysret', 'sysrele' return sequence.
216                              <1>       ;
217                              <1>       ; INPUTS -> none
218                              <1>       ; OUTPUTS ->
219                              <1>       ;     processor status - carry (c) bit is set (means error)
220                              <1>       ;
221                              <1>       ; 26/05/2013 (Stack pointer must be reset here!
222                              <1>       ;           Because, jumps to error procedure
223                              <1>       ;           disrupts push-pop nesting balance)
224                              <1>       ;
225 0000D8DC 8B2D[5C030300]      <1>       mov    ebp, [u.sp] ; interrupt (system call) return (iretd) address
226 0000D8E2 804D0801            <1>       or     byte [ebp+8], 1  ; set carry bit of flags register
227                              <1>                             ; (system call will return with cf = 1)
228                              <1>             ; bis $1,20.(r1) / set c bit in processor status word below
229                              <1>                              ; / users stack
230                              <1>       ; 17/09/2015
231 0000D8E6 83ED30             <1>       sub    ebp, ESPACE ; 48 ; total size of stack frame ('sysdefs.inc')
232                              <1>                             ; for saving/restoring user registers
233                              <1>       ;cmp   ebp, [u.usp]
234                              <1>       ;je    short err0
235 0000D8E9 892D[60030300]      <1>       mov    [u.usp], ebp
236                              <1> ;err0:
237                              <1>       ; 01/09/2015
238 0000D8EF 8B25[60030300]      <1>       mov    esp, [u.usp]     ; Retro Unix 8086 v1 modification!
239                              <1>                               ; 10/04/2013
240                              <1>                               ; (If an I/O error occurs during disk I/O,
241                              <1>                               ; related procedures will jump to 'error'
242                              <1>                               ; procedure directly without returning to
243                              <1>                               ; the caller procedure. So, stack pointer
244                              <1>                                     ; must be restored here.)
245                              <1>       ; 13/05/2016
246                              <1>       ; NOTE: (The last) error code is in 'u.error', it can be retrieved by
247                              <1>       ;     'get last error' system call later.
248                              <1>
249                              <1>       ; 03/09/2015 - 09/06/2015 - 07/08/2013
250 0000D8F5 C605[C6030300]00    <1>       mov    byte [u.kcall], 0 ; namei_r, mkdir_w reset
251                              <1>
252                              <1> sysret: ; < return from system call>
253                              <1>       ; 01/03/2017
254                              <1>       ; 28/02/2017
255                              <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
256                              <1>       ; 16/04/2015 - 10/09/2015 (Retro UNIX 386 v1)
257                              <1>       ; 10/04/2013 - 23/02/2014 (Retro UNIX 8086 v1)
258                              <1>       ;
259                              <1>       ; 'sysret' first checks to see if process is about to be
260                              <1>       ; terminated (u.bsys). If it is, 'sysexit' is called.
261                              <1>       ; If not, following happens:
262                              <1>       ;     1) The user's stack pointer is restored.
263                              <1>       ;     2) r1=0 and 'iget' is called to see if last mentioned
264                              <1>       ;        i-node has been modified. If it has, it is written out
265                              <1>       ;        via 'ppoke'.
266                              <1>       ;     3) If the super block has been modified, it is written out
267                              <1>       ;        via 'ppoke'.
268                              <1>       ;     4) If the dismountable file system's super block has been
269                              <1>       ;        modified, it is written out to the specified device
270                              <1>       ;        via 'ppoke'.
271                              <1>       ;     5) A check is made if user's time quantum (uquant) ran out
272                              <1>       ;        during his execution. If so, 'tswap' is called to give
273                              <1>       ;        another user a chance to run.
274                              <1>       ;     6) 'sysret' now goes into 'sysrele'.
275                              <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>       ;
0000D8FC 31C0           <1>       xor   eax, eax ; 28/02/2017
                        <1> sysret0: ; 29/07/2015 (eax = 0, jump from sysexec)
0000D8FE FEC0           <1>       inc   al ; 04/05/2013
0000D900 3805[B2030300] <1>       cmp   [u.bsys], al ; 1
                        <1>             ; tstb u.bsys / is a process about to be terminated because
0000D906 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
0000D90C FEC8           <1>       dec   al ; mov ax, 0
                        <1>             ; clr r1 / zero r1 to check last mentioned i-node
0000D90E E828520000     <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:
0000D913 803D[A8030300]00 <1>     cmp   byte [u.quant], 0 ; 16/05/2013
                        <1>             ; tstb uquant / is the time quantum 0?
0000D91A 7705           <1>         ja      short swapret
                        <1>             ; bne 1f / no, don't swap it out
                        <1> sysrelease: ; 07/12/2013 (jump from 'clock')
0000D91C E8DD3F0000     <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
0000D921 E815520000     <1>       call  isintr
                        <1>       ; 20/10/2013
0000D926 7405           <1>       jz    short sysrel1
0000D928 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:
0000D92D FA             <1>       cli   ; 14/10/2015
                        <1> sysrel2:
                        <1>       ; 28/02/2017
                        <1>       ; Check if there is a (delayed) callback for current user/process
0000D92E A0[D7030300]   <1>       mov   al, [u.irqwait]
0000D933 240F           <1>       and   al, 0Fh ; is there a waiting IRQ callback service ?
0000D935 7444           <1>       jz    short sysrel8 ; no
                        <1>
                        <1>       ; Set return to IRQ callback service and return from the service
0000D937 0FB6D8         <1>       movzx ebx, al
0000D93A 883D[D7030300] <1>       mov   [u.irqwait], bh ; 0 ; reset
0000D940 8A9B[86490100] <1>       mov   bl, [ebx+IRQenum] ; (available) IRQ index +1 (1 to 9)
                        <1>       ; 01/03/2017
0000D946 FECB           <1>       dec   bl ; IRQ index number, 0 to 8
0000D948 7831           <1>       js    short sysrel8 ; 0 -> FFh (not in use!?)
                        <1>       ;
0000D94A A0[B3030300]   <1>       mov   al, [u.uno] ; current process (user) number
0000D94F 3883[F29B0100] <1>       cmp   [ebx+IRQ.owner], al
0000D955 7524           <1>       jne   short sysrel8 ; it is not the current user/process !?
0000D957 F683[049C0100]01 <1>     test  byte [ebx+IRQ.method], 1 ; callback ?
0000D95E 741B           <1>       jz    short sysrel8 ; not a callback method !?
                        <1>
0000D960 8B93[169C0100] <1>       mov   edx, [ebx+IRQ.addr] ; IRQ callback service address (virtual)
0000D966 C605[D8030300]01 <1>     mov   byte [u.r_lock], 1 ; IRQ callback service in progress flag
                        <1>
0000D96D E834400000     <1>       call  wswap ; save user's registers & status
                        <1>                   ;      (for return from IRQ callback service)
                        <1>
```

```
0000D972 8B2D[5C030300]      <1>       mov    ebp, [u.sp]; kernel's stack, points to EIP (user)
0000D978 895500              <1>       mov    [ebp], edx ; IRQ call back service address
                             <1> sysrel8:
0000D97B FE0D[5B030300]      <1>       dec    byte [sysflg]
                             <1>              ; decb sysflg / turn system flag off
                             <1>
0000D981 A1[B8030300]        <1>       mov    eax, [u.pgdir]
0000D986 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
0000D989 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>       ;
0000D98A A1[64030300]        <1>       mov    eax, [u.r0]  ; ((return value in EAX))
0000D98F 0FA9                <1>       pop    gs
0000D991 0FA1                <1>       pop    fs
0000D993 07                  <1>       pop    es
0000D994 1F                  <1>       pop    ds
                             <1>       ;or    word [esp+8], 200h ; 22/01/2017 ; force enabling interrupts
0000D995 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
0000D996 803D[D4030300]00    <1>       cmp    byte [u.t_lock], 0 ; TIMER INT LOCK
                             <1>       ;je    short sysrel0 ; classic (Retro UNIX 386 type) sysrele
0000D99D 7734                <1>       ja     short sysrel3
                             <1>       ; 27/02/2017
0000D99F 803D[D8030300]00    <1>       cmp    byte [u.r_lock], 0 ; IRQ callback lock
0000D9A6 0F8667FFFFFF        <1>       jna    sysrel0 ; classic sysrele ; 24/03/2017
0000D9AC E859000000          <1>       call   sysrel7
0000D9B1 803D[D8030300]00    <1>       cmp    byte [u.r_lock], 0 ; IRQ callback service lock
0000D9B8 7628                <1>       jna    short sysrel4
0000D9BA C605[D8030300]00    <1>       mov    byte [u.r_lock], 0 ; reset
                             <1>       ;mov   byte [u.irqwait], 0 ; reset ; 28/02/2017
0000D9C1 A0[D9030300]        <1>       mov    al, [u.r_mode]
0000D9C6 08C0                <1>       or     al, al
0000D9C8 7518                <1>       jnz    short sysrel4
0000D9CA FEC8                <1>       dec    al
0000D9CC A2[D9030300]        <1>       mov    [u.r_mode], al ; 0FFh ; not necessary !?
0000D9D1 EB32                <1>       jmp    short sysrel6
                             <1> sysrel3:
                             <1>       ; 27/02/2017
0000D9D3 E832000000          <1>       call   sysrel7
                             <1>       ; 14/01/2017
0000D9D8 28C0                <1>       sub    al, al
0000D9DA 3805[D4030300]      <1>       cmp    [u.t_lock], al ; 0 ; TIMER INT LOCK
0000D9E0 770E                <1>       ja     short sysrel5 ; yes
                             <1> sysrel4:
                             <1>       ; 29/01/2017
0000D9E2 8B44241C            <1>       mov    eax, [esp+28] ; eax
0000D9E6 A3[64030300]        <1>       mov    [u.r0], eax
0000D9EB E93EFFFFFF          <1>       jmp    sysrel2
                             <1> sysrel5:
0000D9F0 A2[D4030300]        <1>       mov    [u.t_lock], al ; 0 ; reset
0000D9F5 A0[D5030300]        <1>       mov    al, [u.t_mode]
0000D9FA 20C0                <1>       and    al, al
                             <1>       ;jnz   short sysrel2 ; 0FFh ; user mode
0000D9FC 75E4                <1>       jnz    short sysrel4 ; 29/01/2017
0000D9FE FEC8                <1>       dec    al
0000DA00 A2[D5030300]        <1>       mov    [u.t_mode], al ; 0FFh ; not necessary !?
                             <1> sysrel6:
                             <1>       ; cpu will continue from the interrupted sytem call addr
0000DA05 61                  <1>       popad         ; edi, esi, ebp, esp, ebx, edx, ecx, eax
0000DA06 83C410              <1>       add    esp, 16      ; pass segment segisters: ds, es, fs, gs
0000DA09 CF                  <1>       iretd         ; eip, cs, eflags
                             <1>
                             <1> sysrel7:
0000DA0A 0FB61D[B3030300]    <1>       movzx  ebx, byte [u.uno] ; current process number
0000DA11 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
0000DA15 8B83[BC000300]      <1>       mov     eax, [ebx+p.upage-4] ; UPAGE address
0000DA1B FA                  <1>       cli    ; disable interrupts till 'iretd'
0000DA1C E9BD3F0000          <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>       ;
0000DA21 FE05[B2030300]         <1>       inc   byte [u.bsys]
                                <1>       ;
0000DA27 8B1D[5C030300]         <1>       mov   ebx, [u.sp] ; esp at the beginning of 'sysent'
0000DA2D 8B03                   <1>       mov   eax, [ebx] ; EIP (return address, not 'INT 30h' address)
0000DA2F 83E802                 <1>       sub   eax, 2 ; CDh, ##h
0000DA32 E8D668FFFF             <1>       call  dwordtohex
0000DA37 8915[5F470100]         <1>       mov   [eip_str], edx
0000DA3D A3[63470100]           <1>       mov   [eip_str+4], eax
0000DA42 A1[64030300]           <1>       mov   eax, [u.r0]
0000DA47 E8C168FFFF             <1>       call  dwordtohex
0000DA4C 8915[4E470100]         <1>       mov   [eax_str], edx
0000DA52 A3[52470100]           <1>       mov   [eax_str+4], eax
                                <1>
0000DA57 66C705[43470100]34-    <1>       mov   word [int_num_str], SYSCALL_INT_NUM ; 25/12/2016
0000DA5F 30                     <1>
                                <1>
0000DA60 BE[15470100]           <1>       mov   esi, ifc_msg ; "invalid funtion call !" msg (trdosk9.s)
0000DA65 E8EB9AFFFF             <1>       call  print_msg
                                <1>
0000DA6A 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
0000DA6C FB                     <1>       sti
                                <1>       ; 07/12/2013
0000DA6D 66FF05[AC030300]       <1>       inc   word [u.quit]
0000DA74 7408                   <1>       jz    short intrct0 ; FFFFh -> 0
0000DA76 66FF0D[AC030300]       <1>       dec   word [u.quit]
                                <1>       ; 16/04/2015
0000DA7D C3                     <1>       retn
                                <1> intrct0:
0000DA7E 58                     <1>       pop   eax ; call intract -> retn
                                <1>       ;
0000DA7F 31C0                   <1>       xor   eax, eax
0000DA81 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
0000DA83 6648           <1>       dec    ax ; 0
0000DA85 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
0000DA8B FA             <1>       cli    ; disable interrupts
                        <1>       ; 23/01/2017 - reset timer frequency (to 18.2Hz)
0000DA8C B036           <1>       mov    al, 00110110b ; 36h
0000DA8E E643           <1>       out    43h, al
0000DA90 28C0           <1>       sub    al, al ; 0
0000DA92 E640           <1>       out    40h, al ; LB
0000DA94 E640           <1>       out    40h, al ; HB
                        <1>       ;
0000DA96 0FB61D[B3030300] <1>     movzx  ebx, byte [u.uno]
                        <1>       ;mov   bl, [u.uno] ; process number of dying process
0000DA9D 3883[FF000300] <1>       cmp    byte [ebx+p.timer-1], al ; 0
0000DAA3 763A           <1>       jna    short sysexit_12 ; no timer events for this process
0000DAA5 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
0000DAAB 8A0D[53960100] <1>       mov    cl, [timer_events] ; 14/11/2017
                        <1>       ;cli   ; disable interrupts
0000DAB1 B410           <1>       mov    ah, 16 ; number of available timer events
0000DAB3 BE[60040300]   <1>       mov    esi, timer_set ; beginning address of timer events
                        <1> sysexit_7:
0000DAB8 8A06           <1>       mov    al, [esi] ; process number (of timer event)
0000DABA 38D8           <1>       cmp    al, bl ; process number comparison
0000DABC 7411           <1>       je     short sysexit_10
0000DABE 20C0           <1>       and    al, al
0000DAC0 7404           <1>       jz     short sysexit_9
                        <1> sysexit_8:
0000DAC2 FEC9           <1>       dec    cl
0000DAC4 7416           <1>       jz     short sysexit_11
                        <1> sysexit_9:
0000DAC6 FECC           <1>       dec    ah
0000DAC8 7415           <1>       jz     short sysexit_12
0000DACA 83C610         <1>       add    esi, 16
0000DACD EBE9           <1>       jmp    short sysexit_7
                        <1>
                        <1> sysexit_10:
                        <1>       ;mov   byte [esi], 0
0000DACF 66C7060000     <1>       mov    word [esi], 0
                        <1>       ;mov   dword [esi+12], 0
                        <1>       ;
0000DAD4 FE0D[53960100] <1>       dec    byte [timer_events] ; 02/01/2017
                        <1>       ;
0000DADA EBE6           <1>       jmp    short sysexit_8
                        <1>
                        <1> sysexit_11:
0000DADC 6629C0         <1>       sub    ax, ax ; 0 ; 26/02/2017
                        <1> sysexit_12:
                        <1>       ; 26/02/2017 (Unlink IRQ callbacks belong to the user)
0000DADF 803D[D6030300]00 <1>     cmp    byte [u.irqc], 0 ; Count of IRQ callbacks
0000DAE6 7E2E           <1>       jng    short sysexit_16 ; zero or invalid
                        <1>       ; 28/02/2017
                        <1>       ; clear IRQ callback flags (for 'sysrele' and 'sysret')
0000DAE8 A2[D7030300]   <1>       mov    [u.irqwait], al ; 0 ; force to clear waiting flag
0000DAED A2[D8030300]   <1>       mov    [u.r_lock], al ; 0 ; force to clear busy flag
0000DAF2 BE[F29B0100]   <1>       mov    esi, IRQ.owner
                        <1> sysexit_13:
0000DAF7 AC             <1>       lodsb
0000DAF8 3A05[B3030300] <1>       cmp    al, [u.uno] ; owner = current user ?
0000DAFE 750C           <1>       jne    short sysexit_14
```

```
0000DB00 C646FF00           <1>       mov    byte [esi-1], 0 ; owner = 0 : Free
0000DB04 FE0D[D6030300]     <1>       dec    byte [u.irqc]
0000DB0A 7408               <1>       jz     short sysexit_15
                            <1> sysexit_14:
0000DB0C 81FE[FA9B0100]     <1>       cmp    esi, IRQ.owner + 8 ; the last IRQ index number ?
0000DB12 76E3               <1>       jna    short sysexit_13 ; no
                            <1> sysexit_15:
0000DB14 30C0               <1>       xor    al, al ; 0
                            <1> sysexit_16: ; 2:
0000DB16 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
0000DB17 E8AF340000         <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
0000DB1C FEC0               <1>       inc    al
                            <1>              ; inc r1 / increment file descriptor
                            <1>       ;cmp   ax, 10
0000DB1E 3C0A               <1>       cmp    al, 10
                            <1>              ; cmp r1,$10. / end of u.fp list?
0000DB20 72F5               <1>       jb     short sysexit_1
                            <1>              ; blt 1b / no, go back
                            <1>       ;movzx ebx, byte [u.uno]
0000DB22 8A1D[B3030300]     <1>       mov    bl, [u.uno] ; 02/01/2017
                            <1>              ; movb u.uno,r1 / yes, move dying process's number to r1
0000DB28 88A3[AF000300]     <1>       mov    [ebx+p.stat-1], ah ; 0, SFREE
                            <1>              ; clrb p.stat-1(r1) / free the process
                            <1>       ; 10/04/2017
0000DB2E 381D[699C0100]     <1>       cmp    [audio_user], bl
0000DB34 7518               <1>       jne    short sysexit_17
                            <1>       ; reset audio device (current) owner and 'initializated' flag
0000DB36 883D[699C0100]     <1>       mov    [audio_user], bh ; 0
                            <1>       ; 27/05/2017
0000DB3C 8B0D[549C0100]     <1>       mov    ecx,  [audio_buffer]
0000DB42 09C9               <1>       or     ecx, ecx
0000DB44 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)
0000DB46 29C9               <1>       sub    ecx, ecx
0000DB48 890D[549C0100]     <1>       mov    [audio_buffer], ecx ; 0
                            <1>       ;pop   ebx
                            <1> sysexit_17:
                            <1>       ;shl   bx, 1
0000DB4E D0E3               <1>       shl    bl, 1
                            <1>              ; asl r1 / use r1 for index into the below tables
0000DB50 668B8B[1E000300]   <1>       mov    cx, [ebx+p.pid-2]
                            <1>              ; mov p.pid-2(r1),r3 / move dying process's name to r3
0000DB57 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
0000DB5E 30DB               <1>       xor    bl, bl ; 0
                            <1>              ; clr r2
0000DB60 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
0000DB62 80C302             <1>       add    bl, 2
                            <1>              ; add $2,r2 / search parent process table
                            <1>                        ; / for dying process's name
0000DB65 66398B[3E000300]   <1>       cmp    [ebx+p.ppid-2], cx
                            <1>              ; cmp p.ppid-2(r2),r3 / found it?
0000DB6C 7513               <1>       jne    short sysexit_4
                            <1>              ; bne 3f / no
                            <1>       ;shr   bx, 1
0000DB6E D0EB               <1>       shr    bl, 1
                            <1>              ; asr r2 / yes, it is a parent
0000DB70 80BB[AF000300]03   <1>       cmp    byte [ebx+p.stat-1], 3 ; SZOMB
                            <1>              ; cmpb p.stat-1(r2),$3 / is the child of this
                            <1>                                   ; / dying process a zombie
0000DB77 7506               <1>       jne    short sysexit_3
                            <1>              ; bne 2f / no
0000DB79 88A3[AF000300]     <1>       mov    [ebx+p.stat-1], ah ; 0, SFREE
                            <1>              ; clrb p.stat-1(r2) / yes, free the child process
                            <1> sysexit_3: ; 2:
                            <1>       ;shr   bx, 1
0000DB7F D0E3               <1>       shl    bl, 1
                            <1>              ; asl r2
                            <1> sysexit_4: ; 3:
                            <1>              ; / search the process name table
                            <1>              ; / for the dying process's parent
0000DB81 663993[1E000300]   <1>       cmp    [ebx+p.pid-2], dx
                            <1>              ; cmp p.pid-2(r2),r4 / found it?
0000DB88 7502               <1>       jne    short sysexit_5
                            <1>              ; bne 3f / no
0000DB8A 89DE               <1>       mov    esi, ebx
                            <1>              ; mov r2,r5 / yes, put index to p.pid table (parents
                            <1>                        ; / process # x2) in r5
                            <1> sysexit_5: ; 3:
                            <1>       ;cmp   bx, nproc + nproc
0000DB8C 80FB20             <1>       cmp    bl, nproc + nproc
                            <1>              ; cmp r2,$nproc+nproc / has whole table been searched?
0000DB8F 72D1               <1>       jb     short sysexit_2
                            <1>              ; blt 1b / no, go back
```

```
800                                     <1>             ; mov r5,r1 / yes, r1 now has parents process # x2
801 0000DB91 21F6                       <1>         and     esi, esi ; r5=r1
802 0000DB93 7436                       <1>         jz      short sysexit_6
803                                     <1>             ; beq 2f / no parent has been found.
804                                     <1>                   ; / The process just dies
805 0000DB95 66D1EE                     <1>         shr     si, 1
806                                     <1>             ; asr r1 / set up index to p.stat
807 0000DB98 8A86[AF000300]             <1>         mov     al, [esi+p.stat-1]
808                                     <1>             ; movb p.stat-1(r1),r2 / move status of parent to r2
809 0000DB9E 20C0                       <1>         and     al, al
810 0000DBA0 7429                       <1>         jz      short sysexit_6
811                                     <1>             ; beq 2f / if its been freed, 2f
812 0000DBA2 3C03                       <1>         cmp     al, 3
813                                     <1>             ; cmp r2,$3 / is parent a zombie?
814 0000DBA4 7425                       <1>         je      short sysexit_6
815                                     <1>             ; beq 2f / yes, 2f
816                                     <1>         ; BH = 0
817 0000DBA6 8A1D[B3030300]             <1>         mov     bl, [u.uno]
818                                     <1>             ; movb u.uno,r3 / move dying process's number to r3
819 0000DBAC C683[AF000300]03           <1>         mov     byte [ebx+p.stat-1], 3  ; SZOMB
820                                     <1>             ; movb $3,p.stat-1(r3) / make the process a zombie
821 0000DBB3 3C01                       <1>         cmp     al, 1 ; SRUN
822 0000DBB5 7414                       <1>         je      short sysexit_6
823                                     <1>         ;cmp    al, 2
824                                     <1>             ; cmp r2,$2 / is the parent waiting for
825                                     <1>                     ; / this child to die
826                                     <1>         ;jne    short sysexit_6
827                                     <1>             ; bne 2f / yes, notify parent not to wait any more
828                                     <1>         ; p.stat = 2 --> waiting
829                                     <1>         ; p.stat = 4 --> sleeping
830 0000DBB7 C686[AF000300]01           <1>         mov     byte [esi+p.stat-1], 1 ; SRUN
831                                     <1>         ;dec    byte [esi+p.stat-1]
832                                     <1>             ; decb p.stat-1(r1) / awaken it by putting it (parent)
833 0000DBBE 6689F0                     <1>         mov     ax, si ; r1  (process number in AL)
834                                     <1>         ;
835                                     <1>         ;mov    ebx, runq + 4
836                                     <1>             ; mov $runq+4,r2 / on the runq
837 0000DBC1 BB[54030300]               <1>         mov     ebx, runq+2 ; normal run queue ; 02/01/2017
838 0000DBC6 E84B3E0000                 <1>         call    putlu
839                                     <1>             ; jsr r0, putlu
840                                     <1> sysexit_6:
841                                     <1>             ; / the process dies
842 0000DBCB C605[B3030300]00           <1>         mov     byte [u.uno], 0
843                                     <1>             ; clrb u.uno / put zero as the process number,
844                                     <1>                 ; / so "swap" will
845 0000DBD2 E8413D0000                 <1>         call    swap
846                                     <1>             ; jsr r0,swap / overwrite process with another process
847                                     <1> hlt_sys:
848                                     <1>         ;sti
849                                     <1> hlts0:
850 0000DBD7 F4                         <1>         hlt
851 0000DBD8 EBFD                       <1>         jmp     short hlts0
852                                     <1>             ; 0 / and thereby kill it; halt?
853                                     <1>
854                                     <1> syswait: ; < wait for a processs to die >
855                                     <1>         ; 17/09/2015
856                                     <1>         ; 02/09/2015
857                                     <1>         ; 01/09/2015
858                                     <1>         ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
859                                     <1>         ; 24/05/2013 - 05/02/2014 (Retro UNIX 8086 v1)
860                                     <1>         ;
861                                     <1>         ; 'syswait' waits for a process die.
862                                     <1>         ; It works in following way:
863                                     <1>         ;    1) From the parent process number, the parent's
864                                     <1>         ;      process name is found. The p.ppid table of parent
865                                     <1>         ;      names is then searched for this process name.
866                                     <1>         ;      If a match occurs, r2 contains child's process
867                                     <1>         ;      number. The child status is checked to see if it is
868                                     <1>         ;      a zombie, i.e; dead but not waited for (p.stat=3)
869                                     <1>         ;      If it is, the child process is freed and it's name
870                                     <1>         ;      is put in (u.r0). A return is then made via 'sysret'.
871                                     <1>         ;      If the child is not a zombie, nothing happens and
872                                     <1>         ;      the search goes on through the p.ppid table until
873                                     <1>         ;      all processes are checked or a zombie is found.
874                                     <1>         ;    2) If no zombies are found, a check is made to see if
875                                     <1>         ;      there are any children at all. If there are none,
876                                     <1>         ;      an error return is made. If there are, the parent's
877                                     <1>         ;      status is set to 2 (waiting for child to die),
878                                     <1>         ;      the parent is swapped out, and a branch to 'syswait'
879                                     <1>         ;      is made to wait on the next process.
880                                     <1>         ;
881                                     <1>         ; Calling sequence:
882                                     <1>         ;       ?
883                                     <1>         ; Arguments:
884                                     <1>         ;       -
885                                     <1>         ; Inputs: -
886                                     <1>         ; Outputs: if zombie found, it's name put in u.r0.
887                                     <1>         ; ..........................................................
888                                     <1>         ;
889                                     <1>
890                                     <1> ; / wait for a process to die
891                                     <1>
892                                     <1> syswait_0:
893 0000DBDA 0FB61D[B3030300]           <1>         movzx   ebx, byte [u.uno] ; 01/09/2015
894                                     <1>             ; movb u.uno,r1 / put parents process number in r1
895 0000DBE1 D0E3                       <1>         shl     bl, 1
896                                     <1>         ;shl    bx, 1
897                                     <1>             ; asl r1 / x2 to get index into p.pid table
898 0000DBE3 668B83[1E000300]           <1>         mov     ax, [ebx+p.pid-2]
899                                     <1>             ; mov p.pid-2(r1),r1 / get the name of this process
900 0000DBEA 31F6                       <1>         xor     esi, esi
901                                     <1>             ; clr r2
902 0000DBEC 31C9                       <1>         xor     ecx, ecx ; 30/10/2013
903                                     <1>         ;xor    cl, cl
904                                     <1>             ; clr r3 / initialize reg 3
```

```
                            <1> syswait_1: ; 1:
0000DBEE 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
0000DBF2 663B86[3E000300]   <1>       cmp   ax, [esi+p.ppid-2]
                            <1>             ; cmp p.ppid-2(r2),r1 / r2 will contain the childs
                            <1>                                   ; / process number
0000DBF9 7535               <1>       jne   short syswait_3
                            <1>             ;bne 3f / branch if no match of parent process name
                            <1>       ;inc  cx
0000DBFB FEC1               <1>       inc   cl
                            <1>             ;inc r3 / yes, a match, r3 indicates number of children
0000DBFD 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).
0000DC00 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?
0000DC07 7524               <1>       jne   short syswait_2
                            <1>             ; bne 2f / no, skip it
0000DC09 88BE[AF000300]     <1>       mov   [esi+p.stat-1], bh ; 0
                            <1>             ; clrb p.stat-1(r2) / yes, free it
0000DC0F 66D1E6             <1>       shl   si, 1
                            <1>             ; asl r2 / r2x2 to get index into p.pid table
0000DC12 0FB786[1E000300]   <1>       movzx eax, word [esi+p.pid-2]
0000DC19 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>       ;
0000DC1E 6629C0             <1>       sub   ax, ax
0000DC21 668986[3E000300]   <1>       mov   [esi+p.ppid-2], ax ; 0 ; 17/09/2015
0000DC28 E9D1FCFFFF         <1>       jmp   sysret0 ; ax = 0
                            <1>       ;
                            <1>       ;jmp  sysret
                            <1>             ; br sysret1 / return cause child is dead
                            <1> syswait_2: ; 2:
0000DC2D 66D1E6             <1>       shl   si, 1
                            <1>             ; asl r2 / r2x2 to get index into p.ppid table
                            <1> syswait_3: ; 3:
0000DC30 6683FE20           <1>       cmp   si, nproc+nproc
                            <1>             ; cmp r2,$nproc+nproc / have all processes been checked?
0000DC34 72B8               <1>       jb    short syswait_1
                            <1>             ; blt 1b / no, continue search
                            <1>       ;and  cx, cx
0000DC36 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
0000DC38 750B               <1>       jnz   short syswait_4
                            <1>       ;jz   error
                            <1>             ; beq error1 / there are no children, error
0000DC3A 890D[64030300]     <1>       mov   [u.r0], ecx ; 0
0000DC40 E997FCFFFF         <1>       jmp   error
                            <1> syswait_4:
0000DC45 8A1D[B3030300]     <1>       mov   bl, [u.uno]
                            <1>             ; movb u.uno,r1 / there are children so put
                            <1>                         ; / parent process number in r1
0000DC4B 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
0000DC51 E8C23C0000         <1>       call  swap
                            <1>             ; jsr r0,swap / swap it out, because it's waiting
0000DC56 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'.
```

```
1010                                    <1>       ;    5) A new process name is gotten from 'mpid' (actually
1011                                    <1>       ;      it is a unique number) and is put in the child's unique
1012                                    <1>       ;      identifier; process id (p.pid).
1013                                    <1>       ;    6) The process name of the parent is then obtained and
1014                                    <1>       ;      placed in the unique identifier of the parent process
1015                                    <1>       ;      name is then put in 'u.r0'.
1016                                    <1>       ;    7) The child process is then written out on disk by
1017                                    <1>       ;      'wswap',i.e., the parent process is copied onto disk
1018                                    <1>       ;      and the child is born. (The child process is written
1019                                    <1>       ;      out on disk/drum with 'u.uno' being the child process
1020                                    <1>       ;      number.)
1021                                    <1>       ;    8) The parent process number is then restored to 'u.uno'.
1022                                    <1>       ;    9) The child process name is put in 'u.r0'.
1023                                    <1>       ;   10) The pc on the stack sp + 18 is incremented by 2 to
1024                                    <1>       ;      create the return address for the parent process.
1025                                    <1>       ;   11) The 'u.fp' list as then searched to see what files
1026                                    <1>       ;      the parent has opened. For each file the parent has
1027                                    <1>       ;      opened, the corresponding 'fsp' entry must be updated
1028                                    <1>       ;      to indicate that the child process also has opened
1029                                    <1>       ;      the file. A branch to 'sysret' is then made.

1030                                    <1>       ;
1031                                    <1>       ; Calling sequence:
1032                                    <1>       ;     from shell ?
1033                                    <1>       ; Arguments:
1034                                    <1>       ;     -
1035                                    <1>       ; Inputs: -
1036                                    <1>       ; Outputs: *u.r0 - child process name
1037                                    <1>       ; ............................................................
1038                                    <1>       ;
1039                                    <1>       ; Retro UNIX 8086 v1 modification:
1040                                    <1>       ;     AX = r0 = PID (>0) (at the return of 'sysfork')
1041                                    <1>       ;     = process id of child a parent process returns
1042                                    <1>       ;     = process id of parent when a child process returns
1043                                    <1>       ;
1044                                    <1>       ;       In original UNIX v1, sysfork is called and returns as
1045                                    <1>       ;     in following manner: (with an example: c library, fork)
1046                                    <1>       ;
1047                                    <1>       ;     1:
1048                                    <1>       ;           sys   fork
1049                                    <1>       ;                 br 1f  / child process returns here
1050                                    <1>       ;           bes   2f     / parent process returns here
1051                                    <1>       ;           / pid of new process in r0
1052                                    <1>       ;           rts   pc
1053                                    <1>       ;     2: / parent process condionally branches here
1054                                    <1>       ;           mov   $-1,r0 / pid = -1 means error return
1055                                    <1>       ;           rts   pc
1056                                    <1>       ;
1057                                    <1>       ;     1: / child process brances here
1058                                    <1>       ;           clr   r0   / pid = 0 in child process
1059                                    <1>       ;           rts   pc
1060                                    <1>       ;
1061                                    <1>       ;     In UNIX v7x86 (386) by Robert Nordier (1999)
1062                                    <1>       ;           // pid = fork();
1063                                    <1>       ;           //
1064                                    <1>       ;           // pid == 0 in child process;
1065                                    <1>       ;           // pid == -1 means error return
1066                                    <1>       ;           // in child,
1067                                    <1>       ;           //    parents id is in par_uid if needed
1068                                    <1>       ;
1069                                    <1>       ;           _fork:
1070                                    <1>       ;                 mov   $.fork,eax
1071                                    <1>       ;                 int   $0x30
1072                                    <1>       ;                 jmp   1f
1073                                    <1>       ;                 jnc   2f
1074                                    <1>       ;                 jmp   cerror
1075                                    <1>       ;           1:
1076                                    <1>       ;                 mov   eax,_par_uid
1077                                    <1>       ;                 xor   eax,eax
1078                                    <1>       ;           2:
1079                                    <1>       ;                 ret
1080                                    <1>       ;
1081                                    <1>       ;     In Retro UNIX 8086 v1,
1082                                    <1>       ;     'sysfork' returns in following manner:
1083                                    <1>       ;
1084                                    <1>       ;           mov   ax, sys_fork
1085                                    <1>       ;           mov   bx, offset @f ; routine for child
1086                                    <1>       ;           int   20h
1087                                    <1>       ;           jc    error
1088                                    <1>       ;
1089                                    <1>       ;     ; Routine for parent process here (just after 'jc')
1090                                    <1>       ;           mov   word ptr [pid_of_child], ax
1091                                    <1>       ;           jmp   next_routine_for_parent
1092                                    <1>       ;
1093                                    <1>       ;     @@: ; routine for child process here
1094                                    <1>       ;           ....
1095                                    <1>       ;     NOTE: 'sysfork' returns to specified offset
1096                                    <1>       ;           for child process by using BX input.
1097                                    <1>       ;           (at first, parent process will return then
1098                                    <1>       ;           child process will return -after swapped in-
1099                                    <1>       ;           'syswait' is needed in parent process
1100                                    <1>       ;           if return from child process will be waited for.)
1101                                    <1>       ;
1102                                    <1>
1103                                    <1> ; / create a new process
1104                                    <1>       ; EBX = return address for child process
1105                                    <1>           ; (Retro UNIX 8086 v1 modification !)
1106 0000DC58 31F6                     <1>       xor   esi, esi
1107                                    <1>             ; clr r1
1108                                    <1> sysfork_1: ; 1: / search p.stat table for unused process number
1109 0000DC5A 46                       <1>       inc   esi
1110                                    <1>             ; inc r1
1111 0000DC5B 80BE[AF000300]00         <1>       cmp   byte [esi+p.stat-1], 0 ; SFREE, 05/02/2014
1112                                    <1>             ; tstb p.stat-1(r1) / is process active, unused, dead
1113 0000DC62 760B                     <1>       jna   short sysfork_2
```

```
1114                                <1>             ; beq 1f / it's unused so branch
1115 0000DC64 6683FE10              <1>       cmp   si, nproc
1116                                <1>             ; cmp r1,$nproc / all processes checked
1117 0000DC68 72F0                  <1>       jb    short sysfork_1
1118                                <1>             ; blt 1b / no, branch back
1119                                <1>       ;
1120                                <1>       ; Retro UNIX 8086 v1. modification:
1121                                <1>       ;     Parent process returns from 'sysfork' to address
1122                                <1>       ;     which is just after 'sysfork' system call in parent
1123                                <1>       ;     process. Child process returns to address which is put
1124                                <1>       ;     in BX register by parent process for 'sysfork'.
1125                                <1>       ;
1126                                <1>             ;add $2,18.(sp) / add 2 to pc when trap occured, points
1127                                <1>                             ; / to old process return
1128                                <1>             ; br error1 / no room for a new process
1129 0000DC6A E96DFCFFFF            <1>       jmp   error
1130                                <1> sysfork_2: ; 1:
1131 0000DC6F E85F7FFFFF            <1>       call  allocate_page
1132 0000DC74 0F8262FCFFFF          <1>       jc    error
1133 0000DC7A 50                    <1>       push  eax   ; UPAGE (user structure page) address
1134                                <1>       ; Retro UNIX 386 v1 modification!
1135 0000DC7B E86281FFFF            <1>       call  duplicate_page_dir
1136                                <1>             ; EAX = New page directory
1137 0000DC80 730B                  <1>       jnc   short sysfork_3
1138 0000DC82 58                    <1>       pop   eax   ; UPAGE (user structure page) address
1139 0000DC83 E82981FFFF            <1>       call  deallocate_page
1140 0000DC88 E94FFCFFFF            <1>       jmp   error
1141                                <1> sysfork_3:
1142                                <1>       ; Retro UNIX 386 v1 modification !
1143 0000DC8D 56                    <1>       push  esi
1144 0000DC8E E8133D0000            <1>       call  wswap ; save current user (u) structure, user registers
1145                                <1>                   ; and interrupt return components (for IRET)
1146 0000DC93 8705[B8030300]        <1>       xchg  eax, [u.pgdir] ; page directory of the child process
1147 0000DC99 A3[BC030300]          <1>       mov   [u.ppgdir], eax ; page directory of the parent process
1148 0000DC9E 5E                    <1>       pop   esi
1149 0000DC9F 58                    <1>       pop   eax   ; UPAGE (user structure page) address
1150                                <1>             ; [u.usp] = esp
1151 0000DCA0 89F7                  <1>       mov   edi, esi
1152 0000DCA2 66C1E702              <1>       shl   di, 2
1153 0000DCA6 8987[BC000300]        <1>       mov   [edi+p.upage-4], eax ; memory page for 'user' struct
1154 0000DCAC A3[B4030300]          <1>       mov   [u.upage], eax ; memory page for 'user' struct (child)
1155                                <1>       ; 28/08/2015
1156 0000DCB1 0FB605[B3030300]      <1>       movzx eax, byte [u.uno] ; parent process number
1157                                <1>             ; movb u.uno,-(sp) / save parent process number
1158 0000DCB8 89C7                  <1>       mov   edi, eax
1159 0000DCBA 50                    <1>         push eax ; **
1160 0000DCBB 8A87[7F000300]        <1>       mov    al, [edi+p.ttyc-1] ; console tty (parent)
1161                                <1>       ; 18/09/2015
1162                                <1>       ;mov    [esi+p.ttyc-1], al ; set child's console tty
1163                                <1>       ;mov    [esi+p.waitc-1], ah ; 0 ; reset child's wait channel
1164 0000DCC1 668986[7F000300]      <1>       mov    [esi+p.ttyc-1], ax ; al - set child's console tty
1165                                <1>                                ; ah - reset child's wait channel
1166 0000DCC8 89F0                  <1>       mov   eax, esi
1167 0000DCCA A2[B3030300]          <1>       mov   [u.uno], al ; child process number
1168                                <1>             ;movb r1,u.uno / set child process number to r1
1169 0000DCCF FE86[AF000300]        <1>         inc    byte [esi+p.stat-1] ; 1, SRUN, 05/02/2014
1170                                <1>             ; incb p.stat-1(r1) / set p.stat entry for child
1171                                <1>                           ; / process to active status
1172                                <1>             ; mov u.ttyp,r2 / put pointer to parent process'
1173                                <1>                           ; / control tty buffer in r2
1174                                <1>                ; beq 2f / branch, if no such tty assigned
1175                                <1>             ; clrb 6(r2) / clear interrupt character in tty buffer
1176                                <1>       ; 2:
1177 0000DCD5 53                    <1>       push  ebx  ; * return address for the child process
1178                                <1>                  ; * Retro UNIX 8086 v1 feature only !
1179                                <1>       ; (Retro UNIX 8086 v1 modification!)
1180                                <1>             ; mov $runq+4,r2
1181 0000DCD6 BB[54030300]          <1>       mov   ebx, runq+2 ; normal run queue ; 02/01/2017
1182 0000DCDB E8363D0000            <1>       call  putlu
1183                                <1>             ; jsr r0,putlu / put child process on lowest priority
1184                                <1>                           ; / run queue
1185 0000DCE0 66D1E6                <1>       shl   si, 1
1186                                <1>             ; asl r1 / multiply r1 by 2 to get index
1187                                <1>                    ; / into p.pid table
1188 0000DCE3 66FF05[4E030300]      <1>       inc   word [mpid]
1189                                <1>             ; inc mpid / increment m.pid; get a new process name
1190 0000DCEA 66A1[4E030300]        <1>       mov   ax, [mpid]
1191 0000DCF0 668986[1E000300]      <1>       mov   [esi+p.pid-2], ax
1192                                <1>             ;mov mpid,p.pid-2(r1) / put new process name
1193                                <1>                                   ; / in child process' name slot
1194 0000DCF7 5A                    <1>       pop   edx  ; * return address for the child process
1195                                <1>                  ; * Retro UNIX 8086 v1 feature only !
1196 0000DCF8 5B                    <1>       pop   ebx  ; **
1197                                <1>       ;mov  ebx, [esp] ; ** parent process number
1198                                <1>             ; movb (sp),r2 / put parent process number in r2
1199 0000DCF9 66D1E3                <1>       shl   bx, 1
1200                                <1>             ;asl r2 / multiply by 2 to get index into below tables
1201                                <1>       ;movzx eax, word [ebx+p.pid-2]
1202 0000DCFC 668B83[1E000300]      <1>       mov   ax, [ebx+p.pid-2]
1203                                <1>             ; mov p.pid-2(r2),r2 / get process name of parent
1204                                <1>                                  ; / process
1205 0000DD03 668986[3E000300]      <1>       mov   [esi+p.ppid-2], ax
1206                                <1>             ; mov r2,p.ppid-2(r1) / put parent process name
1207                                <1>                             ; / in parent process slot for child
1208 0000DD0A A3[64030300]          <1>       mov   [u.r0], eax
1209                                <1>             ; mov r2,*u.r0 / put parent process name on stack
1210                                <1>                            ; / at location where r0 was saved
1211 0000DD0F 8B2D[5C030300]        <1>       mov   ebp, [u.sp] ; points to return address (EIP for IRET)
1212 0000DD15 895500                <1>       mov   [ebp], edx ; *, CS:EIP -> EIP
1213                                <1>                        ; * return address for the child process
1214                                <1>             ; mov $sysret1,-(sp) /
1215                                <1>             ; mov sp,u.usp / contents of sp at the time when
1216                                <1>                            ; / user is swapped out
1217                                <1>             ; mov $sstack,sp / point sp to swapping stack space
1218                                <1>       ; 04/09/2015 - 01/09/2015
```

```
1219                                <1>       ; [u.usp] = esp
1220 0000DD18 68[FCD80000]          <1>       push   sysret ; ***
1221 0000DD1D 8925[60030300]        <1>       mov    [u.usp], esp ; points to 'sysret' address (***)
1222                                <1>                           ; (for child process)
1223 0000DD23 31C0                  <1>       xor    eax, eax
1224 0000DD25 66A3[94030300]        <1>       mov    [u.ttyp], ax ; 0
1225                                <1>       ;
1226 0000DD2B E8763C0000            <1>       call   wswap ; Retro UNIX 8086 v1 modification !
1227                                <1>              ;jsr r0,wswap / put child process out on drum
1228                                <1>              ;jsr r0,unpack / unpack user stack
1229                                <1>              ;mov u.usp,sp / restore user stack pointer
1230                                <1>              ; tst (sp)+ / bump stack pointer
1231                                <1>       ; Retro UNIX 386 v1 modification !
1232 0000DD30 58                    <1>       pop    eax ; ***
1233 0000DD31 66D1E3                <1>       shl    bx, 1
1234 0000DD34 8B83[BC000300]        <1>       mov     eax, [ebx+p.upage-4] ; UPAGE address ; 14/05/2015
1235 0000DD3A E89F3C0000            <1>       call   rswap ; restore parent process 'u' structure,
1236                                <1>                    ; registers and return address (for IRET)
1237                                <1>              ;movb (sp)+,u.uno / put parent process number in u.uno
1238 0000DD3F 0FB705[4E030300]      <1>         movzx   eax, word [mpid]
1239 0000DD46 A3[64030300]          <1>       mov    [u.r0], eax
1240                                <1>              ; mov mpid,*u.r0 / put child process name on stack
1241                                <1>                             ; / where r0 was saved
1242                                <1>              ; add $2,18.(sp) / add 2 to pc on stack; gives parent
1243                                <1>                               ; / process return
1244                                <1>       ;xor   ebx, ebx
1245 0000DD4B 31F6                  <1>       xor     esi, esi
1246                                <1>              ;clr r1
1247                                <1> sysfork_4: ; 1: / search u.fp list to find the files
1248                                <1>             ; / opened by the parent process
1249                                <1>       ; 01/09/2015
1250                                <1>       ;xor   bh, bh
1251                                <1>       ;mov   bl, [esi+u.fp]
1252 0000DD4D 8A86[6A030300]        <1>       mov    al, [esi+u.fp]
1253                                <1>              ; movb u.fp(r1),r2 / get an open file for this process
1254                                <1>         ;or      bl, bl
1255 0000DD53 08C0                  <1>       or     al, al
1256 0000DD55 740D                  <1>       jz     short sysfork_5
1257                                <1>              ; beq 2f / file has not been opened by parent,
1258                                <1>                     ; / so branch
1259 0000DD57 B40A                  <1>       mov    ah, 10 ; Retro UNIX 386 v1 fsp structure size = 10 bytes
1260 0000DD59 F6E4                  <1>       mul    ah
1261                                <1>       ;movzx ebx, ax
1262 0000DD5B 6689C3                <1>       mov    bx, ax
1263                                <1>       ;shl     bx, 3
1264                                <1>              ; asl r2 / multiply by 8
1265                                <1>                    ; asl r2 / to get index into fsp table
1266                                <1>                    ; asl r2
1267 0000DD5E FE83[4E010300]        <1>       inc     byte [ebx+fsp-2]
1268                                <1>              ; incb fsp-2(r2) / increment number of processes
1269                                <1>                          ; / using file, because child will now be
1270                                <1>                          ; / using this file
1271                                <1> sysfork_5: ; 2:
1272 0000DD64 46                    <1>         inc     esi
1273                                <1>              ; inc r1 / get next open file
1274 0000DD65 6683FE0A              <1>         cmp     si, 10
1275                                <1>              ; cmp r1,$10. / 10. files is the maximum number which
1276                                <1>                       ; / can be opened
1277 0000DD69 72E2                  <1>       jb     short sysfork_4
1278                                <1>              ; blt 1b / check next entry
1279 0000DD6B E98CFBFFFF            <1>       jmp    sysret
1280                                <1>              ; br sysret1
1281                                <1>
1282                                <1> syscreat: ; < create file >
1283                                <1>       ; 13/11/2017
1284                                <1>       ; 27/10/2016
1285                                <1>       ; 25/10/2016, 26/10/2016
1286                                <1>       ; 15/10/2016, 16/10/2016, 17/10/2016
1287                                <1>       ; 10/10/2016 (TRDOS 386 = TRDOS v2.0)
1288                                <1>       ;           -derived from INT_21H.ASM-
1289                                <1>       ;             ("loc_INT21h_create_file")
1290                                <1>         ;   10/07/2011 (12/03/2011)
1291                                <1>         ;   INT 21h Function AH = 3Ch
1292                                <1>         ;   Create File
1293                                <1>         ;   INPUT
1294                                <1>         ;      CX = Attributes
1295                                <1>         ;          DS:DX= Address of zero terminaned path name
1296                                <1>         ;
1297                                <1>       ; 27/12/2015 (Retro UNIX 386 v1.1)
1298                                <1>       ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
1299                                <1>       ; 27/05/2013 (Retro UNIX 8086 v1)
1300                                <1>       ;
1301                                <1>       ; 'syscreat' called with two arguments; name and mode.
1302                                <1>       ; u.namep points to name of the file and mode is put
1303                                <1>       ; on the stack. 'namei' is called to get i-number of the file.
1304                                <1>       ; If the file aready exists, it's mode and owner remain
1305                                <1>       ; unchanged, but it is truncated to zero length. If the file
1306                                <1>       ; did not exist, an i-node is created with the new mode via
1307                                <1>       ; 'maknod' whether or not the file already existed, it is
1308                                <1>       ; open for writing. The fsp table is then searched for a free
1309                                <1>       ; entry. When a free entry is found, proper data is placed
1310                                <1>       ; in it and the number of this entry is put in the u.fp list.
1311                                <1>       ; The index to the u.fp (also know as the file descriptor)
1312                                <1>       ; is put in the user's r0.
1313                                <1>       ;
1314                                <1>       ; Calling sequence:
1315                                <1>       ;     syscreate; name; mode
1316                                <1>       ; Arguments:
1317                                <1>       ;     name - name of the file to be created
1318                                <1>       ;     mode - mode of the file to be created
1319                                <1>       ; Inputs: (arguments)
1320                                <1>       ; Outputs: *u.r0 - index to u.fp list
1321                                <1>       ;               (the file descriptor of new file)
1322                                <1>       ; ..........................................................
1323                                <1>       ;
```

```
1324                                    <1>       ; Retro UNIX 8086 v1 modification:
1325                                    <1>       ;       'syscreate' system call has two arguments; so,
1326                                    <1>       ;     * 1st argument, name is pointed to by BX register
1327                                    <1>       ;     * 2nd argument, mode is in CX register
1328                                    <1>       ;
1329                                    <1>       ;     AX register (will be restored via 'u.r0') will return
1330                                    <1>       ;     to the user with the file descriptor/number
1331                                    <1>       ;     (index to u.fp list).
1332                                    <1>       ;
1333                                    <1>       ;call  arg2
1334                                    <1>       ; * name - 'u.namep' points to address of file/path name
1335                                    <1>       ;          in the user's program segment ('u.segmnt')
1336                                    <1>       ;          with offset in BX register (as sysopen argument 1).
1337                                    <1>       ; * mode - sysopen argument 2 is in CX register
1338                                    <1>       ;          which is on top of stack.
1339                                    <1>       ;
1340                                    <1>       ; TRDOS 386 (10/10/2016)
1341                                    <1>       ;
1342                                    <1>         ; INPUT ->
1343                                    <1>         ;       CL = File Attributes
1344                                    <1>       ;            bit 0 (1) - Read only file (R)
1345                                    <1>       ;              bit 1 (1) - Hidden file (H)
1346                                    <1>         ;              bit 2 (1) - System file (R)
1347                                    <1>       ;              bit 3 (1) - Volume label/name (V)
1348                                    <1>         ;              bit 4 (1) - Subdirectory (D)
1349                                    <1>       ;            bit 5 (1) - File has been archived (A)
1350                                    <1>         ;           EBX = Pointer to filename (ASCIIZ) -path-
1351                                    <1>       ;
1352                                    <1>       ; OUTPUT ->
1353                                    <1>       ;          eax = File/Device Handle/Number (index) (AL)
1354                                    <1>       ;          cf = 1 -> Error code in AL
1355                                    <1>       ;
1356                                    <1>       ; Modified Registers: EAX (at the return of system call)
1357                                    <1>       ;
1358                                    <1>       ; Note: If the file is existing and it has not any one
1359                                    <1>       ;     of S,H,R,V,D attributes, it will be truncated
1360                                    <1>       ;     to zero length; otherwise, access error will be
1361                                    <1>       ;     returned.
1362                                    <1>
1363                                    <1> sysmkdir_0:
1364 0000DD70 F6C108                   <1>       test   cl, 08h ; Volume name
1365 0000DD73 740A                     <1>       jz     short syscreat_0
1366                                    <1>
1367                                    <1>       ; Volume name or long name creation
1368                                    <1>       ; is not permitted (in TRDOS 386)!
1369 0000DD75 B80B000000               <1>       mov    eax, ERR_FILE_ACCESS  ; 11 ; 'permission denied !'
1370 0000DD7A E926020000               <1>         jmp  sysopen_dev_err
1371                                    <1>
1372                                    <1> syscreat_0:
1373                                    <1>         ;mov [u.namep], ebx
1374 0000DD7F 51                       <1>       push   ecx
1375 0000DD80 89DE                     <1>       mov    esi, ebx
1376                                    <1>       ; file name is forced, change directory as temporary
1377                                    <1>       ;mov   ax, 1
1378                                    <1>       ;mov   [FFF_Valid], ah ; 0 ; reset ; 17/10/2016
1379                                    <1>       ;call  set_working_path
1380 0000DD82 E8E7510000               <1>       call   set_working_path_x ; 17/10/2016
1381 0000DD87 0F82D7000000             <1>       jc     syscreat_err
1382                                    <1>
1383                                    <1>       ; 16/10/2016
1384 0000DD8D 803D[77960100]00         <1>       cmp    byte [SWP_inv_fname], 0
1385 0000DD94 776C                     <1>       ja     short  syscreat_inv_fname ; invalid file name !
1386                                    <1>
1387                                    <1>       ; Here, we have a valid path and also a valid file name
1388                                    <1>       ; (Working dir has been changed if the path
1389                                    <1>       ;  -file name string- had contained a dir name.)
1390                                    <1>
1391 0000DD96 6631C0                   <1>       xor    ax, ax
1392                                    <1>       ;mov   esi, FindFile_Name
1393 0000DD99 E8E3B6FFFF               <1>       call   find_first_file
1394 0000DD9E 59                       <1>       pop    ecx
1395                                    <1>              ; ESI = Directory Entry (FindFile_DirEntry) Location
1396                                    <1>              ; EDI = Directory Buffer Directory Entry Location
1397                                    <1>              ; EAX = File Size
1398                                    <1>              ;  BL = Attributes of The File/Directory
1399                                    <1>              ;  BH = Long Name Yes/No Status (>0 is YES)
1400                                    <1>              ;  DX > 0 : Ambiguous filename chars are used
1401 0000DD9F 7269                     <1>       jc     short syscreat_1 ; file not found (the good!)
1402                                    <1>                             ; or another error (the bad')
1403                                    <1>
1404                                    <1>       ; (& the uggly!) truncate file to zero length before open
1405                                    <1>
1406                                    <1>       ;'*' and '?' already checked at 'set_working_path' stage
1407                                    <1>       ;and   dx, dx
1408                                    <1>       ;jnz   short sysmkdir_err ; permission denied
1409                                    <1>                              ; invalid filename chars
1410                                    <1>
1411                                    <1>       ;test  cl, 10h ; subdirectory ?
1412                                    <1>       ;jnz   short sysmkdir_err
1413                                    <1>
1414                                    <1>       ; BL = File Attributes:
1415                                    <1>       ;            bit 0 (1) - Read only file (R)
1416                                    <1>       ;              bit 1 (1) - Hidden file (H)
1417                                    <1>         ;              bit 2 (1) - System file (R)
1418                                    <1>       ;              bit 3 (1) - Volume label/name (V)
1419                                    <1>         ;              bit 4 (1) - Subdirectory (D)
1420                                    <1>       ;            bit 5 (1) - File has been archived
1421                                    <1>
1422                                    <1>       ; * existing directory must not be truncated
1423                                    <1>       ;   (we don't know it is empty or not, at this stage)
1424                                    <1>       ; * existing volume name (or a long name) can not be
1425                                    <1>       ;   re-created or truncated by 'syscreat'
1426                                    <1>       ; * A file with S, H, R attributes must not be truncated
1427                                    <1>       ;   (change attributes to normal, if you need truncate it)
1428                                    <1>
```

```
1429 0000DDA1 F6C31F              <1>       test   bl, 00011111b  ; check attributes of existing file
1430 0000DDA4 754E                <1>       jnz    short sysmkdir_err
1431                              <1>
1432                              <1>       ;; normal file, OK to continue...
1433                              <1>
1434                              <1>       ; ESI = FindFile_DirEntry
1435 0000DDA6 668B4614            <1>       mov    ax, [esi+DirEntry_FstClusHI] ; 20
1436 0000DDAA C1E010              <1>       shl    eax, 16 ; 13/11/2017
1437 0000DDAD 668B461A            <1>       mov    ax, [esi+DirEntry_FstClusLO] ; 26
1438                              <1>       ; EAX = First cluster to be truncated/unlinked
1439 0000DDB1 57                  <1>       push   edi
1440 0000DDB2 51                  <1>       push   ecx
1441 0000DDB3 BE00010900          <1>       mov    esi, Logical_DOSDisks
1442 0000DDB8 29C9                <1>       sub    ecx, ecx
1443 0000DDBA 8A2D[868A0100]      <1>       mov    ch, [Current_Drv]
1444 0000DDC0 01CE                <1>       add    esi, ecx
1445                              <1>       ; ESI = Logical dos drive description table address
1446 0000DDC2 E8C9F7FFFF          <1>       call   truncate_cluster_chain
1447 0000DDC7 59                  <1>       pop    ecx
1448 0000DDC8 5F                  <1>       pop    edi
1449 0000DDC9 7230                <1>       jc     short syscreate_truncate_err
1450                              <1>
1451                              <1>       ; 26/10/2016
1452                              <1>       ; EDI = Directory entry address in directory buffer
1453                              <1>       ; Update directory entry
1454 0000DDCB E848DCFFFF          <1>       call   convert_current_date_time
1455                              <1>       ; OUTPUT -> DX = Date in dos dir entry format
1456                              <1>         ;       AX = Time in dos dir entry format
1457 0000DDD0 66894716            <1>       mov    [edi+DirEntry_WrtTime], ax
1458 0000DDD4 66895718            <1>       mov    [edi+DirEntry_WrtDate], dx
1459 0000DDD8 66895712            <1>       mov    [edi+DirEntry_LastAccDate], dx
1460 0000DDDC 31C0                <1>       xor    eax, eax ; file size = 0
1461 0000DDDE 89471C              <1>       mov    [edi+DirEntry_FileSize], eax ; 0
1462 0000DDE1 C605[AC910100]02    <1>       mov    byte [DirBuff_ValidData], 2 ; data changed sign
1463 0000DDE8 BE[78930100]        <1>       mov    esi, FindFile_DirEntry
1464 0000DDED B201                <1>       mov    dl, 1 ; open file for writing
1465 0000DDEF E9AA000000          <1>       jmp    sysopen_2
1466                              <1>
1467                              <1> sysmkdir_err:
1468                              <1>       ; 1 = write, 2 = read & write, >2 = invalid
1469 0000DDF4 B80B000000          <1>         mov  eax, ERR_FILE_ACCESS  ; 11 ; 'permission denied !'
1470 0000DDF9 EB73                <1>         jmp  short sysopen_err
1471                              <1>
1472                              <1> syscreate_truncate_err:
1473 0000DDFB B812000000          <1>       mov    eax, ERR_DRV_WRITE ; 18 ; 'disk write error !'
1474 0000DE00 EB6C                <1>         jmp  short sysopen_err
1475                              <1>
1476                              <1> syscreat_inv_fname:  ; invalid file name chars
1477                              <1>       ; 16/10/2016
1478 0000DE02 B81A000000          <1>       mov    eax, ERR_INV_FILE_NAME  ; 26 ; invalid file name chars
1479 0000DE07 59                  <1>       pop    ecx
1480 0000DE08 EB64                <1>       jmp    sysopen_err
1481                              <1>
1482                              <1> syscreat_1:
1483                              <1>       ; Error code in EAX
1484 0000DE0A 3C02                <1>         cmp  al, 02h ; 'File not found' error
1485 0000DE0C 7560                <1>         jne  sysopen_err
1486                              <1>
1487 0000DE0E F6C110              <1>       test   cl, 10h ; Directory
1488 0000DE11 0F852C020000        <1>       jnz    sysmkdir_2
1489                              <1>
1490                              <1> syscreat_2:
1491 0000DE17 BE[68930100]        <1>       mov    esi, FindFile_Name
1492                              <1>         ;xor edx, edx
1493 0000DE1C 31C0                <1>         xor  eax, eax ; File Size  = 0
1494 0000DE1E 31DB                <1>       xor    ebx, ebx
1495 0000DE20 4B                  <1>       dec    ebx ; FFFFFFFFh -> create empty file
1496                              <1>                  ;               (only for FAT fs)
1497                              <1>       ; CL = File Attributes
1498 0000DE21 E8F8EBFFFF          <1>       call   create_file
1499 0000DE26 7246                <1>       jc     sysopen_err
1500                              <1>              ; EAX = New file's first cluster
1501                              <1>              ; ESI = Logical Dos Drv Descr. Table Addr.
1502                              <1>              ; EBX = offset CreateFile_Size
1503                              <1>              ; ECX = Sectors per cluster (<256)
1504                              <1>              ; EDX = Directory entry index/number (<65536)
1505                              <1>       ; 26/10/2016
1506                              <1>       ;mov   esi, Directory_Buffer
1507                              <1>       ;shl   dx, 5 ; *32
1508                              <1>       ;add   esi, edx
1509                              <1>       ;; esi = directory entry address in directory buffer
1510                              <1>
1511                              <1>       ; Here, directory entry has been created but last
1512                              <1>       ; modification date & time of the parent dir has not
1513                              <1>       ; been updated, yet!
1514                              <1>       ; (Note: Directory and FAT buffers have been updated...)
1515                              <1>
1516 0000DE28 E824DDFFFF          <1>       call   update_parent_dir_lmdt ; now, it is OK too!
1517                              <1>
1518                              <1>       ; 25/10/2016
1519 0000DE2D 66B80018            <1>       mov    ax, 1800h
1520 0000DE31 BE[68930100]        <1>       mov    esi, FindFile_Name
1521 0000DE36 E846B6FFFF          <1>       call   find_first_file
1522 0000DE3B 7231                <1>       jc     short sysopen_err
1523                              <1>
1524                              <1>       ; Only possible error after here is
1525                              <1>       ; "too many open files !" error.
1526                              <1>       ;
1527                              <1>       ; If "syscreat" will return with that error,
1528                              <1>       ; (the file has been created but it could not be opened)
1529                              <1>       ; the user must retry to open this file again
1530                              <1>       ; or must close another file before using
1531                              <1>       ; "sysopen" system call.
1532                              <1>
1533 0000DE3D B201                <1>       mov    dl, 1 ; open file for writing
```

```
1534                                    <1>       ; ESI = Directory Entry (FindFile_DirEntry) Location
1535                                    <1>       ; EAX = File Size (= 0)
1536 0000DE3F EB5D                     <1>       jmp    short sysopen_2
1537                                    <1>
1538                                    <1> sysopen: ;<open file>
1539                                    <1>       ; 26/10/2016
1540                                    <1>       ; 24/10/2016
1541                                    <1>       ; 17/10/2016
1542                                    <1>       ; 15/10/2016
1543                                    <1>       ; 06/10/2016, 07/10/2016, 08/10/2016
1544                                    <1>       ; 05/10/2016 (TRDOS 386 = TRDOS v2.0)
1545                                    <1>       ;          -derived from INT_21H.ASM-
1546                                    <1>       ;            ("loc_INT21h_open_file")
1547                                    <1>         ;    26/02/2011
1548                                    <1>         ;    INT 21h Function AH = 3Dh
1549                                    <1>         ;    Open File
1550                                    <1>         ;    INPUT
1551                                    <1>         ;       AL= File Access Value
1552                                    <1>       ;           0- Open for reading
1553                                    <1>       ;             1- Open for writing
1554                                    <1>         ;             2- Open for reading and writing
1555                                    <1>         ;          DS:DX= Pointer to filename (ASCIIZ)
1556                                    <1>         ;
1557                                    <1>       ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
1558                                    <1>       ; 22/05/2013 - 27/05/2013 (Retro UNIX 8086 v1)
1559                                    <1>       ;
1560                                    <1>       ; 'sysopen' opens a file in following manner:
1561                                    <1>       ;    1) The second argument in a sysopen says whether to
1562                                    <1>       ;      open the file ro read (0) or write (>0).
1563                                    <1>       ;    2) I-node of the particular file is obtained via 'namei'.
1564                                    <1>       ;    3) The file is opened by 'iopen'.
1565                                    <1>       ;    4) Next housekeeping is performed on the fsp table
1566                                    <1>       ;      and the user's open file list - u.fp.
1567                                    <1>       ;      a) u.fp and fsp are scanned for the next available slot.
1568                                    <1>       ;      b) An entry for the file is created in the fsp table.
1569                                    <1>       ;      c) The number of this entry is put on u.fp list.
1570                                    <1>       ;      d) The file descriptor index to u.fp list is pointed
1571                                    <1>       ;         to by u.r0.
1572                                    <1>       ;
1573                                    <1>       ; Calling sequence:
1574                                    <1>       ;      sysopen; name; mode
1575                                    <1>       ; Arguments:
1576                                    <1>       ;      name - file name or path name
1577                                    <1>       ;      mode - 0 to open for reading
1578                                    <1>       ;             1 to open for writing
1579                                    <1>       ; Inputs: (arguments)
1580                                    <1>       ; Outputs: *u.r0 - index to u.fp list (the file descriptor)
1581                                    <1>       ;              is put into r0's location on the stack.
1582                                    <1>       ; ..........................................................
1583                                    <1>       ;
1584                                    <1>       ; Retro UNIX 8086 v1 modification:
1585                                    <1>       ;       'sysopen' system call has two arguments; so,
1586                                    <1>       ;      * 1st argument, name is pointed to by BX register
1587                                    <1>       ;      * 2nd argument, mode is in CX register
1588                                    <1>       ;
1589                                    <1>       ;      AX register (will be restored via 'u.r0') will return
1590                                    <1>       ;      to the user with the file descriptor/number
1591                                    <1>       ;      (index to u.fp list).
1592                                    <1>       ;
1593                                    <1>       ;call  arg2
1594                                    <1>       ; * name - 'u.namep' points to address of file/path name
1595                                    <1>       ;          in the user's program segment ('u.segmnt')
1596                                    <1>       ;          with offset in BX register (as sysopen argument 1).
1597                                    <1>       ; * mode - sysopen argument 2 is in CX register
1598                                    <1>       ;          which is on top of stack.
1599                                    <1>       ;
1600                                    <1>       ; jsr r0,arg2 / get sys args into u.namep and on stack
1601                                    <1>       ;
1602                                    <1>            ; system call registers: ebx, ecx (through 'sysenter')
1603                                    <1>       ;
1604                                    <1>       ; TRDOS 386 (05/10/2016)
1605                                    <1>       ;
1606                                    <1>         ; INPUT ->
1607                                    <1>         ;       CL = File Access Value (Open Mode)
1608                                    <1>       ;            0 - Open file for reading
1609                                    <1>       ;              1 - Open file for writing
1610                                    <1>         ;              2 - Open device for reading
1611                                    <1>       ;            3 - Open device for writing
1612                                    <1>         ;           EBX = Pointer to filename/devicename (ASCIIZ)
1613                                    <1>       ; OUTPUT ->
1614                                    <1>       ;          eax = File/Device Handle/Number (index) (AL)
1615                                    <1>       ;          cf = 1 -> Error code in AL
1616                                    <1>       ;
1617                                    <1>       ; Modified Registers: EAX (at the return of system call)
1618                                    <1>       ;
1619                                    <1>
1620 0000DE41 80F901                   <1>       cmp    cl, 1 ; read file (0), write file (1)
1621 0000DE44 7614                     <1>       jna    short sysopen_0
1622                                    <1>
1623 0000DE46 80F903                   <1>       cmp    cl, 3
1624 0000DE49 0F8640010000             <1>       jna    sysopen_device
1625                                    <1>
1626                                    <1>       ; Invalid access code
1627 0000DE4F B817000000               <1>       mov    eax, ERR_INV_PARAMETER
1628 0000DE54 0F874B010000             <1>       ja     sysopen_dev_err
1629                                    <1>
1630                                    <1> sysopen_0:
1631                                    <1>       ;mov   [u.namep], ebx
1632 0000DE5A 51                       <1>       push   ecx
1633 0000DE5B 89DE                     <1>       mov    esi, ebx
1634                                    <1>       ; file name is forced, change directory as temporary
1635                                    <1>       ;mov   ax, 1
1636                                    <1>       ;mov   [FFF_Valid], ah ; 0 ; reset ; 17/10/2016
1637                                    <1>       ;call  set_working_path
1638 0000DE5D E80C510000               <1>       call   set_working_path_x ; 17/10/2016
```

```
1639 0000DE62 731E                <1>       jnc    short sysopen_1
1640                              <1>
1641                              <1> syscreat_err: ; ecx = file attributes (for 'syscreat')
1642 0000DE64 59                  <1>       pop    ecx ; open mode
1643 0000DE65 21C0                <1>       and    eax, eax  ; 0 -> Bad Path!
1644 0000DE67 7505                <1>       jnz    short sysopen_err
1645                              <1>       ; eax = 0
1646 0000DE69 B80C000000          <1>       mov    eax, ERR_DIR_NOT_FOUND ; Directory not found !
1647                              <1> sysopen_err:
1648 0000DE6E A3[64030300]        <1>       mov    [u.r0], eax
1649 0000DE73 A3[C8030300]        <1>       mov    [u.error], eax
1650 0000DE78 E8C6510000          <1>       call   reset_working_path
1651 0000DE7D E95AFAFFFF          <1>       jmp    error
1652                              <1>
1653                              <1> sysopen_1:
1654                              <1>       ;mov   esi, FindFile_Name
1655 0000DE82 66B80018            <1>         mov  ax, 1800h ; Only files
1656 0000DE86 E8F6B5FFFF          <1>       call   find_first_file
1657 0000DE8B 5A                  <1>       pop    edx
1658 0000DE8C 72E0                <1>       jc     short sysopen_err ; eax = 2 (File not found !)
1659                              <1>
1660                              <1>       ; check_open_file_attr_access_code
1661                              <1>
1662 0000DE8E F6C307              <1>         test bl, 7  ; system, hidden, readonly
1663 0000DE91 740B                <1>         jz   short sysopen_2
1664                              <1>
1665 0000DE93 20D2                <1>       and    dl, dl ; 0 = read mode
1666 0000DE95 7407                <1>       jz     short sysopen_2
1667                              <1>
1668                              <1>       ; 1 = write, 2 = read & write, >2 = invalid
1669 0000DE97 B80B000000          <1>         mov  eax, ERR_FILE_ACCESS  ; 11 = 'permission denied !'
1670 0000DE9C EBD0                <1>         jmp  short sysopen_err
1671                              <1>
1672                              <1> sysopen_2:
1673                              <1>       ; esi = Directory Entry (FindFile_DirEntry) Location
1674 0000DE9E 89F3                <1>       mov    ebx, esi
1675 0000DEA0 31F6                <1>         xor    esi, esi ; 0
1676 0000DEA2 31FF                <1>         xor    edi, edi ; 0
1677                              <1> sysopen_3: ; scan the list of entries in fsp table
1678 0000DEA4 80BE[6A030300]00    <1>         cmp    byte [esi+u.fp], 0
1679 0000DEAB 760F                <1>         jna    short sysopen_4 ; empty slot
1680 0000DEAD 6646                <1>         inc    si
1681 0000DEAF 6683FE0A            <1>         cmp    si, 10
1682 0000DEB3 72EF                <1>       jb     short sysopen_3
1683                              <1> toomanyf:
1684 0000DEB5 B80D000000          <1>       mov    eax, ERR_TOO_MANY_FILES ; too many open files !
1685 0000DEBA EBB2                <1>       jmp    short sysopen_err
1686                              <1>
1687                              <1> sysopen_4:
1688 0000DEBC 80BF[E6990100]00    <1>         cmp    byte [edi+OF_MODE], 0 ; Scan open files table
1689 0000DEC3 760A                <1>       jna    short sysopen_5
1690 0000DEC5 6647                <1>       inc    di
1691 0000DEC7 6683FF0A            <1>       cmp    di, OPENFILES ; max. number of open files (=10)
1692 0000DECB 72EF                <1>       jb     short sysopen_4
1693 0000DECD EBE6                <1>       jmp    short toomanyf
1694                              <1>
1695                              <1> sysopen_5:
1696 0000DECF FEC2                <1>       inc    dl
1697 0000DED1 8897[E6990100]      <1>         mov    [edi+OF_MODE], dl
1698 0000DED7 8A15[26930100]      <1>       mov    dl, [FindFile_Drv]
1699 0000DEDD 8897[DC990100]      <1>         mov    [edi+OF_DRIVE], dl ; Logical DOS drive number
1700 0000DEE3 66C1E702            <1>       shl    di, 2 ; *4 (dword offset)
1701                              <1>
1702 0000DEE7 8987[2C9A0100]      <1>       mov    [edi+OF_SIZE], eax ; File size in bytes
1703                              <1>
1704 0000DEED 668B4314            <1>         mov  ax, [ebx+DirEntry_FstClusHI]
1705 0000DEF1 C1E010              <1>       shl    eax, 16
1706 0000DEF4 668B431A            <1>       mov    ax, [ebx+DirEntry_FstClusLO]
1707 0000DEF8 8987[B4990100]      <1>       mov    [edi+OF_FCLUSTER], eax ; First cluster
1708 0000DEFE 8987[CC9A0100]      <1>       mov    [edi+OF_CCLUSTER], eax ; Current cluster
1709                              <1>
1710 0000DF04 31DB                <1>         xor  ebx, ebx
1711 0000DF06 899F[049A0100]      <1>         mov    [edi+OF_POINTER], ebx ; offset pointer (0)
1712 0000DF0C 899F[F49A0100]      <1>         mov    [edi+OF_CCINDEX], ebx ; cluster index (0)
1713                              <1>
1714 0000DF12 A1[98930100]        <1>       mov    eax, [FindFile_DirFirstCluster]
1715 0000DF17 8987[549A0100]      <1>       mov    [edi+OF_DIRFCLUSTER], eax
1716                              <1>
1717 0000DF1D A1[9C930100]        <1>       mov    eax, [FindFile_DirCluster]
1718 0000DF22 8987[7C9A0100]      <1>       mov    [edi+OF_DIRCLUSTER], eax
1719                              <1>
1720                              <1>       ; Get (& Save) Volume ID
1721                              <1>       ; Important for files of removable drives
1722                              <1>       ; (In order to check the drive has same volume/disk)
1723 0000DF28 88D7                <1>       mov    bh, dl
1724 0000DF2A 81C300010900        <1>         add  ebx, Logical_DOSDisks
1725 0000DF30 8A4303              <1>         mov  al, [ebx+LD_FATType]
1726 0000DF33 3C01                <1>         cmp  al, 1
1727 0000DF35 7209                <1>         jb   short sysopen_6_fs
1728 0000DF37 3C02                <1>         cmp  al, 2
1729 0000DF39 770A                <1>         ja   short sysopen_6_fat32
1730                              <1> sysopen_6_fat:
1731 0000DF3B 8B432D              <1>         mov  eax, [ebx+LD_BPB+VolumeID]
1732 0000DF3E EB08                <1>         jmp  short sysopen_7
1733                              <1> sysopen_6_fs:
1734 0000DF40 8B4328              <1>         mov  eax, [ebx+LD_FS_VolumeSerial]
1735 0000DF43 EB03                <1>         jmp  short sysopen_7
1736                              <1> sysopen_6_fat32:
1737 0000DF45 8B4349              <1>         mov  eax, [ebx+LD_BPB+FAT32_VolID]
1738                              <1> sysopen_7:
1739 0000DF48 A3[7C8A0100]        <1>         mov  [Current_VolSerial], eax
1740                              <1>
1741 0000DF4D 8987[A49A0100]      <1>       mov    [edi+OF_VOLUMEID], eax
1742                              <1>
1743                              <1>       ; 24/10/2016
```

```
1744 0000DF53 66D1EF              <1>       shr    di, 1 ; 4/2, word offset
1745 0000DF56 668B1D[A0930100]    <1>       mov    bx, [FindFile_DirEntryNumber]
1746 0000DF5D 66899F[1C9B0100]    <1>       mov    [edi+OF_DIRENTRY], bx
1747                              <1>
1748 0000DF64 31D2                <1>       xor    edx, edx
1749                              <1>       ;shr   di, 2 ; /4 (byte offset)
1750 0000DF66 66D1EF              <1>       shr    di, 1 ; 2/2, byte offset
1751 0000DF69 8897[FA990100]      <1>       mov    byte [edi+OF_OPENCOUNT], dl ; 0
1752 0000DF6F 8897[F0990100]      <1>       mov    byte [edi+OF_STATUS], dl ; 0
1753                              <1>
1754 0000DF75 89FB                <1>       mov    ebx, edi
1755 0000DF77 FEC3                <1>       inc    bl
1756                              <1>
1757 0000DF79 889E[6A030300]      <1>         mov     [esi+u.fp], bl ; Open File Entry Number
1758 0000DF7F 8935[64030300]      <1>       mov     [u.r0], esi ; move index to u.fp list
1759                              <1>                     ; into eax on stack
1760                              <1>
1761 0000DF85 E8B9500000          <1>         call       reset_working_path
1762                              <1>
1763 0000DF8A E96DF9FFFF          <1>       jmp    sysret
1764                              <1>
1765                              <1>       ; (Retro UNIX 386 v1.0)
1766                              <1>       ; 'fsp' table (10 bytes/entry)
1767                              <1>       ; bit 15                              bit 0
1768                              <1>       ; ---|----------------------------------------
1769                              <1>       ; r/w|       i-number of open file
1770                              <1>       ; ---|----------------------------------------
1771                              <1>       ;               device number
1772                              <1>       ; ----------------------------------------------
1773                              <1>       ; offset pointer, r/w pointer to file (bit 0-15)
1774                              <1>       ; ----------------------------------------------
1775                              <1>       ; offset pointer, r/w pointer to file (bit 16-31)
1776                              <1>       ; ---------------------|------------------------
1777                              <1>       ;  flag that says file    | number of processes
1778                              <1>       ;   has been deleted      | that have file open
1779                              <1>       ; ---------------------|------------------------
1780                              <1>
1781                              <1> sysopen_device:
1782                              <1>       ; 15/10/2016
1783                              <1>       ; 08/10/2016
1784                              <1>       ; 07/10/2016 (TRDOS 386 = TRDOS v2.0)
1785 0000DF8F 51                  <1>       push   ecx ; open mode
1786 0000DF90 89E5                <1>       mov    ebp, esp
1787 0000DF92 B910000000          <1>       mov    ecx, 16 ; transfer length = 16 bytes
1788 0000DF97 29CC                <1>       sub    esp, ecx
1789 0000DF99 89E7                <1>       mov    edi, esp ; destination address
1790 0000DF9B 89DE                <1>       mov    esi, ebx ; dev name in user's memory space
1791 0000DF9D E89A3B0000          <1>       call   transfer_from_user_buffer
1792 0000DFA2 7310                <1>       jnc    short sysopen_dev_0
1793                              <1>       ; eax = ERR_OUT_OF_MEMORY = 4 = ERR_MINOR_IM
1794 0000DFA4 59                  <1>       pop    ecx
1795                              <1> sysopen_dev_err:
1796 0000DFA5 A3[64030300]        <1>       mov    [u.r0], eax
1797 0000DFAA A3[C8030300]        <1>       mov    [u.error], eax
1798 0000DFAF E928F9FFFF          <1>       jmp    error
1799                              <1> sysopen_dev_0:
1800 0000DFB4 89FE                <1>       mov    esi, edi ; Device name addr (max. 16 bytes, ASCIIZ)
1801                              <1>                       ; for example: "tty, TTY, /dev/tty"
1802 0000DFB6 E82E530000          <1>       call   get_device_number
1803 0000DFBB 89EC                <1>       mov    esp, ebp
1804 0000DFBD 59                  <1>       pop    ecx
1805 0000DFBE 7307                <1>       jnc    short sysopen_dev_1
1806 0000DFC0 B818000000          <1>       mov    eax, ERR_INV_DEV_NAME ; 24 ; 'invalid device name !'
1807 0000DFC5 EBDE                <1>       jmp    short sysopen_dev_err
1808                              <1> sysopen_dev_1:
1809                              <1>       ; eax = Device Number (AL)
1810                              <1>       ;  cl = Open mode (2 = device read, 3 = device write)
1811 0000DFC7 31DB                <1>         xor     ebx, ebx ; 0
1812                              <1> sysopen_dev_2: ; scan the list of entries
1813 0000DFC9 389B[6A030300]      <1>         cmp     [ebx+u.fp], bl ; 0
1814 0000DFCF 760E                <1>         jna     short sysopen_dev_3 ; empty slot
1815 0000DFD1 FEC3                <1>         inc     bl
1816 0000DFD3 80FB0A              <1>         cmp     bl, 10
1817 0000DFD6 72F1                <1>       jb     short sysopen_dev_2
1818                              <1>       ;
1819 0000DFD8 B80D000000          <1>       mov    eax, ERR_TOO_MANY_FILES ; too many open files !
1820 0000DFDD EBC6                <1>       jmp    short sysopen_dev_err
1821                              <1> sysopen_dev_3:
1822 0000DFDF 891D[64030300]      <1>       mov    [u.r0], ebx ; File/Device index/handle/descriptor
1823                              <1>       ; eax = device number (entry offset)
1824 0000DFE5 8AA8[78970100]      <1>       mov    ch, [eax+DEV_ACCESS] ; bit 0 = accessable by users
1825                              <1>                                   ; bit 1 = read access perm
1826                              <1>                                   ; bit 2 = write access perm
1827                              <1>                                   ; bit 3 = IOCTL permit to users
1828                              <1>                                   ; bit 4 = block device if set
1829                              <1>                                   ; bit 5 = 16 bit or 1024 byte
1830                              <1>                                   ; bit 6 = 32 bit or 2048 byte
1831                              <1>                                   ; bit 7 = installable device drv
1832 0000DFEB F6C501              <1>       test   ch, 1 ; accessable by normal users (except root)
1833 0000DFEE 7510                <1>       jnz    short sysopen_dev_4 ; yes, permission has been given
1834 0000DFF0 803D[B0030300]00    <1>       cmp    byte [u.uid], 0 ; root?
1835 0000DFF7 7607                <1>       jna    short sysopen_dev_4 ; superuser can open all devices
1836                              <1> sysopen_dev_perm_err:
1837 0000DFF9 B80B000000          <1>       mov    eax, ERR_DEV_ACCESS  ; 11 = 'permission denied !'
1838 0000DFFE EBA5                <1>       jmp    short sysopen_dev_err
1839                              <1> sysopen_dev_4:
1840 0000E000 D0ED                <1>       shr    ch, 1 ; result: 1 = read, 2 = write, 3 = r & w
1841 0000E002 FEC9                <1>       dec    cl  ; result: 1 = read, 2 = write
1842 0000E004 84E9                <1>       test   cl, ch
1843 0000E006 74F1                <1>       jz     short sysopen_dev_perm_err
1844                              <1>
1845 0000E008 D0E5                <1>       shl    ch, 1 ; bit 0 = 0
1846                              <1>       ; eax = device number (entry offset)
1847 0000E00A E8F6530000          <1>       call   device_open
1848 0000E00F 72E8                <1>       jc     short sysopen_dev_perm_err
```

```
                                <1>
                                <1>       ; eax = device number (entry offset)
0000E011 0C80                   <1>       or    al, 80h ; set device bit (set bit 7 to 1)
0000E013 8B1D[64030300]         <1>       mov   ebx, [u.r0]
0000E019 8883[6A030300]         <1>       mov   [ebx+u.fp], al      ; bit 7 (=1) points to device
                                <1>
0000E01F 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>
0000E024 6621C9                 <1>       and   cx, cx ; if cx = 0 -> create a normal subdir
0000E027 7413                   <1>       jz    short sysmkdir_1
                                <1>
0000E029 F6C110                 <1>       test  cl, 10h ; if dir flags set, also use other flags
0000E02C 0F853EFDFFFF           <1>       jnz   sysmkdir_0 ; jump to head of 'syscreat'
                                <1>
                                <1>       ; CX has wrong flags
0000E032 B817000000             <1>       mov   eax, ERR_INV_FLAGS
0000E037 E969FFFFFF             <1>       jmp   sysopen_dev_err
                                <1>
                                <1> sysmkdir_1:
0000E03C B110                   <1>       mov   cl, 10h ; set subdir flag and reset other flags
0000E03E 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
0000E043 BE[68930100]           <1>       mov   esi, FindFile_Name
0000E048 E804D7FFFF             <1>       call  make_sub_directory
0000E04D 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>
0000E053 A3[64030300]           <1>       mov   [u.r0], eax ; New sub dir's first cluster
                                <1>
0000E058 E8E64F0000             <1>         call      reset_working_path
                                <1>
0000E05D 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>
0000E062 89D8                   <1>       mov   eax, ebx
0000E064 31DB                   <1>       xor   ebx, ebx
0000E066 891D[64030300]         <1>       mov   [u.r0], ebx ; 0  ; return value of EAX
0000E06C E85A2F0000             <1>       call  fclose
0000E071 0F8385F8FFFF           <1>       jnc   sysret
0000E077 B80A000000             <1>       mov   eax, ERR_FILE_NOT_OPEN ; file not open !
0000E07C A3[C8030300]           <1>       mov   [u.error], eax ;
0000E081 A3[64030300]           <1>       mov   [u.r0], eax ; ! invalid handle !
0000E086 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
```

```
2059 0000E08B E8892F0000          <1>       call   getf1
2060 0000E090 7277                <1>       jc     short device_read ; read data from device
2061                              <1>       ; EAX = First cluster of the file
2062                              <1>
2063 0000E092 E83F000000          <1>       call   rw1
2064 0000E097 730A                <1>       jnc    short sysread_0
2065                              <1>
2066 0000E099 A3[64030300]        <1>       mov    [u.r0], eax ; error code
2067 0000E09E E939F8FFFF          <1>       jmp    error
2068                              <1>
2069                              <1> sysread_0:
2070 0000E0A3 E880350000          <1>       call   readi
2071 0000E0A8 EB1D                <1>       jmp    short rw0
2072                              <1>
2073                              <1> syswrite: ; < write to file >
2074                              <1>       ; 23/10/2016
2075                              <1>       ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
2076                              <1>       ;          -derived from INT_21H.ASM-
2077                              <1>       ;            ("loc_INT21h_write_file")
2078                              <1>         ;    13/03/2011 (05/03/2011)
2079                              <1>         ;    INT 21h Function AH = 40h
2080                              <1>         ;    Write to a File
2081                              <1>         ;    INPUT
2082                              <1>       ;         BX = File Handle
2083                              <1>         ;       CX = Number of bytes to write
2084                              <1>         ;          DS:DX= Buffer address
2085                              <1>         ;
2086                              <1>       ; Note: TRDOS 386 'sysrwrite' has been derived from
2087                              <1>       ;     Retro UNIX 386 v1 'syswrite', except a few
2088                              <1>       ;     code modifications.
2089                              <1>       ;
2090                              <1>
2091                              <1>       ; 13/05/2015 (Retro UNIX 386 v1)
2092                              <1>       ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
2093                              <1>       ; 23/05/2013 (Retro UNIX 8086 v1)
2094                              <1>       ;
2095                              <1>       ; 'syswrite' is given a buffer to write onto an output file
2096                              <1>       ; and the number of characters to write. If finds the file
2097                              <1>       ; from the file descriptor located in *u.r0 (r0). This file
2098                              <1>       ; descriptor is returned from a successful open or create call
2099                              <1>       ; (sysopen or syscreat). The i-number of file is obtained via
2100                              <1>       ; 'rw1' and buffer is written on the output file via 'write'.
2101                              <1>       ;
2102                              <1>       ; Calling sequence:
2103                              <1>       ;     syswrite; buffer; nchars
2104                              <1>       ; Arguments:
2105                              <1>       ;     buffer - location of contiguous bytes to be writtten.
2106                              <1>       ;     nchars - number of characters to be written.
2107                              <1>       ; Inputs: *u.r0 - file descriptor (& arguments)
2108                              <1>       ; Outputs: *u.r0 - number of bytes written.
2109                              <1>       ; ..........................................................
2110                              <1>       ;
2111                              <1>       ; Retro UNIX 8086 v1 modification:
2112                              <1>       ;       'syswrite' system call has three arguments; so,
2113                              <1>       ;     * 1st argument, file descriptor is in BX register
2114                              <1>       ;     * 2nd argument, buffer address/offset in CX register
2115                              <1>       ;     * 3rd argument, number of bytes is in DX register
2116                              <1>       ;
2117                              <1>       ;     AX register (will be restored via 'u.r0') will return
2118                              <1>       ;     to the user with number of bytes written.
2119                              <1>       ;
2120                              <1>       ; INPUT ->
2121                              <1>         ;      EBX = File handle (descriptor/index)
2122                              <1>       ;        ECX = Buffer address
2123                              <1>         ;          EDX = Number of bytes
2124                              <1>       ; OUTPUT ->
2125                              <1>       ;         EAX = Number of bytes have been written
2126                              <1>       ;         cf = 1 -> Error code in AL
2127                              <1>       ;
2128                              <1>       ; Modified Registers: EAX (at the return of system call)
2129                              <1>       ;
2130                              <1>
2131                              <1>       ; EBX = File descriptor
2132 0000E0AA E86A2F0000          <1>       call   getf1
2133 0000E0AF 7274                <1>       jc     short device_write ; write data to device
2134                              <1>       ; EAX = First cluster of the file
2135                              <1>       ; EBX = File number  (Open file number) ; 23/10/2016
2136                              <1>
2137 0000E0B1 E820000000          <1>       call   rw1
2138 0000E0B6 730A                <1>       jnc    short syswrite_0
2139 0000E0B8 A3[64030300]        <1>       mov    [u.r0], eax ; error code
2140 0000E0BD E91AF8FFFF          <1>       jmp    error
2141                              <1>
2142                              <1> syswrite_0:
2143 0000E0C2 E88D3C0000          <1>       call   writei
2144                              <1> rw0: ; 1:
2145 0000E0C7 A1[8C030300]        <1>         mov  eax, [u.nread]
2146 0000E0CC A3[64030300]        <1>       mov    [u.r0], eax
2147 0000E0D1 E926F8FFFF          <1>       jmp    sysret
2148                              <1>
2149                              <1> rw1:
2150                              <1>       ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
2151                              <1>       ; 14/05/2015 (Retro UNIX 386 v1)
2152                              <1>       ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
2153                              <1>       ; 23/05/2013 - 24/05/2013 (Retro UNIX 8086 v1)
2154                              <1>       ; System call registers: ebx, ecx, edx (through 'sysenter')
2155                              <1>       ;
2156                              <1>       ; EBX = File descriptor
2157                              <1>       ;call  getf1 ; calling point in 'getf' from 'rw1'
2158                              <1>       ;jc    short device_rw ; read/write data from/to device
2159                              <1>       ; EAX = First cluster of the file
2160                              <1>
2161 0000E0D6 83F802              <1>       cmp    eax, 2
2162 0000E0D9 7217                <1>       jb     short rw2
2163                              <1>       ;
```

```
2164 0000E0DB 890D[84030300]      <1>       mov    [u.base], ecx       ; buffer address/offset
2165                              <1>                                  ;(in the user's virtual memory space)
2166 0000E0E1 8915[88030300]      <1>       mov    [u.count], edx
2167                              <1>
2168 0000E0E7 C705[C8030300]0000- <1>         mov     dword [u.error], 0 ; reset the last error code
2168 0000E0EF 0000                <1>
2169 0000E0F1 C3                  <1>       retn
2170                              <1>
2171                              <1> rw2:
2172 0000E0F2 B80A000000          <1>       mov    eax, ERR_FILE_NOT_OPEN ; file not open !
2173 0000E0F7 A3[C8030300]        <1>       mov    dword [u.error], eax
2174 0000E0FC C3                  <1>       retn
2175                              <1> rw3:
2176 0000E0FD B80B000000          <1>       mov    eax, ERR_FILE_ACCESS ; permission denied !
2177 0000E102 A3[C8030300]        <1>       mov    dword [u.error], eax
2178 0000E107 F9                  <1>       stc
2179 0000E108 C3                  <1>       retn
2180                              <1>
2181                              <1> device_read:
2182                              <1>       ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
2183                              <1>       ; cl = DEV_OPENMODE ; open mode
2184                              <1>       ; ch = DEV_ACCESS   ; access flags
2185                              <1>       ; al = DEV_DRIVER   ; device number (eax)
2186                              <1>
2187 0000E109 F6C101              <1>       test   cl, 1 ; 1 = read, 2 = write, 3 = read&write
2188 0000E10C 74EF                <1>       jz     short rw3
2189                              <1>
2190 0000E10E 89C3                <1>       mov    ebx, eax
2191 0000E110 66C1E302            <1>       shl    bx, 2 ; *4
2192                              <1>
2193 0000E114 F6C580              <1>       test   ch, 80h ; bit 7, installable device driver flag
2194 0000E117 7406                <1>       jz     short d_read_2 ; Kernel device
2195                              <1>       ; installable device
2196                              <1> d_read_1:
2197 0000E119 FFA3[34970100]      <1>         jmp  dword [ebx+IDEV_RADDR-4]
2198                              <1> d_read_2:
2199 0000E11F FFA3[CE480100]      <1>       jmp    dword [ebx+KDEV_RADDR-4]
2200                              <1>
2201                              <1> device_write:
2202                              <1>       ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
2203                              <1>       ; cl = DEV_OPENMODE ; open mode
2204                              <1>       ; ch = DEV_ACCESS   ; access flags
2205                              <1>       ; al = DEV_DRIVER   ; device number (eax)
2206                              <1>
2207 0000E125 F6C102              <1>       test   cl, 2 ; 1 = read, 2 = write, 3 = read&write
2208 0000E128 74D3                <1>       jz     short rw3
2209                              <1>
2210 0000E12A 89C3                <1>       mov    ebx, eax
2211 0000E12C 66C1E302            <1>       shl    bx, 2 ; *4
2212                              <1>
2213 0000E130 F6C580              <1>       test   ch, 80h ; bit 7, installable device driver flag
2214 0000E133 7406                <1>       jz     short d_write_2 ; Kernel device
2215                              <1>       ; installable device
2216                              <1> d_write_1:
2217 0000E135 FFA3[54970100]      <1>         jmp  dword [ebx+IDEV_WADDR-4]
2218                              <1> d_write_2:
2219 0000E13B FFA3[1E490100]      <1>       jmp    dword [ebx+KDEV_WADDR-4]
2220                              <1>
2221                              <1>
2222                              <1> sysemt: ; enable (or disable) multi tasking -time sharing-
2223                              <1>       ;
2224                              <1>       ; 23/05/2016 - TRDOS 386 (TRDOS v2.0)
2225                              <1>       ; 14/05/2015 (Retro UNIX 386 v1)
2226                              <1>       ; 10/12/2013 - 20/04/2014 (Retro UNIX 8086 v1)
2227                              <1>       ;
2228                              <1>       ; Retro UNIX 8086 v1 modification:
2229                              <1>       ;     'Enable Multi Tasking'  system call instead
2230                              <1>       ;     of 'Emulator Trap' in original UNIX v1 for PDP-11.
2231                              <1>       ;
2232                              <1>       ; Retro UNIX 8086 v1 feature only!
2233                              <1>       ;     Using purpose: Kernel will start without time-out
2234                              <1>       ;     (internal clock/timer) functionality.
2235                              <1>       ;     Then etc/init will enable clock/timer for
2236                              <1>       ;     multi tasking.
2237                              <1>       ;
2238                              <1>       ; INPUT ->
2239                              <1>       ;     BL = 0 -> disable multi tasking
2240                              <1>       ;     BL > 1 -> enable multi tasking (time sharing)
2241                              <1>       ; OUTPUT ->
2242                              <1>       ;     none
2243                              <1>       ;
2244                              <1>       ;  Note: Multi tasking is disabled during system
2245                              <1>       ;       initialization, it must be enabled by using
2246                              <1>       ;       this system call. (Otherwise, running proces
2247                              <1>       ;       will not be changed by another process within
2248                              <1>       ;       run time sequence/schedule, if running process
2249                              <1>       ;       will not 'release' itself. Only 'wakeup' procedure
2250                              <1>       ;       for waiting processes and programmed timer events
2251                              <1>       ;       for other processes can change running process
2252                              <1>       ;       while multi tasking is disabled.) ** 23/05/2016 **
2253                              <1>
2254 0000E141 803D[B0030300]00    <1>       cmp    byte [u.uid], 0 ; root ?
2255                              <1>       ;ja    error
2256 0000E148 0F87D3F8FFFF        <1>       ja     badsys ; 14/05/2015
2257                              <1>       ;
2258 0000E14E FA                  <1>       cli
2259 0000E14F 881D[52960100]      <1>       mov    [multi_tasking], bl ; 0 to disable, >0 to enable
2260 0000E155 E9A2F7FFFF          <1>       jmp    sysret
2261                              <1>
2262                              <1> systimer:
2263                              <1>       ; 02/01/2017
2264                              <1>       ; 21/12/2016
2265                              <1>       ; 19/12/2016
2266                              <1>       ; 10/12/2016 (callback)
2267                              <1>       ; 10/06/2016
```

```
2268                                    <1>       ; 07/06/2016
2269                                    <1>       ; 06/06/2016
2270                                    <1>       ; 21/05/2016
2271                                    <1>       ; 19/05/2016
2272                                    <1>       ; 18/05/2016 - TRDOS 386 (TRDOS v2.0)
2273                                    <1>       ; (TRDOS 386 feature only!)
2274                                    <1>       ;
2275                                    <1>       ; (start or stop timer event(s))
2276                                    <1>       ;
2277                                    <1>       ; INPUT ->
2278                                    <1>       ;     BL = Signal return byte (response byte)
2279                                    <1>       ;          (Any requested value between 0 and 255)
2280                                    <1>       ;          (Kernel will put it at the requested address)
2281                                    <1>       ;     BH = Time count unit
2282                                    <1>       ;          0 = Stop timer event
2283                                    <1>       ;          1 = 18.2 ticks per second
2284                                    <1>       ;          2 = 10 milliseconds
2285                                    <1>       ;          3 = 1 second (for real time clock interrupt)
2286                                    <1>       ;          4 = time/tick count in current time count unit
2287                                    <1>       ;         // 10/12/2016
2288                                    <1>       ;         80h = Stop timer event (callback method)
2289                                    <1>       ;         81h = 18.2 ticks per second, callback method
2290                                    <1>       ;         82h = 10 milliseconds, callback method
2291                                    <1>       ;         83h = 1 second (for RTC int), callback method
2292                                    <1>       ;         84h = current time count unit, callback method
2293                                    <1>       ;
2294                                    <1>       ;         Note: Only 03h or 83h will set real time clock
2295                                    <1>       ;              (RTC) events (Others are for PIT events)!
2296                                    <1>       ;
2297                                    <1>       ;     NOTE: If callback (user service) method is used,
2298                                    <1>       ;         EDX will point to the return address (of service
2299                                    <1>       ;         procedure) in user's space instead of signal
2300                                    <1>       ;         response byte address. (TRDOS 386 kernel will
2301                                    <1>       ;         direct the cpu to that address -in user's space-
2302                                    <1>       ;         at the return of system call or interrupt
2303                                    <1>       ;         just after the adjusted count/time is elapsed.)
2304                                    <1>       ;         User's sevice routine must be ended with a
2305                                    <1>       ;         'iret'. Normal return addresses from system
2306                                    <1>       ;         calls or and interrupts will be kept same except
2307                                    <1>       ;         the timer returns.
2308                                    <1>       ;
2309                                    <1>       ;     BH = 0 -> Stop timer event
2310                                    <1>       ;     BL = Timer event number (1 to 255) if BH = 0
2311                                    <1>       ;          If BL = 0, all timer events (which are belongs
2312                                    <1>       ;           to running process) will be stopped
2313                                    <1>       ;     ECX = Time/Tick count (depending on time count unit)
2314                                    <1>       ;     EDX = Signal return (Response) byte address
2315                                    <1>       ;           (virtual address in user's memory space)
2316                                    <1>       ; OUTPUT ->
2317                                    <1>       ;     AL = Timer event number   (1 to 255) (max. value = 16)
2318                                    <1>       ;     IF BH Input = 0 & CF = 0 & AL = 0 ->
2319                                    <1>       ;          timer event(s) has/have been stopped/finished
2320                                    <1>       ;     CF = 1 & AL = 0 -> no timer setting space to set
2321                                    <1>       ;     CF = 1 & AL > 0 -> timer count unit is not usable
2322                                    <1>       ;
2323                                    <1>       ;     NOTE: To modify a time count for a user function,
2324                                    <1>       ;           at first, current timer event must be stopped
2325                                    <1>       ;           then a new timer event (which is related with
2326                                    <1>       ;           same user function) must be started.
2327                                    <1>       ;
2328                                    <1>       ;           Signal return (response) byte may be used for
2329                                    <1>       ;           several purposes. Kernel will put this value
2330                                    <1>       ;           to requested address during timer interrupt,
2331                                    <1>       ;           program/user can check this value to understand
2332                                    <1>       ;           which event has been occurred and what is changed.
2333                                    <1>       ;           (Multi timer events can share same signal address)
2334                                    <1>       ;
2335                                    <1>       ;     NOTE: If the process is running while the time count
2336                                    <1>       ;           is reached, kernel will put signal return (response)
2337                                    <1>       ;           byte value at requested address during timer
2338                                    <1>       ;           interrupt and the process will continue to run.
2339                                    <1>       ;           Program/process must call (jump to) it's timer event
2340                                    <1>       ;           function as required, for checking the timer event
2341                                    <1>       ;           status via signal return (response) byte address.
2342                                    <1>       ;
2343                                    <1>       ;           If the process is not running (waiting or sleeping
2344                                    <1>       ;           or released) while the time count is reached,
2345                                    <1>       ;           it is restarted from where it left, to ensure
2346                                    <1>       ;           proper multi media (video, audio, clock, timer)
2347                                    <1>       ;           functionality.
2348                                    <1>       ;
2349                                    <1>       ;           (It is better to use 'syswait' or 'syssleep',
2350                                    <1>       ;           or 'sysrele' system call just after the timer
2351                                    <1>       ;           function. Otherwise, timer events may block other
2352                                    <1>       ;           processes which are not using timer events.)

2353                                    <1>       ;
2354                                    <1>       ; Timer Event Structure: (max. 16 timer events, 16*16 bytes)
2355                                    <1>       ;       Owner:              resb 1 ; 0 = free
2356                                    <1>       ;                          ;>0 = process number (u.uno)
2357                                    <1>       ;     Calback:     resb 1 ; 1 = callback, 0 = response byte
2358                                    <1>       ;     Interrupt:      resb 1 ; 0 = Timer interrupt (or none)
2359                                    <1>       ;                          ; 1 = Real Time Clock interrupt
2360                                    <1>       ;     Response:       resb 1 ; 0 to 255, signal return value
2361                                    <1>       ;     Count Limit: resd 1 ; count of ticks (total/set)
2362                                    <1>       ;     Current Count:      resd 1 ; count of ticks (current)
2363                                    <1>       ;     Response Addr:  resd 1 ; response byte (pointer) address
2364                                    <1>       ;
2365                                    <1>
2366                                    <1>       ; 19/12/2016 (timer callback)
2367 0000E15A C605[909B0100]00         <1>       mov   byte [tcallback], 0
2368 0000E161 C605[919B0100]00         <1>       mov   byte [trtc], 0
2369 0000E168 C705[D0030300]0000-      <1>       mov   dword [u.tcb], 0 ; this is not necessary...
2369 0000E170 0000                     <1>
2370                                    <1>
```

```
2371 0000E172 80FF80              <1>       cmp    bh, 80h
2372 0000E175 7225                <1>       jb     short systimer_cb2
2373 0000E177 7704                <1>       ja     short systimer_cb0
2374                              <1>
2375 0000E179 31D2                <1>       xor    edx, edx ; 0, reset callback address
2376 0000E17B EB0B                <1>       jmp    short systimer_cb1
2377                              <1>
2378                              <1> systimer_cb0:
2379 0000E17D 80FF84              <1>       cmp    bh, 84h
2380 0000E180 7764                <1>       ja     short systimer_5 ;  undefined, error
2381                              <1>
2382                              <1>       ;mov   byte [tcallback], 1 ; 19/12/2016
2383 0000E182 FE05[909B0100]      <1>       inc    byte [tcallback]
2384                              <1>
2385                              <1> systimer_cb1:
2386 0000E188 0FB635[B3030300]    <1>       movzx  esi, byte [u.uno] ; process number
2387 0000E18F 66C1E602            <1>       shl    si, 2
2388 0000E193 8996[0C010300]      <1>       mov    [esi+p.tcb-4], edx ; set process timer callback address
2389                              <1>                                 ; (overwrite prev value if it is set!)
2390 0000E199 80E77F              <1>       and    bh, 7Fh
2391                              <1>
2392                              <1> systimer_cb2:
2393 0000E19C 80FF02              <1>       cmp    bh, 2
2394 0000E19F 7445                <1>         je       short systimer_5  ; only 18.2 ticks per second is usable
2395                              <1>                                    ; 10 milliseconds (100 Hertz) timer
2396                              <1>                                    ; will be set later (18/05/2016)
2397 0000E1A1 774B                <1>         ja       short systimer_6
2398                              <1>
2399 0000E1A3 20FF                <1>       and    bh, bh
2400 0000E1A5 0F84BA000000        <1>         jz       systimer_9        ; stop timer event(s)
2401                              <1>
2402                              <1>       ; bh = 1 (timer interrupt, 18.2 Hz, IBM PC/AT ROMBIOS default)
2403                              <1>
2404                              <1> systimer_19:
2405 0000E1AB B00A                <1>       mov    al, 10 ; (*)
2406                              <1>
2407                              <1> systimer_0:
2408 0000E1AD B710                <1>       mov    bh, 16
2409                              <1>       ;
2410 0000E1AF 383D[53960100]      <1>       cmp    [timer_events], bh ; 16 ; 07/06/2016
2411 0000E1B5 7319                <1>       jnb    short systimer_3  ; max. 16 timer events
2412                              <1>       ;
2413 0000E1B7 50                  <1>       push   eax ; (*)
2414                              <1>
2415 0000E1B8 BF[60040300]        <1>       mov    edi, timer_set  ; beginning address of timer events
2416                              <1>                               ; setting space
2417 0000E1BD 30C0                <1>       xor    al, al ; 0
2418                              <1> systimer_1:
2419 0000E1BF FEC0                <1>       inc    al
2420 0000E1C1 803F00              <1>       cmp    byte [edi], 0       ; is it free space ?
2421 0000E1C4 7639                <1>       jna    short systimer_7 ; yes
2422 0000E1C6 FECF                <1>       dec    bh
2423 0000E1C8 7405                <1>       jz     short systimer_2
2424 0000E1CA 83C710              <1>       add    edi, 16
2425 0000E1CD EBF0                <1>       jmp    short  systimer_1 ; next event space
2426                              <1>
2427                              <1> systimer_2:
2428 0000E1CF 58                  <1>       pop    eax ; (*) discard
2429                              <1> systimer_3:
2430 0000E1D0 C605[64030300]00    <1>       mov    byte [u.r0], 0
2431                              <1> systimer_4:
2432 0000E1D7 C705[C8030300]1B00- <1>         mov     dword [u.error], ERR_MISC
2432 0000E1DF 0000                <1>
2433                              <1>                                    ; one of miscellaneous/other errors
2434 0000E1E1 E9F6F6FFFF          <1>       jmp    error ; cf -> 1
2435                              <1>
2436                              <1> systimer_5:
2437 0000E1E6 883D[64030300]      <1>       mov    [u.r0], bh ; Time count unit (=2 or >3)
2438 0000E1EC EBE9                <1>       jmp    short systimer_4 ; 07/06/2016
2439                              <1>
2440                              <1> systimer_6:
2441 0000E1EE 80FF04              <1>       cmp    bh, 4
2442 0000E1F1 77F3                <1>         ja       short systimer_5  ; undefined time count unit
2443                              <1>       ;jb    short systimer_16
2444                              <1>
2445                              <1>       ;mov   al, 1  ; default (use current timer unit)
2446                              <1>                     ; countdown value is in ECX !
2447                              <1>                     ; max. value of ecx = 4294967296/10
2448                              <1>         ;jmp     short systimer_0
2449                              <1>       ;jmp   short systimer_19
2450 0000E1F3 74B6                <1>       je     short systimer_19
2451                              <1>
2452                              <1> systimer_16:
2453                              <1>       ; bh = 3
2454                              <1>       ; timer event via real time clock interrupt
2455                              <1>       ; interrupt/update frequency: 1 Hz (1 tick per second)
2456                              <1>
2457 0000E1F5 B0B6                <1>       mov    al, 182 ; (*) ; 18.2 * 10
2458 0000E1F7 FE05[919B0100]      <1>       inc    byte [trtc] ; timer event via real time clock
2459 0000E1FD EBAE                <1>         jmp      short systimer_0
2460                              <1>
2461                              <1> systimer_7:
2462 0000E1FF A2[64030300]        <1>       mov    [u.r0], al ; timer event number
2463                              <1>       ;
2464                              <1>       ; edi = address of empty timer event area
2465 0000E204 A0[B3030300]        <1>       mov    al, [u.uno]
2466 0000E209 FA                  <1>       cli    ; disable interrupts
2467 0000E20A AA                  <1>       stosb  ; process number
2468 0000E20B A0[909B0100]        <1>       mov    al, [tcallback] ; timer callback flag
2469 0000E210 AA                  <1>       stosb  ; 1= callback method, 0= signal response byte method
2470 0000E211 A0[919B0100]        <1>       mov    al, [trtc] ; timer interrupt type
2471 0000E216 AA                  <1>       stosb  ; 1= real time clock, 0= programmable interval timer
2472 0000E217 88D8                <1>       mov    al, bl ; Signal return (Response) value
2473 0000E219 AA                  <1>       stosb   ; response byte
2474 0000E21A 58                  <1>       pop    eax ; (*) ; 10 or 182
```

```
2475 0000E21B 89D3              <1>       mov    ebx, edx ; virtual address for response/signal byte
2476 0000E21D F7E1              <1>       mul    ecx
2477                            <1>       ; (eax = 10 * count of 18.2 Hz timer ticks)
2478                            <1>       ; (count down step = 10)
2479 0000E21F AB                <1>       stosd  ; count limit (reset value)
2480 0000E220 AB                <1>       stosd  ; current count value
2481                            <1>
2482                            <1>       ; 19/12/2016
2483 0000E221 803D[909B0100]00  <1>       cmp    byte [tcallback], 0 ; timer callback method ?
2484 0000E228 7604              <1>       jna    short systimer_17 ; no
2485 0000E22A 89D8              <1>       mov    eax, ebx ; virtual address for callback routine
2486 0000E22C EB0D              <1>       jmp    short systimer_18
2487                            <1>
2488                            <1> systimer_17: ; signal response byte method
2489                            <1>       ; ebx = virtual address
2490                            <1>       ; [u.pgdir] = page directory's physical address
2491                            <1>       ; 20/02/2017
2492 0000E22E FE05[929B0100]    <1>       inc     byte [no_page_swap] ; 1
2493                            <1>                  ; Do not add this page to swap queue
2494                            <1>                  ; and remove it from swap queue if it is
2495                            <1>                  ; on the queue.
2496 0000E234 E8AF80FFFF        <1>       call   get_physical_addr
2497 0000E239 721A              <1>       jc     short systimer_8 ; 07/06/2016
2498                            <1>       ; eax = physical address of the virtual address in user's space
2499                            <1> systimer_18:
2500 0000E23B AB                <1>       stosd  ; response addr (physical) or callback addr (virtual)
2501 0000E23C FE05[53960100]    <1>       inc    byte [timer_events] ; 07/06/201
2502                            <1>       ; 02/01/2017
2503 0000E242 0FB605[B3030300]  <1>       movzx  eax, byte [u.uno]
2504 0000E249 FE80[FF000300]    <1>       inc    byte [eax+p.timer-1]
2505                            <1>       ;
2506 0000E24F FB                <1>       sti    ; enable interrupts
2507 0000E250 E9A7F6FFFF        <1>       jmp    sysret
2508                            <1>
2509                            <1> systimer_8:
2510                            <1>       ; 10/06/2016
2511                            <1>       ; 07/06/2016
2512 0000E255 28C0              <1>       sub    al, al ; 0
2513 0000E257 8847F4            <1>       mov    [edi-12], al ; clear process number (free timer event)
2514                            <1>       ;mov   dword [edi], eax ; 0
2515 0000E25A FB                <1>       sti
2516 0000E25B A2[64030300]      <1>       mov    [u.r0], al ; 0
2517 0000E260 E977F6FFFF        <1>       jmp    error
2518                            <1>
2519                            <1> systimer_9:
2520                            <1>       ; 10/06/2016
2521                            <1>       ; 07/06/2016
2522 0000E265 28C0              <1>       sub    al, al
2523 0000E267 A2[64030300]      <1>       mov    byte [u.r0], al ; 0
2524 0000E26C 3805[53960100]    <1>       cmp     byte [timer_events], al ;  0
2525 0000E272 7631              <1>       jna    short systimer_12
2526                            <1>
2527                            <1>       ; Note: ecx and edx are undefined here
2528                            <1>       ;      (for stop timer function)
2529                            <1>
2530 0000E274 BE[60040300]      <1>       mov    esi, timer_set  ; beginning address of timer events
2531                            <1>                             ; setting space
2532 0000E279 A0[B3030300]      <1>       mov    al, [u.uno]
2533                            <1>
2534 0000E27E B710              <1>       mov    bh, 16
2535                            <1>
2536 0000E280 08DB              <1>       or     bl, bl
2537 0000E282 7544              <1>       jnz    short systimer_15
2538                            <1>
2539                            <1>       ; clear timer event areas belong to current process
2540                            <1>       ; (for stopping all timer events belong to current process)
2541 0000E284 FA                <1>       cli    ; disable interrupts
2542                            <1> systimer_10:
2543                            <1>       ; 10/06/2016
2544                            <1>       ; 07/06/2016
2545 0000E285 8A26              <1>       mov    ah, [esi]
2546 0000E287 08E4              <1>       or     ah, ah ; 0 ?
2547 0000E289 7411              <1>       jz     short systimer_11
2548 0000E28B 38C4              <1>       cmp    ah, al ; is the process number (owner) same ?
2549 0000E28D 750D              <1>         jne     short systimer_11 ; no
2550                            <1>
2551                            <1>       ;mov   byte [esi], 0
2552 0000E28F 66C7060000        <1>       mov    word [esi], 0 ; clear
2553                            <1>       ;mov   dword [esi+12], 0 ; clear
2554                            <1>
2555 0000E294 FE0D[53960100]    <1>       dec    byte [timer_events]
2556 0000E29A 7409              <1>       jz     short systimer_12
2557                            <1>
2558                            <1> systimer_11:
2559 0000E29C FECF              <1>       dec    bh
2560 0000E29E 7405              <1>       jz     short systimer_12
2561 0000E2A0 83C610            <1>       add    esi, 16
2562 0000E2A3 EBE0              <1>       jmp    short systimer_10
2563                            <1>
2564                            <1> systimer_12:
2565 0000E2A5 0FB635[B3030300]  <1>       movzx  esi, byte [u.uno]
2566 0000E2AC 08DB              <1>       or     bl, bl ; all timer events or one timer event ?
2567 0000E2AE 740C              <1>       jz     short systimer_13
2568 0000E2B0 8A9E[FF000300]    <1>       mov    bl, [esi+p.timer-1]
2569 0000E2B6 20DB              <1>       and    bl, bl ; previous number of timer events for the process
2570 0000E2B8 7408              <1>       jz     short systimer_14
2571 0000E2BA FECB              <1>       dec    bl  ; previous number of timer events for the process - 1
2572                            <1> systimer_13:
2573 0000E2BC 889E[FF000300]    <1>       mov    [esi+p.timer-1], bl ; 0 ; no timer events for process
2574                            <1> systimer_14:
2575 0000E2C2 FB                <1>       sti    ; enable interrupts
2576 0000E2C3 E934F6FFFF        <1>       jmp    sysret
2577                            <1>
2578                            <1> systimer_15:
2579 0000E2C8 38FB              <1>       cmp    bl, bh ; 16
```

```
2580 0000E2CA 0F8707FFFFFF        <1>         ja      systimer_4      ; max. 16 timer events !
2581                              <1>       ;
2582 0000E2D0 88DA                <1>       mov    dl, bl
2583 0000E2D2 FECA                <1>       dec    dl  ; 16 -> 15 ... 1 -> 0
2584 0000E2D4 C0E204              <1>       shl    dl, 4 ; * 16
2585 0000E2D7 0FB6FA              <1>       movzx  edi, dl
2586 0000E2DA 01F7                <1>       add    edi, esi ; timer_set
2587                              <1>
2588 0000E2DC 3A07                <1>       cmp    al, [edi] ; process number
2589 0000E2DE 0F85F3FEFFFF        <1>         jne     systimer_4
2590                              <1>
2591                              <1>       ; same process ID
2592 0000E2E4 FA                  <1>       cli    ; disable interrupts
2593                              <1>       ; 10/06/2016 ; 02/01/2017
2594                              <1>       ;mov   byte [edi], 0
2595 0000E2E5 66C7070000          <1>       mov    word [edi], 0 ; clear
2596                              <1>       ;mov   dword [edi+12], 0 ; clear
2597 0000E2EA FE0D[53960100]      <1>       dec    byte [timer_events]
2598 0000E2F0 EBB3                <1>       jmp    short systimer_12
2599                              <1>
2600                              <1> sysvideo: ; VIDEO DATA TRANSFER FUNCTIONS
2601                              <1>       ; 15/02/2021
2602                              <1>       ; 12/02/2021
2603                              <1>       ; 11/02/2021
2604                              <1>       ; 10/02/2021
2605                              <1>       ; 08/02/2021
2606                              <1>       ; 07/02/2021
2607                              <1>       ; 01/02/2021, 02/02/2021, 05/02/2021
2608                              <1>       ; 29/01/2021, 30/01/2021, 31/01/2021
2609                              <1>       ; 23/01/2021, 24/01/2021, 28/01/2021
2610                              <1>       ; 18/01/2021, 19/01/2021, 22/01/2021
2611                              <1>       ; 04/01/2021, 10/01/2021, 11/01/2021
2612                              <1>       ; 01/01/2021, 02/01/2021, 03/01/2021
2613                              <1>       ; 28/12/2020, 29/12/2020, 30/12/2020
2614                              <1>       ; 25/12/2020, 26/12/2020
2615                              <1>       ; 21/12/2020, 23/12/2020
2616                              <1>       ; 12/12/2020, 14/12/2020
2617                              <1>       ; 10/12/2020, 11/12/2020
2618                              <1>       ; 03/12/2020, 04/12/2020
2619                              <1>       ; 22/11/2020, 23/11/2020
2620                              <1>       ; 21/11/2020 (TRDOS 386 v2.0.3)
2621                              <1>       ; 12/05/2017
2622                              <1>       ; 11/07/2016
2623                              <1>       ; 13/06/2016
2624                              <1>       ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
2625                              <1>       ;
2626                              <1>       ; VIDEO DATA TRANSFER FUNCTIONS:
2627                              <1>       ;
2628                              <1>       ; Inputs:
2629                              <1>       ;            ; 07/02/2021
2630                              <1>       ;     BH = 0 = VIDEO BIOS Mode 3, tty/text mode data transfers
2631                              <1>       ;          BL =
2632                              <1>       ;            Bits 0&1, Transfer direction
2633                              <1>       ;                  0 - System to system
2634                              <1>       ;                  1 - User to system
2635                              <1>       ;                  2 - System to user
2636                              <1>       ;                  3 - Exhange (Swap) - 28/01/2021
2637                              <1>       ;            Bits 2, Transfer Type
2638                              <1>       ;                  0 - Display page (complete) transfer
2639                              <1>       ;                  1 - Display page window (col,row) transfer
2640                              <1>       ;            ; 28/01/2021
2641                              <1>       ;            Bits 3..7 - Reserved, undefined (must be 0)
2642                              <1>       ;             ; 28/01/2021
2643                              <1>       ;          /// BL = 0 -> System to system (display page) transfer
2644                              <1>       ;             CL = Source page (0FFh = current video page)
2645                              <1>       ;             DL = Destination page (0FFh = current video page)
2646                              <1>       ;             (Note: Nothing to do if src & dest are same page)
2647                              <1>       ;          /// BL = 1&2 -> user to system & system to user transfer
2648                              <1>       ;             ECX = User's buffer address
2649                              <1>       ;             DL = Video page (0FFh = current video page)
2650                              <1>       ;          /// BL = 3 -> exchange (swap) display page ; 28/01/2021
2651                              <1>       ;             ECX = User's buffer address
2652                              <1>       ;             DL = Video page (0FFh = current video page)
2653                              <1>       ;             EDI = Swap address in user's memory (must be > 0)
2654                              <1>       ;          /// BL = 5&6&7 -> user to system, system to user transfer
2655                              <1>       ;            (system window is in current/active display page)
2656                              <1>       ;             ESI = User's buffer address
2657                              <1>       ;             ECX Low 16 bits = Top left column (X1 position)
2658                              <1>       ;             ECX High 16 bits = Top row (Y1 position)
2659                              <1>       ;             EDX Low 16 bits = Bottom right column (X2 position)
2660                              <1>       ;             EDX High 16 bits = Bottom row (Y2 position)
2661                              <1>         ;               If BL = 5 or BL bit 0 & bit 1 are 1 ; 28/01/2021
2662                              <1>       ;             EDI = Swap address (in user's memory space)
2663                              <1>       ;             (If swap address > 0, previous content of the window
2664                              <1>       ;             will be saved into swap area in user's memory space)
2665                              <1>       ;          /// BL = 4 -> system to system transfer
2666                              <1>       ;             ESI = System's source buffer (video page) address
2667                              <1>       ;             ECX Low 16 bits = Top left column (X1 position)
2668                              <1>       ;             ECX High 16 bits = Top row (Y1 position)
2669                              <1>       ;             EDX Low 16 bits = Bottom right column (X2 position)
2670                              <1>       ;             EDX High 16 bits = Bottom row (Y2 position)
2671                              <1>       ;             EDI = System's destination buffer (video page) addr
2672                              <1>       ;
2673                              <1>       ;            ; 06/02/2021
2674                              <1>       ;            ; 05/02/2021
2675                              <1>       ;            ; 01/02/2021, 02/02/2021
2676                              <1>       ;            ; 30/01/2021, 31/02/2021
2677                              <1>       ;            ; 29/01/2021 (major modification)
2678                              <1>       ;            ; 23/11/2020 (major modification)
2679                              <1>       ;            ; 22/11/2020 (bugfixes and extensions)
2680                              <1>       ;     BH = 1 = VGA Graphics (0A0000h) data transfers
2681                              <1>       ;            BL bit 7
2682                              <1>       ;               resolution (screen width) option
2683                              <1>       ;               0 = 320 pixels
2684                              <1>       ;               1 = 640 pixels
```

```
2685                    <1>     ;           .. followings are same with SVGA transfer function
2686                    <1>     ;           BL bit 6
2687                    <1>     ;              direction option
2688                    <1>     ;              0 = user to system (video memory)
2689                    <1>     ;              1 = system to user
2690                    <1>     ;           BL bit 5
2691                    <1>     ;              masked/direct (non-masked) operations
2692                    <1>     ;              1 = masked, 0 = non-masked (direct)
2693                    <1>     ;           BL bit 4
2694                    <1>     ;              page/window option
2695                    <1>     ;              1 = window, 0 = display page (screen)
2696                    <1>     ;           BL bit 0 to 3 (pixel operation types)
2697                    <1>     ;              0 = Copy pixels (colors) ((mask color))
2698                    <1>     ;              1 = Change (New, Fill) color
2699                    <1>     ;              2 = Add color
2700                    <1>     ;              3 = Sub color
2701                    <1>     ;              4 = OR color
2702                    <1>     ;              5 = AND color
2703                    <1>     ;              6 = XOR color
2704                    <1>     ;              7 = NOT color
2705                    <1>     ;              8 = NEG color
2706                    <1>     ;              9 = INC color
2707                    <1>     ;             10 = DEC color
2708                    <1>     ;             11 = Mix (Average) colors
2709                    <1>     ;             12 = Replace pixel colors
2710                    <1>     ;             13 = Copy pixel block(s)
2711                    <1>     ;             14 = Write line(s)
2712                    <1>     ;             15 = Write character (font)
2713                    <1>     ;
2714                    <1>     ;       Input Registers for pixel operations:
2715                    <1>     ;            Same with LFB data transfer function below
2716                    <1>     ;
2717                    <1>     ;           ; 06/02/2021
2718                    <1>     ;           ; 05/02/2021
2719                    <1>     ;           ; 01/02/2021, 02/02/2021
2720                    <1>     ;           ; 30/01/2021, 31/02/2021
2721                    <1>     ;           ; 29/01/2021 (major modification)
2722                    <1>     ;           ; 23/11/2020 (major modification)
2723                    <1>     ;           ; 22/11/2020 (bugfixes and extensions)
2724                    <1>     ;     BH = 2 = Super VGA, LINEAR FRAME BUFFER data transfers
2725                    <1>     ;           BL bit 7
2726                    <1>     ;              unused (invalid), must be 0
2727                    <1>     ;           BL bit 6
2728                    <1>     ;              direction option
2729                    <1>     ;              0 = user to system (video memory)
2730                    <1>     ;              1 = system to user
2731                    <1>     ;           BL bit 5
2732                    <1>     ;              masked/direct (non-masked) operations
2733                    <1>     ;              1 = masked, 0 = non-masked (direct)
2734                    <1>     ;           BL bit 4
2735                    <1>     ;              page/window option
2736                    <1>     ;              1 = window, 0 = display page (screen)
2737                    <1>     ;           BL bit 0 to 3 (pixel operation types)
2738                    <1>     ;              0 = Copy pixels (colors) ((mask color))
2739                    <1>     ;              1 = Change (New, Fill) color
2740                    <1>     ;              2 = Add color
2741                    <1>     ;              3 = Sub color
2742                    <1>     ;              4 = OR color
2743                    <1>     ;              5 = AND color
2744                    <1>     ;              6 = XOR color
2745                    <1>     ;              7 = NOT color
2746                    <1>     ;              8 = NEG color
2747                    <1>     ;              9 = INC color
2748                    <1>     ;             10 = DEC color
2749                    <1>     ;             11 = Mix (Average) colors
2750                    <1>     ;             12 = Replace pixel colors
2751                    <1>     ;             13 = Copy pixel block(s)
2752                    <1>     ;             14 = Write line(s)
2753                    <1>     ;             15 = Write character (font)
2754                    <1>     ;
2755                    <1>     ;        Note: If HW of EBX > 0, it is VESA VBE mode number
2756                    <1>     ;              otherwise, function will be applied
2757                    <1>     ;              to current (VESA VBE) video mode.
2758                    <1>     ;
2759                    <1>     ;        Input Registers for pixel operations:
2760                    <1>     ;           -- user to system & system to system --
2761                    <1>     ;           -- (BL = 0 to 0Fh) -- non-masked, screen --
2762                    <1>     ;           -- (BL = 10h to 1Fh) -- non-masked, window --
2763                    <1>     ;           -- (BL = 20h to 2Fh) -- masked, screen --
2764                    <1>     ;           -- (BL = 30h to 3Fh) -- masked, window --
2765                    <1>     ;           (*) window, (**) masked (***) sys to sys
2766                    <1>     ;           for BL bit 0 to 3
2767                    <1>     ;           00h: COPY PIXELS
2768                    <1>     ;               ECX = start position (row, column) (*)
2769                    <1>     ;                    (HW = row, CX = column)
2770                    <1>     ;               EDX = size (rows, colums) (*)
2771                    <1>     ;                    (HW = rows, DX = columns)
2772                    <1>     ;                     (0 -> invalid)
2773                    <1>     ;                     (1 -> horizontal or vertical line)
2774                    <1>     ;               ESI = user's buffer address
2775                    <1>     ;               EBP = mask color (**)
2776                    <1>     ;                    (this color will be excluded)
2777                    <1>     ;           01h: CHANGE PIXEL COLORS
2778                    <1>     ;           02h: ADD PIXEL COLORS
2779                    <1>     ;           03h: SUB PIXEL COLORS
2780                    <1>     ;           04h: OR PIXEL COLORS
2781                    <1>     ;           05h: AND PIXEL COLORS
2782                    <1>     ;           06h: XOR PIXEL COLORS
2783                    <1>     ;           0Bh: MIX PIXEL COLORS
2784                    <1>     ;                CL = color (8 bit, 256 colors)
2785                    <1>     ;               ECX = color (16 bit and true colors)
2786                    <1>     ;               EDX = start position (row, column) (*)
2787                    <1>     ;                   (HW = row, DX = column)
2788                    <1>     ;               ESI = size (rows, colums) (*)
2789                    <1>     ;                    (HW = rows, SI = columns)
```

```
2790                    <1>     ;               EBP = mask color (**)
2791                    <1>     ;                    (this color will be excluded)
2792                    <1>     ;           07h: NOT PIXEL COLORS
2793                    <1>     ;           08h: NEG PIXEL COLORS
2794                    <1>     ;           09h: INC PIXEL COLORS
2795                    <1>     ;           0Ah: DEC PIXEL COLORS
2796                    <1>     ;               ECX = start position (row, column) (*)
2797                    <1>     ;                    (HW = row, CX = column)
2798                    <1>     ;               EDX = size (rows, colums) (*)
2799                    <1>     ;                    (HW = rows, DX = columns)
2800                    <1>     ;                      (0 -> invalid)
2801                    <1>     ;                      (1 -> horizontal or vertical line)
2802                    <1>     ;               EBP = mask color (**)
2803                    <1>     ;                    (this color will be excluded)
2804                    <1>     ;           0Ch: REPLACE PIXEL COLORS
2805                    <1>     ;                CL = current color (8 bit, 256 colors)
2806                    <1>     ;               ECX = current color (16 bit and true colors)
2807                    <1>     ;                DL = new color (8 bit, 256 colors)
2808                    <1>     ;               EDX = new color (16 bit and true colors)
2809                    <1>     ;               ESI = start position (row, column) (*)
2810                    <1>     ;                    (HW = row, SI = column)
2811                    <1>     ;               EDI = size (rows, colums) (*)
2812                    <1>     ;                    (HW = rows, DI = columns)
2813                    <1>     ;           0Dh: COPY PIXEL BLOCK(S) -full screen-
2814                    <1>     ;              -If BL bit 5 is 0-
2815                    <1>     ;               ECX = start position (row, column) (*)
2816                    <1>     ;                    (HW = row, CX = column)
2817                    <1>     ;               EDX = size (rows, colums) (*)
2818                    <1>     ;                    (HW = rows, DX = columns)
2819                    <1>     ;                      (0 -> invalid)
2820                    <1>     ;                      (1 -> horizontal or vertical line)
2821                    <1>     ;               ESI = destination (row, column) (***)
2822                    <1>     ;              -If BL bit 5 is 1-
2823                    <1>     ;                  CL = color (8 bit, 256 colors)
2824                    <1>     ;                 ECX = color (16 bit and true colors)
2825                    <1>     ;                 EDX = count of blocks (not bytes)
2826                    <1>     ;                      (limit: 2048 blocks)
2827                    <1>     ;                 ESI = user's buffer address
2828                    <1>     ;                    contains 64 bits block data
2829                    <1>     ;                    BLOCK ADDRESS - (row, col), dword
2830                    <1>     ;                    (first 32 bits)
2831                    <1>     ;                    BLOCK SIZE - (rows, cols), dword
2832                    <1>     ;                    (second 32 bits)
2833                    <1>     ;           ; 10/02/2021
2834                    <1>     ;           0Eh: WRITE LINE(s) -full screen-
2835                    <1>     ;              -If BL bit 5 is 0-
2836                    <1>     ;               CL = color (8 bit, 256 colors)
2837                    <1>     ;              ECX = color (16 bit and true colors)
2838                    <1>     ;               DX = low 12 bits - size (length)
2839                    <1>     ;                   high 4 bits - direction or type
2840                    <1>     ;                       0 - Horizontal line
2841                    <1>     ;                       1 - Vertical line
2842                    <1>     ;                     > 1 - undefined, invalid
2843                    <1>     ;              ESI = start position (row, column)
2844                    <1>     ;                   (HW = row, SI = column)
2845                    <1>     ;              -If BL bit 5 is 1-
2846                    <1>     ;                CL = color (8 bit, 256 colors)
2847                    <1>     ;               ECX = color (16 bit and true colors)
2848                    <1>     ;                DX = number of lines (in user buffer)
2849                    <1>     ;                     (limit: 2048 lines)
2850                    <1>     ;               ESI = user's buffer
2851                    <1>     ;                     contains 64 bit data for lines
2852                    <1>     ;                     START POINT: 32 bit (row, col)
2853                    <1>     ;                     LENGTH: 32 bit
2854                    <1>     ;                     high 16 bits - 0
2855                    <1>     ;                     bit 0-11 - length
2856                    <1>     ;                     bit 12-15 - type (length)
2857                    <1>     ;           0Fh: WRITE CHARACTER (FONT)
2858                    <1>     ;                CL = char's color (8 bit, 256 colors)
2859                    <1>     ;               ECX = char's color (16 bit and true colors)
2860                    <1>     ;                DL = Character's ASCII code
2861                    <1>     ;                DH bit 0 -> font height
2862                    <1>     ;                     0 -> 8x16 character font
2863                    <1>     ;                     1 -> 8x8 character font
2864                    <1>     ;                DH bit 1 & 2 -> scale
2865                    <1>     ;                     0 = 1/1 (8 pixels per char row)
2866                    <1>     ;                     1 = 2/1 (16 pixels per char row)
2867                    <1>     ;                     2 = 3/1 (24 pixels per char row)
2868                    <1>     ;                     3 = 4/1 (32 pixels per char row)
2869                    <1>     ;                DH bit 6 -> [ufont] option (1 = use [ufont])
2870                    <1>     ;                If DH bit 7 = 1
2871                    <1>     ;                   USER FONT (from user buffer)
2872                    <1>     ;                      DL = 0 -> 8x8 (width: 1 byte per row)
2873                    <1>     ;                      DL = 1 -> 8x16
2874                    <1>     ;                      DL = 2 -> 16x16 (width: 2 bytes)
2875                    <1>     ;                      DL = 3 -> 16x32
2876                    <1>     ;                      DL = 4 -> 24x24 (width: 3 bytes)
2877                    <1>     ;                      DL = 5 -> 24x48
2878                    <1>     ;                      DL = 6 -> 32x32 (width: 4 bytes)
2879                    <1>     ;                      DL = 7 -> 32x64
2880                    <1>     ;                      DL > 7 -> invalid (unused)
2881                    <1>     ;                   EDI = user's font buffer address
2882                    <1>     ;                      (NOTE: byte order is as row0,row1,row2..)
2883                    <1>     ;               ESI = start position (row, column) (*)
2884                    <1>     ;                     (HW = row, SI = column)
2885                    <1>     ;
2886                    <1>     ;           -- system to user --
2887                    <1>     ;              BL (bit 0 to 7)
2888                    <1>     ;           40h: COPY PIXELS (full screen, display page)
2889                    <1>     ;               EDI = user's buffer address
2890                    <1>     ;           41h: COPY PIXELS (window)
2891                    <1>     ;               ECX = start position (row, column) (*)
2892                    <1>     ;                    (HW = row, CX = column)
2893                    <1>     ;               EDX = size (rows, colums) (*)
2894                    <1>     ;                    (HW = rows, DX = columns)
```

```
2895                    <1>     ;                   (<=1 -> horizontal or vertical line)
2896                    <1>     ;               EDI = user's buffer address
2897                    <1>     ;
2898                    <1>     ;           Example: (29/01/2021)
2899                    <1>     ;               ecx = 00400064h (start at row 64, column 100)
2900                    <1>     ;               edx = 00320048h (size: 50 rows, 72 columns)
2901                    <1>     ;                                 (end at row 114, column 172)
2902                    <1>     ;               If video memory starts at 0A0000h
2903                    <1>     ;               and if resolution is 320x200 (256 colors) ..
2904                    <1>     ;               window start offset: (64*320)+100 = 20580
2905                    <1>     ;               window size: 16072 bytes (pixels)
2906                    <1>     ;               window end offset: 20580+16072 = 36652
2907                    <1>     ;               window start address: 0A0000h+564h = 0A5064h
2908                    <1>     ; Outputs:
2909                    <1>     ;     EAX = transfer/byte count
2910                    <1>     ;
2911                    <1>     ;     NOTE: If the source or destination address passes out of
2912                    <1>     ;     video pages (display memory limits), data will not be transferred
2913                    <1>     ;     and EAX will return as 0.
2914                    <1>     ;
2915                    <1>     ; 08/02/2021
2916                    <1>     ; 07/02/2021
2917                    <1>     ; 04/01/2021
2918                    <1>     ; PIXEL READ/WRITE (in current/active video mode)
2919                    <1>     ;
2920                    <1>     ;     BH = 3 = Read/Write pixel(s) -for all graphics modes-
2921                    <1>     ;          BL =
2922                    <1>     ;            0 = Read pixel
2923                    <1>     ;            1 = Write pixel
2924                    <1>     ;            2 = swap pixel colors
2925                    <1>     ;            3 = mix pixel colors
2926                    <1>     ;          29/01/2021
2927                    <1>     ;            4 = read pixels from user defined positions
2928                    <1>     ;            5 = write single color pixels to user defined positions
2929                    <1>     ;            6 = write multi color pixels to user defined positions
2930                    <1>     ;
2931                    <1>     ;           > 6 = invalid/unimplemented
2932                    <1>     ;
2933                    <1>     ;          .. for BL = 0 to 5
2934                    <1>     ;          CL = color for writing pixel(s) or
2935                    <1>     ;          ECX = color for writing pixel(s) in true color modes
2936                    <1>     ;
2937                    <1>     ;          EDX = Offset from start of video memory (0A0000h)
2938                    <1>     ;              or start of linear frame buffer
2939                    <1>     ;
2940                    <1>     ;          07/02/2021
2941                    <1>     ;          .. for BL = 4
2942                    <1>     ;          EDI = user's destination buffer address for pixel colors
2943                    <1>     ;          29/01/2021
2944                    <1>     ;          .. for BL = 4 & 5
2945                    <1>     ;          ESI = user's source buffer address for BL = 4 & 5
2946                    <1>     ;          (buffer contains dword offset positions for pixels)
2947                    <1>     ;          EDX = number of pixels
2948                    <1>     ;          .. for BL = 6
2949                    <1>     ;          ESI = user's buffer address for BL = 6
2950                    <1>     ;          (buffer contains dword offset position and dword color
2951                    <1>     ;           value for each pixel)
2952                    <1>     ;          EDX = number of pixels
2953                    <1>     ;
2954                    <1>     ; Note:
2955                    <1>     ;     Pixel read/write will be performed in current video mode.
2956                    <1>     ;     If [CRT_MODE] < 0FFh, 0A0000h will be used
2957                    <1>     ;        as video memory and limit will be 65536
2958                    <1>     ;        (new/mix pixel color will be in CL)
2959                    <1>     ;     if [CRT_MODE] = 0FFh (VESA VBE video mode)
2960                    <1>     ;        LFB base address will be used as video memory
2961                    <1>     ;        and limit will be video page size
2962                    <1>     ;        (new/mix pixel color will be in CL)
2963                    <1>     ;
2964                    <1>     ; Outputs:
2965                    <1>     ;     EAX = pixel color (according to BL and ECX input)
2966                    <1>     ;     EAX = 0 (pixel color is 0 or there is an error)
2967                    <1>     ;           (BL will return as 0FFh if there is an error)
2968                    <1>     ;     ; 29/01/2021
2969                    <1>     ;     EAX = number of pixels (for BL input = 4&5&6)
2970                    <1>     ;
2971                    <1>     ; DIRECT (STANDARD VGA/CGA) DISPLAY MEMORY ACCESS FUNCTIONS:
2972                    <1>     ;
2973                    <1>     ;     BH = 4 = CGA direct video memory (0B8000h, 32K) access
2974                    <1>     ;            Page directory & page tables of the user's
2975                    <1>     ;            program will be updated to direct access to
2976                    <1>     ;            0B8000h (32K) video (CGA, color) memory; if
2977                    <1>     ;            there is not a permission conflict or lock!
2978                    <1>     ;              (User's program/process will have permission to
2979                    <1>     ;            access locked display memory if the owner is
2980                    <1>     ;            it's parent.)
2981                    <1>       ;
2982                    <1>     ;         Screen width = 320
2983                    <1>     ;
2984                    <1>     ;     BH = 5 = VGA direct video memory (0A0000h, 64K) access
2985                    <1>     ;            Page directory & page tables of the user's
2986                    <1>     ;            program will be updated to direct access to
2987                    <1>     ;            0A0000h (64K) video (VGA) memory; if there is not
2988                    <1>     ;            a permission conflict or lock!
2989                    <1>     ;              (User's program/process will have permission to
2990                    <1>     ;            access locked display memory if the owner is
2991                    <1>     ;            it's parent.)
2992                    <1>     ;
2993                    <1>     ;         ; 23/11/2020
2994                    <1>     ;         Screen width options = 320, 640, 800
2995                    <1>     ;
2996                    <1>     ; Outputs:
2997                    <1>     ;     EAX = Display memory address for direct access
2998                    <1>     ;           0A0000h for VGA, 0B8000h for CGA
2999                    <1>     ;     (Display memory size: 32K for CGA, 64K for VGA)
```

```
3000                    <1>     ;      EAX = 0 if display page access permission has been denied.
3001                    <1>     ;            (Locked!)
3002                    <1>     ;
3003                    <1>     ; LINEAR FRAME BUFFER ACCESS FUNCTIONS:
3004                    <1>     ;
3005                    <1>     ;      BH = 6 = Linear Frame Buffer direct video memory access
3006                    <1>     ;
3007                    <1>     ;            Page directory & page tables of the user's
3008                    <1>     ;            program will be updated to direct access to
3009                    <1>     ;            the configured LFB (Linear Frame Buffer) address,
3010                    <1>     ;            if there is not a permission conflict or lock!
3011                    <1>     ;              (User's program/process will have permission to
3012                    <1>     ;            access locked display memory if the owner is
3013                    <1>     ;            it's parent.)
3014                    <1>     ;
3015                    <1>     ;          ; 10/12/2020
3016                    <1>     ;          BL = 0FFh -> Direct LFB access for current video mode
3017                    <1>     ;          BL = XXh < 0FFh -> Direct LFB access
3018                    <1>     ;                            for VESA video mode 1XXh
3019                    <1>     ;
3020                    <1>     ;            Return: EAX = Linear Frame Buffer address
3021                    <1>     ;                    (EAX = 0 -> error)
3022                    <1>     ;                  If EAX > 0
3023                    <1>     ;                    EDX = Frame Buffer Size in bytes
3024                    <1>     ;                      BH = Requested Video Mode - 100h
3025                    <1>     ;                           (VESA VBE video modes)
3026                    <1>     ;                      BL = bits per pixel
3027                    <1>     ;                          8 = 256 colors, 8
3028                    <1>     ;                           16 = 65536 colors, 5-6(G)-5
3029                    <1>     ;                           24 = RGB, 16M colors, 8-8-8
3030                    <1>     ;                           32 = RGB + alpha bytes, 8-8-8-8
3031                    <1>     ;                  If BH = 0FFh
3032                    <1>     ;                     BL = VGA/CGA video mode (also EAX = 0)
3033                    <1>     ;
3034                    <1>     ;            ** Function will return with EAX = 0 if the mode
3035                    <1>     ;            is not a valid VESA VBE video mode as 1??h **
3036                    <1>     ;
3037                    <1>     ;            ECX = Pixel resolution
3038                    <1>     ;                  CX = Width (320, 640, 800, 1024, 1366, 1920)
3039                    <1>     ;                  High 16 bits of ECX = Height
3040                    <1>     ;
3041                    <1>     ;      23/11/2020
3042                    <1>     ; ***  GET VIDEO MODE & LINEAR FRAME BUFFER INFO
3043                    <1>     ;      (This function -7- also is used for VGA and CGA modes)
3044                    <1>     ;
3045                    <1>     ;      BH = 7 = Get Linear Frame Buffer info
3046                    <1>     ;
3047                    <1>     ;         ; 22/01/2021
3048                    <1>     ;         ; 10/12/2020
3049                    <1>     ;         BL = any -not used- (22/01/2021)
3050                    <1>     ;
3051                    <1>     ;            Return:
3052                    <1>     ;            EAX = Frame Buffer Address (0 = is not in use)
3053                    <1>     ;            EDX = Frame Buffer Size in bytes
3054                    <1>     ;             BH = Current Video Mode - 100h  ; 22/01/2021
3055                    <1>     ;                 (VESA VBE video modes)
3056                    <1>     ;             BL = bits per pixel
3057                    <1>     ;                   8 = 256 colors, 8
3058                    <1>     ;                   16 = 65536 colors, 5-6(G)-5
3059                    <1>     ;                   24 = RGB, 16M colors, 8-8-8
3060                    <1>     ;                   32 = RGB + alpha bytes, 8-8-8-8
3061                    <1>     ;            If BH = 0FFh
3062                    <1>     ;               BL = VGA/CGA video mode (also EAX = 0)
3063                    <1>     ;
3064                    <1>     ;            Note:
3065                    <1>     ;            Alpha byte will be used as virtual color index.
3066                    <1>     ;            (32 bit pixel colors.. byte 0,1,2 rgb and 3 alpha)
3067                    <1>     ;
3068                    <1>     ;            ** Function will return with EAX = 0 if the mode
3069                    <1>     ;            is not a valid VESA VBE video mode as 1??h **
3070                    <1>     ;
3071                    <1>     ;            ECX = Pixel resolution
3072                    <1>     ;                  CX = Width (320, 640, 800, 1024, 1366, 1920)
3073                    <1>     ;                  High 16 bits of ECX = Height
3074                    <1>     ;
3075                    <1>     ;      NOTE: Each process will have it's own frame buffer
3076                    <1>     ;            address and resolution parameters in 'u' area.
3077                    <1>     ;            Then, if the current frame buffer & resolution
3078                    <1>     ;            is different, frame buffer r/w functions
3079                    <1>     ;            will use scale factor to convert process's
3080                    <1>     ;               pixel coordinates to actual screen coordinates.
3081                    <1>     ;            resolution -> dimensional scale
3082                    <1>     ;            color size -> color scale
3083                    <1>     ;           * RGB (TRUE) colors to 256 colors conversion:
3084                    <1>     ;               TRUE Colors -> 8,8,8 (R,G,B; byte 0 is R)
3085                    <1>     ;            256 colors -> 2,2,2,2 (R,G,B,L; bit 0&1 is R)
3086                    <1>     ;              bit 6&7 -> luminosity base level (0,1,2,3)
3087                    <1>     ;              bit 4&5 -> blue level (0,1,2,3)
3088                    <1>     ;              bit 2%3 -> green level (0,1,2,3)
3089                    <1>     ;              bit 0&1 -> red level (0,1,2,3)
3090                    <1>     ;            Example: total red level : luminosity + red level
3091                    <1>     ;            Luminosity base level: 0 -> 16
3092                    <1>     ;                                   1 -> 32
3093                    <1>     ;                                   2 -> 64
3094                    <1>     ;                                   3 -> 128
3095                    <1>     ;            Color level:
3096                    <1>     ;                                  0 -> 0
3097                    <1>     ;                                  1 -> luminosity level
3098                    <1>     ;                                  2 -> luminosity level + 64
3099                    <1>     ;                                  3 -> 255
3100                    <1>     ;           Luminosity base level = min (R,G,B)
3101                    <1>     ;                   if it is <16, it will be set to 16
3102                    <1>     ;           Color levels: Color values are fixed to (nearest)
3103                    <1>     ;               one of all possible set level (step) values
3104                    <1>     ;               (according to luminosity base level); then
```

```
3105                                    <1>      ;              color levels are set to R-L, G-L, B-L.
3106                                    <1>      ;      For example: If luminosity base level is 32
3107                                    <1>      ;             all possible set values are 0, 32, 96, 255.
3108                                    <1>      ;
3109                                    <1>      ;         * RGB (TRUE) colors to 16 colors conversion:
3110                                    <1>      ;         16 colors: R,B,G,L bits (4 bits)
3111                                    <1>      ;              If any one of R,G,B >= 128 L = 1
3112                                    <1>      ;              If max. value of (R,G,B) >= 32, it is 1
3113                                    <1>      ;                 else all color bits (R&G&B&L) are 0
3114                                    <1>      ;              If the second value >= max. value / 2
3115                                    <1>      ;                 it is 1
3116                                    <1>      ;              If third value value >= max. value / 2
3117                                    <1>      ;                 it is 1
3118                                    <1>      ;         Example: R = 132, G = 64, B = 78
3119                                    <1>      ;                L = 1, R = 1
3120                                    <1>      ;                G < 66 --> G = 0
3121                                    <1>      ;                B >= 66 --> B = 1
3122                                    <1>      ;
3123                                    <1>      ; 10/12/2020
3124                                    <1>      ; SET VIDEO MODE (& RETURN LFB INFO for VESA VBE VIDEO MODES)
3125                                    <1>      ;
3126                                    <1>      ;     BH = 8 = Set Video Mode
3127                                    <1>      ;
3128                                    <1>      ;            BL = Requested Video Mode (method)
3129                                    <1>      ;            If BL = 0FFh
3130                                    <1>      ;               CX = VESA VBE Video Mode
3131                                    <1>      ;               ; 11/12/2020
3132                                    <1>      ;               If EDX > 0 -> LFB INFO (user) buffer addr
3133                                    <1>      ;            If BL < 0FFh, it is VGA/CGA video mode and
3134                                    <1>      ;                  CX & EDX will not be used
3135                                    <1>      ;
3136                                    <1>      ;         NOTE: The last VESA VBE video mode is 11Bh but
3137                                    <1>      ;         TRDOS 386 will permit to set video mode upto 11Fh.
3138                                    <1>      ;         Above 11Fh, from 140h to 1FEh, it will be accepted
3139                                    <1>      ;         as Bochs/Plex86 emulator video mode and it will be
3140                                    <1>      ;         used only if [vbe3] = 2 and detected
3141                                    <1>      ;         video bios is BOCHS/PLEX86/QEMU/VIRTUALBOX vbios.
3142                                    <1>      ;
3143                                    <1>      ; Outputs:
3144                                    <1>      ;     EAX = Requested (Proper) video mode number + 1
3145                                    <1>      ;           ("dec eax" by user will give requested video mode),
3146                                    <1>      ;
3147                                    <1>      ;     If BL input is 0FFh
3148                                    <1>      ;        EDX = LFBINFO table/structure (in user's buffer addr)
3149                                    <1>      ;        EDX = 0 -> Invalid LFBINFO (do not use it)
3150                                    <1>      ;
3151                                    <1>      ;     EAX = 0 -> Error (but EDX will not be changed)
3152                                    <1>      ;
3153                                    <1>      ; 03/12/2020
3154                                    <1>      ; VESA VBE3 VIDEO BIOS (32 bit) PROTECTED MODE INTERFACE SETTINGS
3155                                    <1>      ;
3156                                    <1>      ;     BH = 9 = set/get VBE3 Protected Mode Interface parameters
3157                                    <1>      ;
3158                                    <1>      ;           BL = 0 - Disable protected mode interface
3159                                    <1>      ;                  ([pmi32] = 0)
3160                                    <1>      ;                  Return: AL = 1
3161                                    <1>      ;           BL = 1 - Enable protected mode Interface
3162                                    <1>      ;                  ([pmi32] = 1)
3163                                    <1>      ;                  Return: AL = 2
3164                                    <1>      ;           BL = 2 - Get protected mode interface status
3165                                    <1>      ;                  Return: AL = [pmi32] + 1 (AL = 1 or 2)
3166                                    <1>      ;
3167                                    <1>      ;           If [vbe3] <> 3 --> AL = 0
3168                                    <1>      ;
3169                                    <1>      ;           ; 17/01/2021
3170                                    <1>      ;           BL = 3 - Disable/Cancel restore permission to user
3171                                    <1>      ;                  Return: AL = 1 (if disabled) or 0
3172                                    <1>      ;           BL = 4 - Enable/Give restore permission to user
3173                                    <1>      ;                  Return: AL = 2 (if enabled) or 0
3174                                    <1>      ;           BL = 5 - Get video state save/restore status
3175                                    <1>      ;                   (permission status)
3176                                    <1>      ;                   Return: AL = Status (enabled = 1)
3177                                    <1>      ;                           ; 22/01/2021
3178                                    <1>      ;                           AH = state options ([srvso])
3179                                    <1>      ;           BL = 6 - Return VESA VBE number/status
3180                                    <1>      ;                   Return: AX = status
3181                                    <1>      ;                        if AH =  2, AL > 0 : Emulator
3182                                    <1>      ;                         AH =  3, VESA VBE3 video bios
3183                                    <1>      ;
3184                                    <1>      ;           BL > 6 : not implemented (17/01/2021)
3185                                    <1>      ;
3186                                    <1>      ;     ; 19/01/2021 ([u.uid] check)
3187                                    <1>      ;     Note: Enabling/Disabling are done by root ([u.uid] = 0)
3188                                    <1>      ;           while [multi_tasking] = 0.
3189                                    <1>      ;
3190                                    <1>      ; 23/12/2020
3191                                    <1>      ; VIDEO MEMORY MAPPING:
3192                                    <1>      ;     BH = 10 = Map video memory to user's buffer
3193                                    <1>      ;
3194                                    <1>      ;        BL = 0 : CGA memory (0B8000h) map (32K)
3195                                    <1>      ;        BL = 1 : VGA memory (0A0000h) map (64K)
3196                                    <1>      ;        BL = 2 : SVGA memory (LFB) map to user's buffer
3197                                    <1>      ;
3198                                    <1>      ;     ECX = User's buffer addr (low 12 bits will be cleared)
3199                                    <1>      ;     EDX = Buffer size in bytes (if BL = 2)
3200                                    <1>      ;           (will be trimmed if LFB size < EDX)
3201                                    <1>      ; Return:
3202                                    <1>        ;    EAX = physical address of video memory (buffer)
3203                                    <1>      ;     EBX = mapped (actual) size of video memory (bytes)
3204                                    <1>      ;     ECX = virtual start address of user's video buffer
3205                                    <1>      ;     EDX is same with EDX input
3206                                    <1>      ;
3207                                    <1>      ;     (Note: Memory page boundaries will be applied
3208                                    <1>      ;      to buffer size and buff start addr by rounding down.
3209                                    <1>      ;      Rounded size & address values must not be zero.)
```

```
3210                    <1>     ;      -Normally, it is expected to request mapping by using
3211                    <1>     ;       correct buffer size of current or desired video mode-
3212                    <1>     ;
3213                    <1>     ;      EAX = 0 -> error ! memory can not mapped to user
3214                    <1>     ;
3215                    <1>     ; 04/01/2021
3216                    <1>     ; SET/READ COLOR PALETTE (set/read DAC color registers)
3217                    <1>     ;      ((256 colors (8bpp) VGA/CGA video hardware feature))
3218                    <1>     ;
3219                    <1>     ;      BH = 11 = Set/Read DAC color registers
3220                    <1>     ;
3221                    <1>     ;         (BL<4 Original method for std VGA video hardware)
3222                    <1>     ;         BL = 0 : Read all DAC color registers (256 colors)
3223                    <1>     ;              (6 bit colors, in RGB order)
3224                    <1>     ;         BL = 1 : Set all DAC color registers (256 colors)
3225                    <1>     ;              (6 bit colors, in RGB order)
3226                    <1>     ;         BL = 2 : Read single DAC color register
3227                    <1>     ;              (6 bit color, in RGB order)
3228                    <1>     ;             ((EAX will return with color value))
3229                    <1>     ;             CL = DAC color register (index)
3230                    <1>     ;         BL = 3 : Set/Write single DAC color register
3231                    <1>     ;              (6 bit color, in RGB order, bit 6&7 are 0)
3232                    <1>     ;             ECX byte 0 - DAC color register
3233                    <1>     ;             ECX byte 1 - Red (6 bit)
3234                    <1>     ;             ECX byte 2 - Green (6 bit)
3235                    <1>     ;             ECX byte 3 - Blue (6 bit)
3236                    <1>     ;         (BL>3 Alternative method for BMP files etc.)
3237                    <1>     ;         BL = 4 : Read all DAC color registers (256 colors)
3238                    <1>     ;              (8 bit colors, in BGR order, bit 0&1 is 0)
3239                    <1>     ;         BL = 5 : Set all DAC color registers (256 colors)
3240                    <1>     ;              (8 bit colors, in BGR order, bit 0&1 is 0)
3241                    <1>     ;         BL = 6 : Read single DAC color register
3242                    <1>     ;              (8 bit color, in BGR order, bit 0&1 is 0)
3243                    <1>     ;             ((EAX will return with color value))
3244                    <1>     ;             CL = DAC color register (index)
3245                    <1>     ;          BL = 7 : Set/Write single DAC color register
3246                    <1>     ;               (8 bit color, bit 0&1 are 0)
3247                    <1>     ;             ECX byte 0 - DAC color register
3248                    <1>     ;             ECX byte 1 - Blue (8 bit)
3249                    <1>     ;             ECX byte 2 - Green (8 bit)
3250                    <1>     ;             ECX byte 3 - Red (8 bit)
3251                    <1>     ;
3252                    <1>     ;         BL > 7 : invalid (not implemented)
3253                    <1>     ;
3254                    <1>     ;   if BL bit 2 is 1, 6 bit colors converted to 8 bit colors
3255                    <1>     ;      (low two bits of color bytes will be 0)
3256                    <1>     ;     ((color byte 00111111b will be converted to 11111100b))
3257                    <1>     ;      and RGB byte order will be
3258                    <1>     ;             byte 0 - Blue (low 2 bits ares 0)
3259                    <1>     ;             byte 1 - Green (low 2 bits are 0)
3260                    <1>     ;             byte 2 - Red (low 2 bits are 0)
3261                    <1>     ;             byte 3 - pad (or zero byte)
3262                    <1>     ;      and 256 colors buffer size must be 256*4 = 1024 bytes
3263                    <1>     ;   if BL bit 2 is 0, 6 bit colors will be used directly
3264                    <1>     ;        (high two bits of 8 bit color bytes will be 0)
3265                    <1>     ;     ((dac color 111111b will be converted to 00111111b))
3266                    <1>     ;             byte 0 - Red (high 2 bits are 0)
3267                    <1>     ;             byte 1 - Green (high 2 bits are 0)
3268                    <1>     ;             byte 2 - Blue (high 2 bits are 0)
3269                    <1>     ;      and 256 colors buffer size must be 256*3 = 768 bytes
3270                    <1>     ;
3271                    <1>     ;      ECX = User's buffer addr (256*3 = 768 bytes) or
3272                    <1>     ;            Color
3273                    <1>     ;
3274                    <1>     ; Return:
3275                    <1>     ;    EAX = buffer size (for BL input = 0,1,4,5)
3276                    <1>     ;             or color value (for BL input = 2,3,6,7)
3277                    <1>     ;
3278                    <1>     ; 10/01/2021
3279                    <1>     ; SET/READ FONT DATA
3280                    <1>     ;
3281                    <1>     ;      BH = 12 = Set/Read Character Font Data
3282                    <1>     ;
3283                    <1>     ;         BL = 0 : Disable system font overwrite
3284                    <1>     ;         BL = 1 : Enable system font overwrite
3285                    <1>     ;         BL = 2 : Read system font 8x8
3286                    <1>     ;         BL = 3 : Read system font 8x14
3287                    <1>     ;         BL = 4 : Read system font 8x16
3288                    <1>     ;         BL = 5 : Read user defined font 8x8
3289                    <1>     ;         BL = 6 : Read user defined font 8x16
3290                    <1>     ;         BL = 7 : Write system font 8x8
3291                    <1>     ;         BL = 8 : Write system font 8x14
3292                    <1>     ;         BL = 9 : Write system font 8x16
3293                    <1>     ;         BL = 10 : Write user defined font 8x8
3294                    <1>     ;         BL = 11 : Write user defined font 8x16
3295                    <1>     ;
3296                    <1>     ;        BL > 11 : invalid (not implemented)
3297                    <1>     ;
3298                    <1>     ;      For BL = 1 to 11
3299                    <1>     ;         ECX = number of characters (<= 256)
3300                    <1>     ;         EDX = first character (ascii code in DL)
3301                    <1>     ;         ESI = user's buffer address
3302                    <1>     ;
3303                    <1>     ; Return:
3304                    <1>     ;    EAX = number of characters (ecx input)
3305                    <1>     ;      EAX = 0 -> error
3306                    <1>     ;      (EAX = 256 for BL = 0 and 1 if successful)
3307                    <1>     ;
3308                    <1>     ; Note: system font overwrite permission will be
3309                    <1>     ;       given if [multi_tasking] = 0
3310                    <1>     ;      and [u.uid] = 0 (BL = 1) ; 19/01/2021
3311                    <1>     ;       and if [ufont] bit 7 is 1 (BL = 7,8,9)
3312                    <1>     ;
3313                    <1>     ; 18/01/2021
3314                    <1>     ; SAVE/RESTORE STANDARD VGA VIDEO STATE
```

```
3315                                    <1>      ;
3316                                    <1>      ;     BH = 13 = Save/Restore std VGA video state
3317                                    <1>      ;
3318                                    <1>      ;        BL = 0 : Save VGA state (without DAC regs)
3319                                    <1>      ;            Return: EAX = VideoStateID (>0)
3320                                    <1>      ;                    EAX = 0 (failed!)
3321                                    <1>      ;             (size: 110 bytes for TRDOS 386 v2.0.3)
3322                                    <1>      ;        BL = 1 : Restore VGA state (without DAC regs)
3323                                    <1>      ;            ECX = VideoStateID (to be verified)
3324                                    <1>      ;              Return: EAX = Restore size (>0)
3325                                    <1>      ;        BL = 2 : Save VGA state (complete)
3326                                    <1>      ;            Return: EAX = VideoStateID (>0)
3327                                    <1>      ;                    EAX = 0 (failed!)
3328                                    <1>      ;             (size: 882 bytes for TRDOS 386 v2.0.3)
3329                                    <1>      ;        BL = 3 : Restore VGA state (complete)
3330                                    <1>      ;            ECX = VideoStateID (to be verified)
3331                                    <1>      ;              Return: EAX = Restore size (>0)
3332                                    <1>      ;
3333                                    <1>      ;     * Above options are for saving
3334                                    <1>      ;     *       video state to system memory
3335                                    <1>      ;     *     (location: VBE3VIDEOSTATE, 2048 bytes)
3336                                    <1>      ;
3337                                    <1>      ;        BL = 4 : Save VGA state (without DAC regs)
3338                                    <1>      ;            ECX = buffer address
3339                                    <1>      ;            Return: EAX = transfer count
3340                                    <1>      ;             (size: 110 bytes for TRDOS 386 v2.0.3)
3341                                    <1>      ;            ECX = buffer address
3342                                    <1>      ;        BL = 5 : Restore VGA state (without DAC regs)
3343                                    <1>      ;            ECX = buffer address
3344                                    <1>      ;              Return: EAX = transfer count
3345                                    <1>      ;        BL = 6 : Save VGA state (complete)
3346                                    <1>      ;            ECX = buffer address
3347                                    <1>      ;            Return: EAX = transfer count
3348                                    <1>      ;             (size: 882 bytes for TRDOS 386 v2.0.3)
3349                                    <1>      ;         BL = 7 : Restore VGA state (complete)
3350                                    <1>      ;            ECX = buffer address
3351                                    <1>      ;              Return: EAX = transfer count
3352                                    <1>      ;
3353                                    <1>      ;     * Above options are for saving
3354                                    <1>      ;     *       video state to user's buffer
3355                                    <1>      ;     *     (buffer size: 110 bytes or 882 bytes)
3356                                    <1>      ;
3357                                    <1>      ;       BL > 7 : invalid (not implemented)
3358                                    <1>      ;
3359                                    <1>      ; 18/01/2021
3360                                    <1>      ; SAVE/RESTORE SUPER VGA (VESA VBE 2/3) VIDEO STATE
3361                                    <1>      ;
3362                                    <1>      ;     BH = 14 = Save/Restore SVGA video state
3363                                    <1>      ;
3364                                    <1>      ;        BL = options
3365                                    <1>      ;            bit 0 - Save (0) or Restore (1)
3366                                    <1>      ;            bit 1 - controller hardware state
3367                                    <1>      ;            bit 2 - BIOS data state
3368                                    <1>      ;            bit 3 - DAC state
3369                                    <1>      ;            bit 4 - (extended) Register state
3370                                    <1>      ;            bit 5 - system (0) or user (1) memory
3371                                    <1>      ;            bit 6 - verify without transfer
3372                                    <1>      ;            bit 7 - not used (must be 0)
3373                                    <1>      ;
3374                                    <1>      ;          if bit 0 = 0 and bit 5 = 0
3375                                    <1>      ;            Return:      EAX = VideoStateID (>0)
3376                                    <1>      ;          if bit 0 = 1
3377                                    <1>      ;            ECX = VideoStateID (bit 5 = 0)
3378                                    <1>      ;            Return: EAX = restore (transfer) size
3379                                    <1>      ;          if bit 5 = 1
3380                                    <1>      ;            ECX = Buffer address
3381                                    <1>      ;            Return: EAX = transfer count (size)
3382                                    <1>      ;
3383                                    <1>      ;          ECX = Buffer address or VideoStateID
3384                                    <1>      ;
3385                                    <1>      ;       BL > 127 : invalid (not implemented)
3386                                    <1>      ;
3387                                    <1>      ;  Note: Required buffer size may be > 2048 bytes
3388                                    <1>      ;       (function fails when buff size > 2048 bytes)
3389                                    <1>      ;       proper option must be used
3390                                    <1>      ;
3391                                    <1>      ; 18/01/2021
3392                                    <1>      ; READ VESA EDID (EXTENDED DISPLAY IDENTIFICATION DATA)
3393                                    <1>      ;
3394                                    <1>      ;     BH = 15 = Read VESA EDID for connected monitor
3395                                    <1>      ;              (copy EDID to user)
3396                                    <1>      ;
3397                                    <1>      ;        BL = any
3398                                    <1>      ;
3399                                    <1>      ;     Input:
3400                                    <1>      ;         ECX = user's (EDID) buffer address
3401                                    <1>      ;             (buffer size: 128 bytes)
3402                                    <1>      ;     Output:
3403                                    <1>      ;         EAX = 128 (EDID size)
3404                                    <1>      ;         or EAX = 0 -> Error!
3405                                    <1>      ;            (EDID not ready or buffer addr error)
3406                                    <1>      ;
3407                                    <1>
3408                                    <1>      ; 16/05/2016
3409 0000E2F2 31C0                     <1>      xor    eax, eax
3410 0000E2F4 A3[64030300]             <1>      mov    [u.r0], eax
3411 0000E2F9 20FF                     <1>      and    bh, bh
3412 0000E2FB 0F858B020000             <1>      jnz    sysvideo_13 ; 11/07/2016
3413                                    <1>
3414                                    <1>      ;; 21/11/2020 (TRDOS 386 v2.0.3)
3415                                    <1>      ;; tty/text mode transfers are only for video mode 3
3416                                    <1>
3417                                    <1>      ; 22/11/2020
3418                                    <1>      ;cmp   byte [CRT_MODE], 3 ; 80x25 text, 16 colors
3419                                    <1>      ;jne   sysret ; invalid (nothing to do), [u.r0] = 0
```

```
                                <1>
                                <1>       ; 23/11/2020
                                <1>       ; bit 7,6,5,4 of BL are reserved and it must be 0
                                <1>       ;             for current 'sysvideo' version
                                <1>       ;test  bl, 0F0h
                                <1>       ;;jnz  sysret ; invalid (undefined) !
                                <1>       ;; 28/01/2021
                                <1>       ;jnz   short sysvideo_1_2 ; invalid (undefined) !
                                <1>       ; 28/01/2021
0000E301 80FB07                 <1>       cmp    bl, 7
0000E304 776E                   <1>       ja     short sysvideo_1_2 ; invalid (undefined) !
                                <1>
                                <1>       ; Video mode 0, 80*25 text mode, CGA 16 colors
                                <1>       ; [CRT_MODE] = 3
                                <1>       ;mov   bh, bl
                                <1>       ;shr   bh, 2 ; 4..7 -> 1, 8..11 -> 2, 12..15 -> 3
                                <1>       ;; 21/11/2020
                                <1>       ;;and  bh, bh
                                <1>         ;jnz sysvideo_4  ; Display page window transfer etc.
                                <1>
                                <1>       ; 28/01/2021
0000E306 F6C304                 <1>       test   bl, 4 ; bit 2
0000E309 0F85A2000000           <1>       jnz    sysvideo_4 ; Display page window transfer
                                <1>
                                <1>       ; Display page (complete) transfer
0000E30F 80FA07                 <1>       cmp    dl, 7
                                <1>       ;jnz   sysret ; invalid (nothing to do), [u.r0] = 0
0000E312 760A                   <1>       jna    short sysvideo_0 ; 28/01/2021
0000E314 FEC2                   <1>       inc    dl ; 0FFh -> 0 ("use current video page")
0000E316 755C                   <1>       jnz    short sysvideo_1_2  ; invalid
                                <1>       ; dl = 0 -> use current current page
0000E318 8A15[EE890100]         <1>       mov    dl, [ACTIVE_PAGE]
                                <1> sysvideo_0:
                                <1>       ; 28/01/2021
0000E31E 80FB03                 <1>       cmp    bl, 3
0000E321 7206                   <1>       jb     short sysvideo_0_0
0000E323 09FF                   <1>       or     edi, edi
0000E325 744D                   <1>       jz     short sysvideo_1_2  ; invalid
0000E327 89FE                   <1>       mov    esi, edi ; save swap/exchange buffer addr
                                <1>       ; ecx = user buffer for new video page content
                                <1>       ; esi = user (swap) buffer for saving current video page
                                <1> sysvideo_0_0:
0000E329 BF00800B00             <1>       mov    edi, 0B8000h
                                <1>       ; dl = display page number, destination
0000E32E 66B80010               <1>       mov    ax, 4096 ; 21/11/2020
0000E332 20D2                   <1>       and    dl, dl
0000E334 7408                   <1>       jz     short sysvideo_1
                                <1>       ; 07/02/2021
0000E336 88D6                   <1>       mov    dh, dl
                                <1> sysvideo_0_1:
                                <1>       ; page length = 4096 bytes (but page content is 80*25*2 bytes)
0000E338 01C7                   <1>       add    edi, eax ; 21//11/2020 ([CRT_LEN] = 1000h for mode 3)
0000E33A FECE                   <1>       dec    dh
0000E33C 75FA                   <1>       jnz    short sysvideo_0_1
                                <1> sysvideo_1:
0000E33E 80E303                 <1>       and    bl, 3
0000E341 7536                   <1>       jnz    short sysvideo_2 ; user to system display page transfer
                                <1>       ; system to system video page (content) transfer
                                <1>       ; cl = display page number, source
0000E343 80F907                 <1>       cmp    cl, 7
                                <1>       ;ja    sysret ; invalid (nothing to do), [u.r0] = 0
0000E346 760A                   <1>       jna    short sysvideo_1_0
0000E348 FEC1                   <1>       inc    cl ; 0FFh -> 0 ("use current video page")
0000E34A 7528                   <1>       jnz    short sysvideo_1_2  ; invalid
0000E34C 8A0D[EE890100]         <1>       mov    cl, [ACTIVE_PAGE]
                                <1> sysvideo_1_0:
                                <1>       ; 28/01/2021
0000E352 38D1                   <1>       cmp    cl, dl
0000E354 741E                   <1>       je     short sysvideo_1_2  ; same video page !
                                <1>
                                <1>       ; system to system video/display page transfer (mode 0)
                                <1>       ; 21/11/2020
                                <1>       ;mov   esi, 0B8000h
                                <1>       ;movzx eax, cl
                                <1>       ;mov   edx, 4096 ; [CRT_LEN] = 1000h for video mode 3
                                <1>       ;mov   ecx, edx
                                <1>       ;mul   edx
                                <1>       ;add   esi, eax
                                <1>       ; 28/01/2021
                                <1>       ;movzx esi, cl
                                <1>       ;shl   si, 12 ; * 4096
                                <1>       ;add   esi, 0B8000h
                                <1>
                                <1>       ; 28/01/2021
0000E356 A3[64030300]           <1>       mov    [u.r0], eax ; 4096
0000E35B BE00800B00             <1>       mov    esi, 0B8000h
0000E360 08C9                   <1>       or     cl, cl
0000E362 740A                   <1>       jz     short sysvideo_1_1
                                <1>       ; 07/02/2021
0000E364 88C8                   <1>       mov    al, cl ; display/video page
0000E366 30E4                   <1>       xor    ah, ah
0000E368 66C1E00C               <1>       shl    ax, 12 ; * 4096
0000E36C 01C6                   <1>       add    esi, eax
                                <1> sysvideo_1_1:
                                <1>       ; 21/11/2020
                                <1>       ;;mov  ecx, 4096
                                <1>       ;mov   ecx, eax ; 4096
                                <1>       ;;mov  [u.r0], ecx ; 4096 bytes
                                <1>       ; 28/01/2021
                                <1>       ;mov   [u.r0], cx
0000E36E 31C9                   <1>       xor    ecx, ecx
0000E370 B504                   <1>       mov    ch, 4 ; mov ecx, 1024
                                <1>       ;shr   cx, 2 ; / 4
0000E372 F3A5                   <1>       rep    movsd
                                <1> sysvideo_1_2:
```

```
3525 0000E374 E983F5FFFF          <1>       jmp    sysret
3526                              <1> sysvideo_2:
3527 0000E379 80FB02              <1>       cmp    bl, 2
3528                              <1>         ;ja  sysret; invalid (user to user), [u.r0] = 0
3529                              <1>       ; 28/01/2021
3530 0000E37C 7226                <1>       jb     short sysvideo_3 ; user to system
3531 0000E37E 7404                <1>       je     short sysvideo_2_0 ; system to user
3532                              <1>       ; bl = 3
3533 0000E380 89CA                <1>       mov    edx, ecx ; save user's buffer addr
3534 0000E382 89F1                <1>       mov    ecx, esi ; save swap address
3535                              <1> sysvideo_2_0:
3536                              <1>       ; bl = 2 (or bl = 3, stage 1)
3537                              <1>       ; system to user video/display page transfer (mode 0)
3538 0000E384 89FE                <1>       mov    esi, edi
3539 0000E386 89CF                <1>       mov    edi, ecx ; user buffer ; 28/01/2021
3540                              <1>       ; 21/11/2020
3541 0000E388 89C1                <1>       mov    ecx, eax ; 4096
3542 0000E38A E863370000          <1>       call   transfer_to_user_buffer ; fast transfer
3543                              <1>       ;jc    sysret ; [u.r0] = 0
3544 0000E38F 72E3                <1>       jc     short sysvideo_1_2 ; 28/01/2021
3545                              <1>       ; 28/01/2021
3546 0000E391 80FB03              <1>       cmp    bl, 3
3547 0000E394 7408                <1>       je     short sysvideo_2_2
3548                              <1> sysvideo_2_1:
3549                              <1>       ; 21/11/2020
3550 0000E396 890D[64030300]      <1>       mov    [u.r0], ecx
3551                              <1>       ;;mov  [u.r0], cx
3552                              <1>       ;jmp   sysret
3553 0000E39C EBD6                <1>       jmp    short sysvideo_1_2
3554                              <1>
3555                              <1> sysvideo_2_2:
3556                              <1>       ; bl = 3 (exchange/swap) complete display page
3557                              <1>       ; esi = video page start address
3558                              <1>       ; edx = user's buffer address
3559                              <1>
3560                              <1>       ;mov   ecx, 4096
3561 0000E39E 89F7                <1>       mov    edi, esi ; video page start address
3562 0000E3A0 89D6                <1>       mov    esi, edx ; user's (new page) buffer address
3563 0000E3A2 EB04                <1>       jmp    short sysvideo_2_3
3564                              <1> sysvideo_3:
3565                              <1>       ; bl = 1 (or bl = 3, stage 2)
3566                              <1>       ; user to system video/display page transfer (mode 0)
3567 0000E3A4 89CE                <1>       mov    esi, ecx ; user buffer
3568                              <1>       ; edi = video page address
3569                              <1>       ; 21/11/2020
3570 0000E3A6 89C1                <1>       mov    ecx, eax ; 4096
3571                              <1> sysvideo_2_3:
3572 0000E3A8 E88F370000          <1>       call   transfer_from_user_buffer ; fast transfer
3573                              <1>       ;jc    sysret ; [u.r0] = 0
3574 0000E3AD 72C5                <1>       jc     short sysvideo_1_2 ; 28/01/2021
3575 0000E3AF EBE5                <1>       jmp    short sysvideo_2_1
3576                              <1>
3577                              <1>       ; 21/11/2020
3578                              <1>       ;mov   [u.r0], ecx
3579                              <1>       ;;mov  [u.r0], cx
3580                              <1>       ;;jmp  sysret
3581                              <1>       ;jmp   short sysvideo_1_2 ; 28/01/2021
3582                              <1>
3583                              <1> sysvideo_4:
3584                              <1>       ; 23/11/2020 (TRDOS 386 v2.0.3)
3585                              <1>
3586                              <1>       ; Display page window transfer etc.
3587 0000E3B1 80E303              <1>       and    bl, 3
3588 0000E3B4 0F85F7000000        <1>         jnz     sysvideo_9 ; user to system or system to user
3589                              <1>       ; 21/11/2020
3590                              <1>       ; system to system video/display page window transfer (mode 0)
3591 0000E3BA 81FE00800B00        <1>       cmp    esi, 0B8000h ; source buffer address (system)
3592 0000E3C0 0F8236F5FFFF        <1>       jb     sysret
3593 0000E3C6 81FE00000C00        <1>       cmp    esi, 0B8000h+(4096*8)
3594 0000E3CC 0F832AF5FFFF        <1>       jnb    sysret
3595 0000E3D2 81FF00800B00        <1>       cmp    edi, 0B8000h ; destination buffer address (system)
3596 0000E3D8 0F821EF5FFFF        <1>       jb     sysret
3597 0000E3DE 81FF00000C00        <1>         cmp     edi, 0B8000h+(4096*8)
3598 0000E3E4 0F8312F5FFFF        <1>       jnb    sysret
3599                              <1>       ;
3600 0000E3EA 51                  <1>       push   ecx ; X1 and Y1 position - top left column, row
3601 0000E3EB 0FB7C1              <1>       movzx  eax, cx ; top left column
3602                              <1>       ; 21/11/2020
3603 0000E3EE C1E910              <1>       shr    ecx, 16 ; top row
3604 0000E3F1 740E                <1>       jz     short sysvideo_4_0 ; bypass following code
3605 0000E3F3 52                  <1>       push   edx ; X2 and Y2 position - bottom right column, row
3606 0000E3F4 50                  <1>       push   eax
3607 0000E3F5 66B8A000            <1>       mov    ax, 80*2 ; 80 colums, 160 bytes per row
3608 0000E3F9 F7E1                <1>       mul    ecx
3609                              <1>              ; eax = offset for start row number
3610 0000E3FB 01C6                <1>       add    esi, eax
3611 0000E3FD 01C7                <1>       add    edi, eax
3612 0000E3FF 58                  <1>       pop    eax
3613 0000E400 5A                  <1>       pop    edx
3614                              <1> sysvideo_4_0:
3615 0000E401 66D1E0              <1>       shl    ax, 1 ; * 2 ; convert start column number to offset
3616 0000E404 7404                <1>       jz     short sysvideo_4_1
3617 0000E406 01C6                <1>       add    esi, eax
3618 0000E408 01C7                <1>       add    edi, eax
3619                              <1>              ; esi = source page window start offset
3620                              <1>              ; edi = destination page window start offset
3621                              <1> sysvideo_4_1:
3622 0000E40A 59                  <1>       pop    ecx
3623                              <1>       ;mov   eax, 0B8000h+(80*25*2*8)
3624                              <1>       ; 21/11/2020
3625 0000E40B B800000C00          <1>       mov    eax, 0B8000h+(4096*8)
3626 0000E410 39C6                <1>       cmp    esi, eax
3627 0000E412 0F83E4F4FFFF        <1>       jnb    sysret ; out of video page
3628 0000E418 39C6                <1>       cmp    esi, eax
3629 0000E41A 0F83DCF4FFFF        <1>       jnb    sysret  ;out of video page
```

```
3630                                    <1>
3631 0000E420 56                        <1>     push    esi ; ****
3632 0000E421 57                        <1>     push    edi ; ***
3633 0000E422 52                        <1>     push    edx ; **
3634 0000E423 51                        <1>     push    ecx ; *
3635 0000E424 C1E910                    <1>     shr     ecx, 16 ; top row
3636 0000E427 C1EA10                    <1>     shr     edx, 16 ; bottom row
3637                                    <1>     ; 21/11/2020
3638                                    <1>     ;cmp    ecx, 24 ; max. 25 rows
3639 0000E42A 6683F918                  <1>     cmp     cx, 24
3640 0000E42E 7778                      <1>     ja      short sysvideo_6 ; invalid, [u.r0] = 0
3641                                    <1>     ;cmp    edx, 24 ; max. 25 rows
3642 0000E430 6683FA18                  <1>     cmp     dx, 24
3643 0000E434 7772                      <1>     ja      short sysvideo_6 ; invalid, [u.r0] = 0
3644 0000E436 28CA                      <1>     sub     dl, cl ; end >= start
3645 0000E438 726E                      <1>     jc      short sysvideo_6 ; invalid, [u.r0] = 0
3646                                    <1>     ; 21/11/2020
3647 0000E43A 89D3                      <1>     mov     ebx, edx ; row count - 1
3648 0000E43C 7415                      <1>     jz      short sysvideo_4_2
3649 0000E43E 50                        <1>     push    eax ; *****
3650 0000E43F B8A0000000                <1>     mov     eax, 80*2 ; bytes per row
3651 0000E444 F7E3                      <1>     mul     ebx ; 21/11/2020
3652                                    <1>             ; eax = window end offset
3653                                    <1>             ; (for the last row, before adding column bytes)
3654 0000E446 01C6                      <1>     add     esi, eax
3655 0000E448 01C7                      <1>     add     edi, eax
3656 0000E44A 58                        <1>     pop     eax ; ***** ; mode 3 video memory end (0C000h)
3657 0000E44B 39C6                      <1>     cmp     esi, eax
3658                                    <1>     ;ja     short sysvideo_6 ; invalid, [u.r0] = 0
3659 0000E44D 7359                      <1>     jnb     short sysvideo_6 ; 21/11/2020
3660 0000E44F 39C7                      <1>     cmp     edi, eax
3661                                    <1>     ;ja     short sysvideo_6 ; invalid, [u.r0] = 0
3662 0000E451 7355                      <1>     jnb     short sysvideo_6 ; 21/11/2020
3663                                    <1> sysvideo_4_2:
3664 0000E453 59                        <1>     pop     ecx ; *
3665 0000E454 5A                        <1>     pop     edx ; **
3666 0000E455 81E1FFFF0000              <1>     and     ecx, 0FFFFh
3667 0000E45B 81E2FFFF0000              <1>     and     edx, 0FFFFh
3668                                    <1>     ; 21/11/2020
3669                                    <1>     ;cmp    ecx, 79 ; max. 80 columns
3670 0000E461 6683F94F                  <1>     cmp     cx, 79
3671 0000E465 7743                      <1>     ja      short sysvideo_7 ; invalid, [u.r0] = 0
3672                                    <1>     ;cmp    edx, 79 ; max. 80 columns
3673 0000E467 6683FA4F                  <1>     cmp     dx, 79
3674 0000E46B 773D                      <1>     ja      short sysvideo_7 ; invalid, [u.r0] = 0
3675 0000E46D 28CA                      <1>     sub     dl, cl
3676 0000E46F 7639                      <1>     jna     short sysvideo_7 ; invalid, [u.r0] = 0
3677                                    <1>     ; 21/11/2020
3678 0000E471 740E                      <1>     jz      short sysvideo_4_3
3679                                    <1>     ; edx = column count (width) - 1
3680 0000E473 D0E2                      <1>     shl     dl, 1 ; * 2 ; byte offset (in end row)
3681 0000E475 01D6                      <1>     add     esi, edx
3682 0000E477 01D7                      <1>     add     edi, edx
3683                                    <1>             ; esi = source page window end offset
3684                                    <1>             ; edi = destination page window end offset
3685 0000E479 39C6                      <1>     cmp     esi, eax ; video memory end
3686                                    <1>     ;ja     short sysvideo_7
3687 0000E47B 732D                      <1>     jnb     short sysvideo_7 ; 21/11/2020
3688 0000E47D 39C7                      <1>     cmp     edi, eax ; video memory end
3689                                    <1>     ;ja     short sysvideo_7
3690 0000E47F 7329                      <1>     jnb     short sysvideo_7 ; 21/11/2020
3691                                    <1> sysvideo_4_3:
3692 0000E481 5F                        <1>     pop     edi ; ***
3693 0000E482 5E                        <1>     pop     esi ; ****
3694 0000E483 FEC3                      <1>     inc     bl ; row count - 1 -> row count
3695 0000E485 FEC2                      <1>     inc     dl ; column count
3696 0000E487 88D7                      <1>     mov     bh, dl
3697 0000E489 D0E2                      <1>     shl     dl, 1 ; convert column count to byte offset
3698                                    <1>     ; 21/11/2020
3699                                    <1>     ; esi = source page window start offset
3700                                    <1>     ; edi = destination page window start offset
3701 0000E48B B8A0000000                <1>     mov     eax, 80*2 ; bytes per row
3702                                    <1>     ; Note: 160 bytes per row (even if move count < 160)
3703                                    <1> sysvideo_5:
3704 0000E490 88F9                      <1>     mov     cl, bh ; move/transfer -word- count per row
3705 0000E492 0115[64030300]            <1>     add     [u.r0], edx ; transfer count in bytes
3706 0000E498 F366A5                    <1>     rep     movsw
3707 0000E49B 01C6                      <1>     add     esi, eax ; + 160 bytes to next row
3708 0000E49D 01C7                      <1>     add     edi, eax ; + 160 bytes to next row
3709 0000E49F FECB                      <1>     dec     bl ; remain count of rows
3710 0000E4A1 75ED                      <1>     jnz     short sysvideo_5
3711 0000E4A3 E954F4FFFF                <1>     jmp     sysret
3712                                    <1>
3713                                    <1> sysvideo_6:
3714 0000E4A8 59                        <1>     pop     ecx ; *
3715 0000E4A9 5A                        <1>     pop     edx ; **
3716                                    <1> sysvideo_7:
3717 0000E4AA 5F                        <1>     pop     edi ; ***
3718 0000E4AB 5E                        <1>     pop     esi ; ****
3719                                    <1> sysvideo_8:
3720 0000E4AC E94BF4FFFF                <1>     jmp     sysret
3721                                    <1>
3722                                    <1> sysvideo_9:
3723                                    <1>     ; user to system or system to user window transfer
3724                                    <1>     ; 28/01/2021 (bl = 3 -> swap/exchange)
3725                                    <1>     ;cmp    bl, 2
3726                                    <1>     ;;ja    sysret ; user to user transfer is invalid
3727                                    <1>     ;               ; [u.r0] = 0
3728                                    <1>     ;ja     short sysvideo_8 ; 26/12/2020
3729                                    <1>
3730                                    <1>     ; 28/01/2021
3731 0000E4B1 80FB02                    <1>     cmp     bl, 2
3732 0000E4B4 7604                      <1>     jna     short sysvideo_9_8
3733                                    <1>
3734                                    <1>     ; swap/ exchange video memory and user mem windows
```

```
                                <1>       ; edi = swap address in user's memory space
0000E4B6 21FF                   <1>       and   edi, edi
0000E4B8 74F2                   <1>       jz    short sysvideo_8 ; invalid ; 28/01/2021
                                <1>
                                <1> sysvideo_9_8:
0000E4BA 56                     <1>       push  esi ; ****
0000E4BB 57                     <1>       push  edi ; ***
0000E4BC 52                     <1>       push  edx ; **
0000E4BD 51                     <1>       push  ecx ; *
                                <1>
0000E4BE C1E910                 <1>       shr   ecx, 16 ; top row
0000E4C1 C1EA10                 <1>       shr   edx, 16 ; bottom row
                                <1>
                                <1>       ; 21/11/2020
                                <1>       ;cmp  ecx, 24 ; max. 25 rows
0000E4C4 6683F918               <1>       cmp   cx, 24
0000E4C8 77DE                   <1>       ja    short sysvideo_6 ; invalid, [u.r0] = 0
                                <1>       ;cmp  edx, 24 ; max. 25 rows
0000E4CA 6683FA18               <1>       cmp   dx, 24
0000E4CE 77D8                   <1>       ja    short sysvideo_6 ; invalid, [u.r0] = 0
0000E4D0 28CA                   <1>       sub   dl, cl
0000E4D2 72D4                   <1>       jc    short sysvideo_6 ; invalid, [u.r0] = 0
                                <1>
                                <1>       ;mov  ch, cl ; top row
                                <1>       ;mov  cl, [ACTIVE_PAGE]
                                <1>
                                <1>       ;mov  edi, 80*25*2 ; 4000
                                <1>       ; 21/11/2020
                                <1>       ;mov  edi, 4096 ; [CRT_LEN = 4096 for video mode 3
                                <1>       ;shl  edi, cl  ; ! wrong for page 2 to page 7 !
                                <1>       ;;add edi, 0B8000h - 80*25*2
                                <1>       ;add  edi, 0B8000h - 1000h ; - 4096
                                <1>
                                <1>       ; 21/11/2020
                                <1>       ;xor  eax, eax
                                <1>       ;mov  edi, 0B8000h
                                <1>       ;and  cl, cl ; is video page = 0 ?
                                <1>       ;jz   short sysvideo_9_1 ; yes
                                <1>       ; eax = 0
                                <1>
                                <1> ;sysvideo_9_0:
                                <1>       ;add  ax, 4096
                                <1>       ;dec  cl
                                <1>       ;jnz  short sysvideo_9_0
                                <1>       ;add  edi, eax
                                <1>       ;     ; edi = video page start address
                                <1>
                                <1>       ; 21/11/2020
0000E4D4 BF00800B00             <1>       mov   edi, 0B8000h
0000E4D9 803D[EE890100]00       <1>       cmp   byte [ACTIVE_PAGE], 0
0000E4E0 760C                   <1>       jna   short sysvideo_9_1
                                <1> stsvideo_9_0:
0000E4E2 B010                   <1>       mov   al, 16 ; 4096/256
0000E4E4 F625[EE890100]         <1>       mul   byte [ACTIVE_PAGE]
0000E4EA 86E0                   <1>       xchg  ah, al ; * 256
0000E4EC 01C7                   <1>       add   edi, eax
                                <1>             ; edi = video page start address
                                <1> sysvideo_9_1:
                                <1>       ; bl = transfer direction
                                <1>       ;     (1 = from user, 2 = to user)
                                <1>       ;     (3 = swap) ; 28/01/2021
0000E4EE 88D7                   <1>       mov   bh, dl ; row count - 1
                                <1>       ;mov  dl, ch ; top row
                                <1>       ; 21/11/2020
0000E4F0 08C9                   <1>       or    cl, cl ; top row number
0000E4F2 7408                   <1>       jz    short sysvideo_9_2
                                <1>
                                <1>       ;mov  eax, 80*2
0000E4F4 66B8A000               <1>       mov   ax, 80*2 ; 160, bytes per row
0000E4F8 F7E1                   <1>       mul   ecx ; 22/11/2020
0000E4FA 01C7                   <1>       add   edi, eax
                                <1>             ; edi = window start address for top row
                                <1> sysvideo_9_2:
0000E4FC 59                     <1>       pop   ecx ; *
0000E4FD 5A                     <1>       pop   edx ; **
0000E4FE 81E1FFFF0000           <1>       and   ecx, 0FFFFh
0000E504 81E2FFFF0000           <1>       and   edx, 0FFFFh
                                <1>       ; 21/11/2020
                                <1>       ;cmp  ecx, 79 ; max. 80 columns
0000E50A 6683F94F               <1>       cmp   cx, 79
0000E50E 779A                   <1>       ja    short sysvideo_7 ; invalid, [u.r0] = 0
                                <1>       ;cmp  edx, 79 ; max. 80 columns
0000E510 6683FA4F               <1>       cmp   dx, 79
0000E514 7794                   <1>       ja    short sysvideo_7 ; invalid, [u.r0] = 0
                                <1>
0000E516 28CA                   <1>       sub   dl, cl
0000E518 7290                   <1>       jc    short sysvideo_7 ; invalid, [u.r0] = 0
                                <1>
0000E51A 08C9                   <1>       or    cl, cl ; left column
0000E51C 7404                   <1>       jz    short sysvideo_9_3 ; 0
                                <1>
                                <1>       ; 21/11/2020
0000E51E D0E1                   <1>       shl   cl, 1  ; column * 2
0000E520 01CF                   <1>       add   edi, ecx
                                <1>             ; edi = window start addr for top left column
                                <1> sysvideo_9_3:
0000E522 88D1                   <1>       mov   cl, dl
0000E524 FEC1                   <1>       inc   cl ; column count
0000E526 D0E1                   <1>       shl   cl, 1 ; column count * 2
                                <1>             ; ecx = transfer count per row
                                <1>
0000E528 58                     <1>       pop   eax ; *** (swap address)
0000E529 5E                     <1>       pop   esi ; ****
                                <1>
0000E52A FEC7                   <1>       inc   bh  ; row count
```

```
3840                                    <1>
3841                                    <1>       ;mov   edx, 80*2
3842 0000E52C B2A0                      <1>       mov    dl, 80*2  ; bytes per row
3843                                    <1>       ;
3844                                    <1>       ;cmp   bl, 1 ; transfer direction
3845                                    <1>       ;ja    short sysvideo_11 ; system to user transfer
3846                                    <1>       ; 28/01/2021
3847 0000E52E F6C301                    <1>       test   bl, 1
3848 0000E531 7439                      <1>       jz     short sysvideo_11 ; system to user transfer
3849                                    <1>
3850                                    <1>       ; user to system video/display page window transfer (mode 0)
3851 0000E533 21C0                      <1>       and    eax, eax ; swap address
3852 0000E535 741B                      <1>       jz     short sysvideo_10 ; no window swap
3853                                    <1> sysvideo_9_7: ; 28/01/2021
3854                                    <1>       ; save previous window content in user's buffer (swap address)
3855 0000E537 56                        <1>       push   esi ; user buffer
3856 0000E538 57                        <1>       push   edi ; beginning address of the window
3857                                    <1>       ; 21/11/2020
3858 0000E539 53                        <1>       push   ebx ; save bh
3859 0000E53A 89FE                      <1>       mov    esi, edi
3860 0000E53C 89C7                      <1>       mov    edi, eax
3861                                    <1> sysvideo_9_4:
3862 0000E53E E8AF350000                <1>       call   transfer_to_user_buffer ; fast transfer
3863 0000E543 7208                      <1>       jc     short sysvideo_9_5
3864                                    <1>       ; ecx = actual transfer count (must be same with input)
3865 0000E545 01D6                      <1>       add    esi, edx ; next row address of (video page) window
3866 0000E547 01CF                      <1>       add    edi, ecx ; next row address of user's window
3867                                    <1>              ; Note: ecx may be less than row length of video page
3868                                    <1>              ; user's window uses offset according to window width
3869 0000E549 FECF                      <1>       dec    bh
3870 0000E54B 75F1                      <1>       jnz    short sysvideo_9_4 ; repeat for next row
3871                                    <1> sysvideo_9_5:
3872 0000E54D 5B                        <1>       pop    ebx ; restore bh
3873 0000E54E 5F                        <1>       pop    edi
3874 0000E54F 5E                        <1>       pop    esi
3875                                    <1>       ;jnc   short sysvideo_10
3876 0000E550 7215                      <1>       jc     short sysvideo_9_6 ; 28/01/2021
3877                                    <1> ;sysvideo_9_6:
3878                                    <1> ;     jmp    sysret ; [u.r0] = 0
3879                                    <1>
3880                                    <1> sysvideo_10:
3881                                    <1>       ; user to system video/display page window transfer (mode 0)
3882                                    <1>       ; esi =      user buffer
3883 0000E552 E8E5350000                <1>       call   transfer_from_user_buffer ; fast transfer
3884                                    <1>       ;jc    sysret
3885 0000E557 720E                      <1>       jc     short sysvideo_9_6 ; 28/01/2021
3886                                    <1>       ; ecx = actual transfer count (must be same with input)
3887 0000E559 010D[64030300]            <1>       add    [u.r0], ecx ; actual transfer count
3888 0000E55F 01D7                      <1>       add    edi, edx ; next row address of (video page) window
3889 0000E561 01CE                      <1>       add    esi, ecx ; next row address of user's window
3890                                    <1>              ; Note: ecx may be less than row length of video page
3891                                    <1>              ; user's window uses offset according to window width
3892 0000E563 FECF                      <1>       dec    bh
3893 0000E565 75EB                      <1>       jnz    short sysvideo_10 ; repeat for next row
3894                                    <1>       ;jmp   sysret
3895                                    <1> sysvideo_9_6:
3896 0000E567 E990F3FFFF                <1>       jmp    sysret
3897                                    <1>
3898                                    <1> sysvideo_11:
3899                                    <1>       ; system to user video/display page window transfer (mode 0)
3900 0000E56C 87FE                      <1>       xchg   edi, esi
3901                                    <1> sysvideo_12:
3902                                    <1>       ; esi = beginning addr of the (screen, video page) window
3903                                    <1>       ; edi =      user's buffer
3904 0000E56E E87F350000                <1>       call   transfer_to_user_buffer ; fast transfer
3905 0000E573 0F8283F3FFFF              <1>       jc     sysret
3906                                    <1>       ; ecx = actual transfer count (must be same with input)
3907 0000E579 010D[64030300]            <1>       add    [u.r0], ecx
3908 0000E57F 01D6                      <1>       add    esi, edx ; next row (edx = 160)
3909 0000E581 01CF                      <1>       add    edi, ecx ; next row of the user's window
3910                                    <1>                       ; (ecx <= 160)
3911 0000E583 FECF                      <1>       dec    bh
3912 0000E585 75E7                      <1>       jnz    short sysvideo_12
3913                                    <1> sysvideo_12_0:
3914 0000E587 E970F3FFFF                <1>       jmp    sysret
3915                                    <1>
3916                                    <1> sysvideo_13:
3917                                    <1>       ; 28/12/2020
3918 0000E58C 80FF01                    <1>       cmp    bh, 1
3919 0000E58F 7753                      <1>         ja      short sysvideo_15 ; 23/11/2020
3920                                    <1>
3921                                    <1>       ; 12/02/2021
3922                                    <1>       ; 31/01/2021
3923                                    <1>       ; 29/01/2021
3924                                    <1>       ; 23/11/2020 (TRDOS 386 v2.0.3)
3925                                    <1>       ; (major modification, from mode 13h to all VGA modes,
3926                                    <1>       ;  except super VGA modes and liner frame buffer method)
3927                                    <1>
3928                                    <1>       ; BH = 1 = VGA Graphics mode (0A0000h) data transfers
3929                                    <1>
3930                                    <1>       ; 29/01/2021
3931 0000E591 66B84001                  <1>       mov    ax, 320      ; 320 pixels
3932 0000E595 F6C380                    <1>       test   bl, 80h      ; bit 7 (screen width, 640 pixels)
3933 0000E598 7408                      <1>       jz     short sysvideo_13_0
3934 0000E59A 66D1E0                    <1>       shl    ax, 1  ; 640 pixels
3935                                    <1>       ;
3936 0000E59D 80E37F                    <1>       and    bl, 7Fh
3937 0000E5A0 7405                      <1>       jz     short sysvideo_14
3938                                    <1> sysvideo_13_0:
3939                                    <1>       ; 29/01/2021
3940 0000E5A2 80FB41                    <1>       cmp    bl, 41h
3941 0000E5A5 77E0                      <1>       ja     short sysvideo_12_0 ; invalid (unknown) sub function
3942                                    <1> sysvideo_14:
3943 0000E5A7 66A3[86120300]            <1>       mov    [v_width], ax; save screen width
3944 0000E5AD C705[8A120300]0000-       <1>       mov    dword [v_mem], 0A0000h  ; save video memory address
```

```
3944 0000E5B5 0A00                <1>
3945 0000E5B7 C705[8E120300]0000- <1>       mov    dword [v_siz], 65536      ; save video memory size
3945 0000E5BF 0100                <1>
3946 0000E5C1 C705[96120300]0000- <1>       mov    dword [v_end], 0B0000h    ; save end of video page
3946 0000E5C9 0B00                <1>
3947 0000E5CB B708                <1>       mov    bh, 8
3948 0000E5CD 883D[89120300]      <1>       mov    [v_bpp], bh ; 8     ; bits per pixel (256 colors)
3949 0000E5D3 881D[88120300]      <1>       mov    [v_ops], bl  ; VGA data transfer options
3950                              <1>       ;mov   [maskbuff], ebp
3951 0000E5D9 892D[9A120300]      <1>       mov    [maskcolor], ebp ; 31/01/2021
3952                              <1>                     ; save mask color or bitmask buffer address
3953 0000E5DF E9AD000000          <1>       jmp    sysvideo_15_7
3954                              <1>
3955                              <1> sysvideo_15:
3956                              <1>       ; 28/12/2020
3957 0000E5E4 80FF02              <1>       cmp    bh, 2
3958 0000E5E7 0F87B31E0000        <1>         ja      sysvideo_16
3959                              <1>
3960                              <1>       ; 12/02/2021
3961                              <1>       ; 31/01/2021
3962                              <1>       ; 30/01/2021
3963                              <1>       ; 29/01/2021
3964                              <1>       ; 01/01/2021
3965                              <1>       ; 26/12/2020, 27/12/2020
3966                              <1>       ; 25/12/2020 (TRDOS 386 v2.0.3)
3967                              <1>       ;
3968                              <1>       ; BH = 2 = SVGA (VESA VBE) Graphics mode (LFB) data transfers
3969                              <1>
3970                              <1>       ; 25/12/2020
3971                              <1>       ; resolution table entry will be saved into EBP register
3972                              <1>
3973 0000E5ED 803D[54090000]02    <1>       cmp    byte [vbe3], 2 ; VESA VBE 3 video bios
3974                              <1>                      ; or BOCHS/QEMU/VIRTUALBOX emu video bios
3975 0000E5F4 724E                <1>       jb     short sysvideo_15_4 ; no, nothing to do !
3976 0000E5F6 770B                <1>       ja     short sysvideo_15_0 ; yes
3977                              <1>
3978                              <1>       ; Only Bochs/Plex86 (emu) vbe2 video bios is usable in pmid
3979                              <1>       ; (if [vbe3] = 2)
3980 0000E5F8 A0[55090000]        <1>       mov    al, [vbe2bios] ; Bochs vbios sign is from C0h to C5h
3981 0000E5FD 24F0                <1>       and    al, 0F0h
3982 0000E5FF 3CC0                <1>       cmp    al, 0C0h
3983 0000E601 7541                <1>       jne    short sysvideo_15_4 ; unknown (vbe2) video bios
3984                              <1> sysvideo_15_0:
3985                              <1>       ; 29/01/2021
3986 0000E603 80FB41              <1>       cmp    bl, 41h
3987 0000E606 773C                <1>       ja     short sysvideo_15_4 ; invalid (unknown) sub function
3988                              <1>       ; 29/01/2021
3989 0000E608 881D[88120300]      <1>       mov    [v_ops], bl  ; SVGA data transfer options
3990                              <1>
3991 0000E60E 89D8                <1>       mov    eax, ebx ; hw of ebx is vesa vbe video mode
3992 0000E610 C1E810              <1>       shr    eax, 16 ; ax = vesa vbe video mode
3993 0000E613 7513                <1>       jnz    short sysvideo_15_2
3994                              <1>       ; ax = 0
3995                              <1>
3996                              <1>       ; check & use current video mode
3997 0000E615 803D[CA6F0000]FF    <1>       cmp    byte [CRT_MODE], 0FFh ; extended (SVGA) mode ?
3998 0000E61C 7526                <1>       jne    short sysvideo_15_4 ; no
3999                              <1> sysvideo_15_1:
4000                              <1>       ; use current vbe (svga) video mode
4001 0000E61E 66A1[1E120300]      <1>       mov    ax, [video_mode] ; extended (SVGA, VESA VBE) mode
4002 0000E624 6625FF01            <1>       and    ax, 1FFh ; vesa vbe video mode: 1XXh
4003                              <1> sysvideo_15_2:
4004                              <1>       ; 29/01/2021
4005                              <1>       ;mov   [maskbuff], ebp
4006 0000E628 892D[9A120300]      <1>       mov    [maskcolor], ebp ; 31/01/2021
4007                              <1>                     ; save mask color or bitmask buffer address
4008 0000E62E BD[56730000]        <1>       mov    ebp, b_vbe_modes ; vbe mode table (in 'vidata.s')
4009                              <1> sysvideo_15_3:
4010 0000E633 663B4500            <1>       cmp    ax, [ebp]
4011 0000E637 7410                <1>       je     short sysvideo_15_5
4012 0000E639 83C508              <1>       add    ebp, 8 ; vbe mode table entry size
4013 0000E63C 81FD[16740000]      <1>       cmp    ebp, end_of_b_vbe_modes
4014 0000E642 72EF                <1>       jb     short sysvideo_15_3
4015                              <1> sysvideo_15_4:
4016                              <1>       ; desired video mode is not a valid (implemented)
4017                              <1>       ;       extended (VESA VBE, SVGA) video mode
4018                              <1>       ;
4019                              <1>       ; nothing to do !
4020                              <1>
4021                              <1>       ; [u.r0] = 0  ; return value of EAX
4022 0000E644 E9B3F2FFFF          <1>       jmp    sysret
4023                              <1>
4024                              <1> sysvideo_15_5:
4025                              <1>       ; get LFB address
4026 0000E649 A1[2C120300]        <1>       mov    eax, [LFB_ADDR] ; [LFB_Info+LFBINFO.LFB_addr]
4027 0000E64E 09C0                <1>       or     eax, eax
4028 0000E650 7509                <1>       jnz    short sysvideo_15_6
4029 0000E652 66A1[E30E0000]      <1>       mov    ax, [def_LFB_addr] ; default LFB addr
4030                              <1>                                 ; (for vbe mode 118h)
4031 0000E658 C1E010              <1>       shl    eax, 16
4032                              <1>       ; 27/12/2020
4033                              <1>       ;jz    short sysvideo_15_4
4034                              <1> sysvideo_15_6:
4035                              <1>       ; 29/01/2021
4036 0000E65B A3[8A120300]        <1>       mov    [v_mem], eax ; save video memory address
4037                              <1>
4038                              <1>       ; 27/12/2020
4039                              <1>       ; 26/12/2020
4040 0000E660 8B4502              <1>       mov    eax, [ebp+2] ; width, height
4041                              <1>       ; 29/01/2021
4042 0000E663 66A3[86120300]      <1>       mov    [v_width], ax ; save screen width
4043                              <1>       ; 28/12/2020
4044 0000E669 8A7D06              <1>       mov    bh, [ebp+6] ; bpp
4045                              <1>       ; 29/01/2021
4046 0000E66C 883D[89120300]      <1>       mov    [v_bpp], bh ; bits per pixel
```

```
                                <1>
0000E672 52                     <1>      push   edx ; *
0000E673 0FB7D0                 <1>      movzx  edx, ax  ; width
0000E676 C1E810                 <1>      shr    eax, 16  ; height
0000E679 F7E2                   <1>      mul    edx
                                <1>      ; eax = linear frame buffer size (pixels)
                                <1>      ; 29/01/2021
0000E67B A3[8E120300]           <1>      mov    [v_siz], eax ; save video page size
0000E680 E8F8000000             <1>      call   pixels_to_byte_count
0000E685 0305[8A120300]         <1>      add    eax, [v_mem]
0000E68B A3[96120300]           <1>      mov    [v_end], eax ; save end of video page
0000E690 5A                     <1>      pop    edx ; *
                                <1>
                                <1>      ; bh = bits per pixel
                                <1>      ; (bh will not be used after here, 29/01/2021)
                                <1>
                                <1>      ; bl = pixel operations & options
                                <1>      ; ecx, edx, esi, edi input parameters
                                <1>      ; [maskcolor] = ebp input ; 31/01/2021
                                <1>
                                <1> sysvideo_15_7:
                                <1>      ; 29/01/2021
                                <1>      ;test  byte [v_ops], 40h  ; system to user ?
0000E691 F6C340                 <1>      test   bl, 40h
0000E694 7517                   <1>      jnz    short sysvideo_15_9
                                <1>
0000E696 31C0                   <1>      xor    eax, eax
0000E698 88D8                   <1>      mov    al, bl
0000E69A BB[BBE70000]           <1>      mov    ebx, pixel_ops
0000E69F 240F                   <1>      and    al, 0Fh ; isolate 16 pixel operations
0000E6A1 C0E002                 <1>      shl    al, 2 ; * 4 for dword table pointers
0000E6A4 01C3                   <1>      add    ebx, eax
                                <1>
                                <1>      ; ebx = subroutine address
                                <1>
                                <1>      ; ecx, edx, esi, edi input parameters
                                <1>      ; [maskbuff] = ebp input
                                <1>      ; [maskcolor] = ebp input  ; 31/01/2021
                                <1>
0000E6A6 FF13                   <1>      call   [ebx]
                                <1> sysvideo_15_8:
0000E6A8 E94FF2FFFF             <1>      jmp    sysret
                                <1>
                                <1> sysvideo_15_9:
                                <1>      ; system to user display page or window copy
                                <1>      ;test  byte [v_ops], 1 ; window copy ?
0000E6AD F6C301                 <1>      test   bl, 1
0000E6B0 7521                   <1>      jnz    short sysvideo_15_10
                                <1>
                                <1>      ; display page (full screen copy)
0000E6B2 8B35[8A120300]         <1>      mov    esi, [v_mem] ; LFB start address
0000E6B8 A1[8E120300]           <1>      mov    eax, [v_siz]
0000E6BD E8BB000000             <1>      call   pixels_to_byte_count
0000E6C2 89C1                   <1>      mov    ecx, eax ; transfer count in bytes
                                <1>      ;edi = user's buffer address
0000E6C4 E829340000             <1>      call   transfer_to_user_buffer
0000E6C9 72DD                   <1>      jc     short sysvideo_15_8
0000E6CB 890D[64030300]         <1>      mov    [u.r0], ecx
0000E6D1 EBD5                   <1>      jmp    short sysvideo_15_8
                                <1>
                                <1> sysvideo_15_10:
0000E6D3 E820000000             <1>      call   sysvideo_15_12 ; window preparations
0000E6D8 72CE                   <1>      jc     short sysvideo_15_8
                                <1>
0000E6DA 8B35[92120300]         <1>      mov    esi, [v_str]
                                <1> sysvideo_15_11:
                                <1>      ; esi = window's current row address (video mem)
                                <1>      ; edi = current row (virtual) addr in user's buff
                                <1>      ; ecx = transfer count per row
0000E6E0 E80D340000             <1>      call   transfer_to_user_buffer
0000E6E5 72C1                   <1>      jc     short sysvideo_15_8
0000E6E7 010D[64030300]         <1>      add    [u.r0], ecx
0000E6ED 4B                     <1>      dec    ebx
0000E6EE 74B8                   <1>      jz     short sysvideo_15_8 ; ok.
                                <1>      ; next row
0000E6F0 01CF                   <1>      add    edi, ecx ; next row in user's buffer
0000E6F2 01D6                   <1>      add    esi, edx ; next row of window (system)
0000E6F4 EBEA                   <1>      jmp    short sysvideo_15_11
                                <1>
                                <1> sysvideo_15_14:
0000E6F6 F9                     <1>      stc     ; error !
                                <1> sysvideo_15_15:
0000E6F7 C3                     <1>      retn
                                <1>
                                <1> sysvideo_15_12:
                                <1>      ; 30/01/2021
                                <1>      ; 29/01/2021
                                <1>      ; Window address preparations for window copy
0000E6F8 6621D2                 <1>      and    dx, dx
0000E6FB 74F9                   <1>      jz     short sysvideo_15_14 ; invalid (zero columns)
                                <1>      ;test  edx, 0FFFF0000h
                                <1>      ;jz    short sysvideo_15_14 ; invalid (zero rows)
0000E6FD 81FA00000100           <1>      cmp    edx, 65536
0000E703 72F2                   <1>      jb     short sysvideo_15_15 ; invalid (zero rows)
0000E705 89C8                   <1>      mov    eax, ecx ; start position (row, column)
0000E707 E894000000             <1>      call   calc_pixel_offset
0000E70C 3B05[8E120300]         <1>      cmp    eax, [v_siz]
0000E712 73E2                   <1>      jnb    short sysvideo_15_14 ; out of display page
                                <1>                           ; nothing to do
0000E714 E864000000             <1>      call   pixels_to_byte_count
0000E719 0305[8A120300]         <1>      add    eax, [v_mem]
0000E71F A3[92120300]           <1>      mov    [v_str], eax ; window start address
                                <1>                          ; (addr of top left corner)
                                <1>      ; check column limit
0000E724 89C8                   <1>      mov    eax, ecx
```

```
4152 0000E726 6601D0              <1>       add    ax, dx  ; add columns to start column
4153 0000E729 72CC                <1>       jc     short sysvideo_15_15 ; cf = 1
4154 0000E72B 663B05[86120300]    <1>       cmp    ax, [v_width]
4155 0000E732 77C2                <1>       ja     short sysvideo_15_14
4156                              <1>
4157 0000E734 89D0                <1>       mov    eax, edx ; size
4158 0000E736 2D00000100          <1>       sub    eax, 65536 ; row count -> 0 based row #
4159 0000E73B E860000000          <1>       call   calc_pixel_offset
4160 0000E740 3B05[8E120300]      <1>       cmp    eax, [v_siz] ; video (display) page size
4161 0000E746 77AE                <1>       ja     short sysvideo_15_14 ; out of display page
4162                              <1>                             ; nothing to do
4163 0000E748 E830000000          <1>       call   pixels_to_byte_count
4164 0000E74D 0305[92120300]      <1>       add    eax, [v_str] ; window start address
4165 0000E753 3B05[96120300]      <1>       cmp    eax, [v_end] ; window end address (+1)
4166                              <1>                       ; (addr of bottom right corner +1)
4167 0000E759 779B                <1>       ja     short sysvideo_15_14 ; out of display page
4168                              <1>                             ; nothing to do
4169 0000E75B 89D3                <1>       mov    ebx, edx
4170 0000E75D C1EB10              <1>       shr    ebx, 16
4171                              <1>       ; ebx = row count
4172 0000E760 81E2FFFF0000        <1>       and    edx, 0FFFFh
4173                              <1>       ; edx = transfer count per row (from user's buffer)
4174                              <1>       ;      (in pixels, window width)
4175 0000E766 89D0                <1>       mov    eax, edx
4176 0000E768 E810000000          <1>       call   pixels_to_byte_count
4177 0000E76D 89C1                <1>       mov    ecx, eax
4178                              <1>       ; ecx = transfer count per row (from user's buffer)
4179                              <1>       ;      (in bytes, window width)
4180 0000E76F 66A1[86120300]      <1>       mov    ax, [v_width]
4181 0000E775 E803000000          <1>       call   pixels_to_byte_count
4182 0000E77A 89C2                <1>       mov    edx, eax
4183                              <1>       ; edx = byte count per row
4184 0000E77C C3                  <1>       retn ; cf = 0
4185                              <1>
4186                              <1> pixels_to_byte_count:
4187                              <1>       ; 29/01/2021
4188                              <1>       ; INPUT:
4189                              <1>       ;   eax = pixel count
4190                              <1>       ; OUTPUT:
4191                              <1>       ;   eax = byte count
4192                              <1>       ;
4193 0000E77D 803D[89120300]08    <1>       cmp    byte [v_bpp], 8
4194 0000E784 7619                <1>       jna    short pixtobc_3 ; 8 bit colors
4195 0000E786 803D[89120300]18    <1>       cmp    byte [v_bpp], 24
4196 0000E78D 720A                <1>       jb     short pixtobc_1 ; 16 bit colors
4197 0000E78F 770B                <1>       ja     short pixtobc_2 ; 32 bit colors
4198                              <1>       ; 24 bit pixels
4199                              <1>       ; eax = eax * 3
4200                              <1>       ;push  edx
4201                              <1>       ;mov   edx, eax
4202                              <1>       ;shl   eax, 1
4203                              <1>       ;add   eax, edx
4204                              <1>       ;pop   edx
4205 0000E791 50                  <1>       push   eax
4206 0000E792 D1E0                <1>       shl    eax, 1
4207 0000E794 010424              <1>       add    [esp], eax
4208 0000E797 58                  <1>       pop    eax
4209 0000E798 C3                  <1>       retn
4210                              <1> pixtobc_1:
4211                              <1>       ; 32 bit pixels
4212                              <1>       ; eax = eax * 2
4213 0000E799 D1E0                <1>       shl    eax, 1
4214 0000E79B C3                  <1>       retn
4215                              <1> pixtobc_2:
4216                              <1>       ; 16 bit pixels
4217                              <1>       ; eax = eax * 4
4218 0000E79C C1E002              <1>       shl    eax, 2
4219                              <1> pixtobc_3:
4220 0000E79F C3                  <1>       retn
4221                              <1>
4222                              <1> calc_pixel_offset:
4223                              <1>       ; 29/01/2021
4224                              <1>       ; INPUT:
4225                              <1>       ;   eax = pixel position (row, column)
4226                              <1>       ; OUTPUT:
4227                              <1>       ;   eax = pixel offset (linear address)
4228                              <1>       ;
4229 0000E7A0 52                  <1>       push   edx
4230 0000E7A1 50                  <1>       push   eax
4231 0000E7A2 C1E810              <1>       shr    eax, 16
4232 0000E7A5 7409                <1>       jz     short cpixo_0
4233                              <1>       ; eax = row
4234 0000E7A7 0FB715[86120300]    <1>       movzx  edx, word [v_width]
4235 0000E7AE F7E2                <1>       mul    edx
4236                              <1> cpixo_0:
4237                              <1>       ; eax = row * screen width
4238 0000E7B0 5A                  <1>       pop    edx
4239 0000E7B1 81E2FFFF0000        <1>       and    edx, 0FFFFh
4240                              <1>       ; edx = column
4241 0000E7B7 01D0                <1>       add    eax, edx
4242                              <1>       ; eax = (row * screen width) + column
4243 0000E7B9 5A                  <1>       pop    edx
4244 0000E7BA C3                  <1>       retn
4245                              <1>
4246                              <1>       ; 02/02/2021
4247                              <1>       ; 29/01/2021
4248                              <1> pixel_ops:
4249 0000E7BB [FBE70000]          <1>       dd     pix_op_cpy ; copy pixels (user to system)
4250 0000E7BF [40ED0000]          <1>       dd     pix_op_new ; change (new, fill) color
4251 0000E7C3 [63E80000]          <1>       dd     pix_op_add ; add color (up to 0FFh)
4252 0000E7C7 [15E90000]          <1>       dd     pix_op_sub ; sub color (down to 0)
4253 0000E7CB [2AEB0000]          <1>       dd     pix_op_orc ; or color
4254 0000E7CF [DCEB0000]          <1>       dd     pix_op_and ; and color
4255 0000E7D3 [8EEC0000]          <1>       dd     pix_op_xor ; xor color
4256 0000E7D7 [FEED0000]          <1>       dd     pix_op_not ; not color
```

```
4257 0000E7DB [A8EE0000]          <1>       dd    pix_op_neg ; neg color
4258 0000E7DF [52EF0000]          <1>       dd    pix_op_inc ; inc color
4259 0000E7E3 [FCEF0000]          <1>       dd    pix_op_dec ; dec color
4260 0000E7E7 [C7E90000]          <1>       dd    pix_op_mix ; mix color
4261 0000E7EB [88EA0000]          <1>       dd    pix_op_rpl ; replace color
4262 0000E7EF [A6F00000]          <1>       dd    pix_op_blk ; copy pixel block(s) (sys)
4263 0000E7F3 [5AF10000]          <1>       dd    pix_op_lin ; write line(s)
4264 0000E7F7 [3EF50000]          <1>       dd    pix_op_chr ; write character (font)
4265                              <1>
4266                              <1> pix_op_cpy:
4267                              <1>       ; 06/02/2021
4268                              <1>       ; 30/01/2021
4269                              <1>       ; COPY PIXELS
4270                              <1>       ;
4271                              <1>       ; INPUT:
4272                              <1>       ;  ECX = start position (row, column)
4273                              <1>       ;         (HW = row, CX = column)
4274                              <1>       ;  EDX = size (rows, colums)
4275                              <1>       ;         (HW = rows, DX = columns)
4276                              <1>       ;       (0 -> invalid
4277                              <1>       ;         (1 -> horizontal or vertical line)
4278                              <1>       ;  ESI = user's buffer address
4279                              <1>       ;  [maskcolor] = mask color (to be excluded)
4280                              <1>       ;
4281                              <1>       ; OUTPUT:
4282                              <1>       ;     [u.r0] will be > 0 if succesful
4283                              <1>
4284 0000E7FB F605[88120300]10   <1>       test  byte [v_ops], 10h ; display page or window ?
4285 0000E802 752E               <1>       jnz   short pix_op_cpy_w ; window
4286                              <1>
4287 0000E804 A1[8E120300]        <1>       mov   eax, [v_siz] ; video page size
4288 0000E809 E86FFFFFFF          <1>       call  pixels_to_byte_count
4289 0000E80E 89C1                <1>       mov   ecx, eax ; transfer count
4290 0000E810 8B3D[8A120300]      <1>       mov   edi, [v_mem] ; LFB address
4291                              <1>       ; esi = user's buffer address (virtual)
4292 0000E816 F605[88120300]20    <1>       test  byte [v_ops], 20h ; masked copy ?
4293 0000E81D 7405                <1>       jz    short pix_op_cpy_0 ; no
4294 0000E81F E9640F0000          <1>       jmp   m_pix_op_cpy ; copy pixels except mask color
4295                              <1> pix_op_cpy_0:
4296 0000E824 E813330000          <1>       call  transfer_from_user_buffer
4297 0000E829 7206                <1>       jc    short pix_op_cpy_1
4298 0000E82B 890D[64030300]      <1>       mov   [u.r0], ecx
4299                              <1> pix_op_cpy_1:
4300 0000E831 C3                  <1>       retn  ; 06/02/2021
4301                              <1>
4302                              <1> pix_op_cpy_w:
4303 0000E832 E8C1FEFFFF          <1>       call  sysvideo_15_12 ; window preparations
4304 0000E837 72F8                <1>       jc    short pix_op_cpy_1
4305                              <1>       ; ecx = bytes per row (to be applied)
4306                              <1>       ; edx = screen width in bytes
4307                              <1>       ; ebx = row count
4308 0000E839 8B3D[92120300]      <1>       mov   edi, [v_str]
4309 0000E83F F605[88120300]20    <1>       test  byte [v_ops], 20h ; masked copy ?
4310 0000E846 7405                <1>       jz    short pix_op_cpy_w_0 ; no
4311 0000E848 E905100000          <1>       jmp   m_pix_op_cpy_w ; window copy except mask color
4312                              <1> pix_op_cpy_w_0:
4313                              <1>       ; esi = current row (virtual) addr in user's buff
4314                              <1>       ; edi = window's current row address (video mem)
4315                              <1>       ; ecx = transfer count per row
4316 0000E84D E8EA320000          <1>       call  transfer_from_user_buffer
4317 0000E852 72DD                <1>       jc    short pix_op_cpy_1
4318 0000E854 010D[64030300]      <1>       add   [u.r0], ecx
4319 0000E85A 4B                  <1>       dec   ebx
4320 0000E85B 74D4                <1>       jz    short pix_op_cpy_1 ; ok.
4321                              <1>       ; next row
4322 0000E85D 01CE                <1>       add   esi, ecx ; next row in user's buffer
4323 0000E85F 01D7                <1>       add   edi, edx ; next row of window (system)
4324 0000E861 EBEA                <1>       jmp   short pix_op_cpy_w_0
4325                              <1>
4326                              <1> pix_op_add:
4327                              <1>       ; 31/01/2021
4328                              <1>       ; 30/01/2021
4329                              <1>       ; ADD COLOR
4330                              <1>       ;
4331                              <1>       ; INPUT:
4332                              <1>       ;   CL = color (8 bit, 256 colors)
4333                              <1>       ;  ECX = color (16 bit and true colors)
4334                              <1>       ;  EDX = start position (row, column)
4335                              <1>       ;         (HW = row, DX = column)
4336                              <1>       ;  ESI = size (rows, colums)
4337                              <1>       ;         (HW = rows, SI = columns)
4338                              <1>       ;
4339                              <1>       ;  [maskcolor] = mask color (to be excluded)
4340                              <1>       ;
4341                              <1>       ; OUTPUT:
4342                              <1>       ;     [u.r0] will be > 0 if succesful
4343                              <1>
4344 0000E863 F605[88120300]10   <1>       test  byte [v_ops], 10h ; display page or window ?
4345 0000E86A 7555                <1>       jnz   short pix_op_add_w ; window
4346                              <1>
4347 0000E86C 8B3D[8A120300]      <1>       mov   edi, [v_mem]
4348 0000E872 89FE                <1>       mov   esi, edi
4349                              <1>       ; ecx = color (CL, CX, ECX)
4350 0000E874 89C8                <1>       mov   eax, ecx
4351 0000E876 8B0D[8E120300]      <1>       mov   ecx, [v_siz] ; display page pixel count
4352                              <1>
4353 0000E87C F605[88120300]20    <1>       test  byte [v_ops], 20h ; masked color adding ?
4354 0000E883 7405                <1>       jz    short pix_op_add_0 ; no
4355 0000E885 E9BF100000          <1>       jmp   m_pix_op_add ; add color except mask color
4356                              <1> pix_op_add_0:
4357 0000E88A 803D[89120300]08    <1>       cmp   byte [v_bpp], 8 ; 8bpp
4358 0000E891 7707                <1>       ja    short pix_op_add_1
4359                              <1>
4360                              <1>       ; 256 colors (8bpp)
4361 0000E893 E8460A0000          <1>       call  pix_op_add_8
```

```
4362 0000E898 EB1E                <1>       jmp    short pix_op_add_4
4363                              <1>
4364                              <1> pix_op_add_1:
4365 0000E89A 803D[89120300]18    <1>       cmp    byte [v_bpp], 24 ; 24bpp
4366 0000E8A1 7710                <1>       ja     short pix_op_add_3 ; 32bpp
4367 0000E8A3 7207                <1>       jb     short pix_op_add_2 ; 16bpp
4368                              <1>
4369                              <1>       ; 24 bit true colors
4370 0000E8A5 E8540A0000          <1>       call   pix_op_add_24
4371 0000E8AA EB0C                <1>       jmp    short pix_op_add_4
4372                              <1>
4373                              <1>       ; 65536 colors (16bpp)
4374                              <1> pix_op_add_2:
4375 0000E8AC E83B0A0000          <1>       call   pix_op_add_16
4376 0000E8B1 EB05                <1>       jmp    short pix_op_add_4
4377                              <1>
4378                              <1>       ; 32 bit true colors
4379                              <1> pix_op_add_3:
4380 0000E8B3 E8660A0000          <1>       call   pix_op_add_32
4381                              <1> pix_op_add_4:
4382 0000E8B8 29F7                <1>       sub    edi, esi
4383 0000E8BA 893D[64030300]      <1>       mov    [u.r0], edi
4384                              <1> pix_op_add_5:
4385 0000E8C0 C3                  <1>       retn
4386                              <1>
4387                              <1> pix_op_add_w:
4388                              <1>       ; 31/01/2021
4389 0000E8C1 51                  <1>       push   ecx ; * ; color
4390 0000E8C2 89D1                <1>       mov    ecx, edx ; win start pos
4391 0000E8C4 89F2                <1>       mov    edx, esi ; size (rows, cols)
4392 0000E8C6 E82DFEFFFF          <1>       call   sysvideo_15_12 ; window preparations
4393 0000E8CB 58                  <1>       pop    eax ; * ; color
4394 0000E8CC 72F2                <1>       jc     short pix_op_add_5
4395                              <1>
4396 0000E8CE F605[88120300]20    <1>       test   byte [v_ops], 20h ; masked color adding ?
4397 0000E8D5 7405                <1>       jz     short pix_op_add_w_0 ; no
4398 0000E8D7 E91B110000          <1>       jmp    m_pix_op_add_w
4399                              <1>                      ; window add color except mask color
4400                              <1> pix_op_add_w_0:
4401                              <1>       ; ecx = bytes per row (to be applied)
4402                              <1>       ; edx = screen width in bytes
4403                              <1>       ; ebx = row count
4404                              <1>       ; eax = color
4405                              <1>
4406 0000E8DC 8B3D[92120300]      <1>       mov    edi, [v_str]
4407 0000E8E2 803D[89120300]08    <1>       cmp    byte [v_bpp], 8 ; 8bpp
4408 0000E8E9 7707                <1>       ja     short pix_op_add_w_1
4409                              <1>
4410                              <1>       ; 256 colors (8bpp)
4411 0000E8EB BD[DEF20000]        <1>       mov    ebp, pix_op_add_8
4412 0000E8F0 EB1E                <1>       jmp    short pix_op_add_w_4
4413                              <1>
4414                              <1> pix_op_add_w_1:
4415 0000E8F2 803D[89120300]18    <1>       cmp    byte [v_bpp], 24 ; 24bpp
4416 0000E8F9 7710                <1>       ja     short pix_op_add_w_3 ; 32bpp
4417 0000E8FB 7207                <1>       jb     short pix_op_add_w_2 ; 16bpp
4418                              <1>
4419                              <1>       ; 24 bit true colors
4420 0000E8FD BD[FEF20000]        <1>       mov    ebp, pix_op_add_24
4421 0000E902 EB0C                <1>       jmp    short pix_op_add_w_4
4422                              <1>
4423                              <1>       ; 65536 colors (16bpp)
4424                              <1> pix_op_add_w_2:
4425 0000E904 BD[ECF20000]        <1>       mov    ebp, pix_op_add_16
4426 0000E909 EB05                <1>       jmp    short pix_op_add_w_4
4427                              <1>
4428                              <1>       ; 32 bit true colors
4429                              <1> pix_op_add_w_3:
4430 0000E90B BD[1EF30000]        <1>       mov    ebp, pix_op_add_32
4431                              <1> pix_op_add_w_4:
4432 0000E910 E95F010000          <1>       jmp    pix_op_add_w_x
4433                              <1>
4434                              <1> pix_op_sub:
4435                              <1>       ; 31/01/2021
4436                              <1>       ; SUB COLOR
4437                              <1>       ;
4438                              <1>       ; INPUT:
4439                              <1>       ;   CL = color (8 bit, 256 colors)
4440                              <1>       ;  ECX = color (16 bit and true colors)
4441                              <1>       ;  EDX = start position (row, column)
4442                              <1>       ;        (HW = row, DX = column)
4443                              <1>       ;  ESI = size (rows, colums)
4444                              <1>       ;        (HW = rows, SI = columns)
4445                              <1>       ;
4446                              <1>       ;  [maskcolor] = mask color (to be excluded)
4447                              <1>       ;
4448                              <1>       ; OUTPUT:
4449                              <1>       ;      [u.r0] will be > 0 if succesful
4450                              <1>
4451 0000E915 F605[88120300]10    <1>       test   byte [v_ops], 10h ; display page or window ?
4452 0000E91C 7555                <1>       jnz    short pix_op_sub_w ; window
4453                              <1>
4454 0000E91E 8B3D[8A120300]      <1>       mov    edi, [v_mem]
4455 0000E924 89FE                <1>       mov    esi, edi
4456                              <1>       ; ecx = color (CL, CX, ECX)
4457 0000E926 89C8                <1>       mov    eax, ecx
4458 0000E928 8B0D[8E120300]      <1>       mov    ecx, [v_siz] ; display page pixel count
4459                              <1>
4460 0000E92E F605[88120300]20    <1>       test   byte [v_ops], 20h ; masked color subtract ?
4461 0000E935 7405                <1>       jz     short pix_op_sub_0 ; no
4462 0000E937 E9EE100000          <1>       jmp    m_pix_op_sub ; sub color except mask color
4463                              <1> pix_op_sub_0:
4464 0000E93C 803D[89120300]08    <1>       cmp    byte [v_bpp], 8 ; 8bpp
4465 0000E943 7707                <1>       ja     short pix_op_sub_1
4466                              <1>
```

```
                                <1>       ; 256 colors (8bpp)
0000E945 E8E3090000             <1>       call   pix_op_sub_8
0000E94A EB1E                   <1>       jmp    short pix_op_sub_4
                                <1>
                                <1> pix_op_sub_1:
0000E94C 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000E953 7710                   <1>       ja     short pix_op_sub_3 ; 32bpp
0000E955 7207                   <1>       jb     short pix_op_sub_2 ; 16bpp
                                <1>
                                <1>       ; 24 bit true colors
0000E957 E8F4090000             <1>       call   pix_op_sub_24
0000E95C EB0C                   <1>       jmp    short pix_op_sub_4
                                <1>
                                <1>       ; 65536 colors (16bpp)
                                <1> pix_op_sub_2:
0000E95E E8DA090000             <1>       call   pix_op_sub_16
0000E963 EB05                   <1>       jmp    short pix_op_sub_4
                                <1>
                                <1>       ; 32 bit true colors
                                <1> pix_op_sub_3:
0000E965 E8010A0000             <1>       call   pix_op_sub_32
                                <1> pix_op_sub_4:
0000E96A 29F7                   <1>       sub    edi, esi
0000E96C 893D[64030300]         <1>       mov    [u.r0], edi
                                <1> pix_op_sub_5:
0000E972 C3                     <1>       retn
                                <1>
                                <1> pix_op_sub_w:
                                <1>       ; 31/01/2021
0000E973 51                     <1>       push   ecx ; * ; color
0000E974 89D1                   <1>       mov    ecx, edx ; win start pos
0000E976 89F2                   <1>       mov    edx, esi ; size (rows, cols)
0000E978 E87BFDFFFF             <1>       call   sysvideo_15_12 ; window preparations
0000E97D 58                     <1>       pop    eax ; * ; color
0000E97E 72F2                   <1>       jc     short pix_op_sub_5
                                <1>
0000E980 F605[88120300]20       <1>       test   byte [v_ops], 20h ; masked color subtract ?
0000E987 7405                   <1>       jz     short pix_op_sub_w_0 ; no
0000E989 E945110000             <1>       jmp    m_pix_op_sub_w
                                <1>                      ; window sub color except mask color
                                <1> pix_op_sub_w_0:
                                <1>       ; ecx = bytes per row (to be applied)
                                <1>       ; edx = screen width in bytes
                                <1>       ; ebx = row count
                                <1>       ; eax = color
                                <1>
0000E98E 8B3D[92120300]         <1>       mov    edi, [v_str]
0000E994 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000E99B 7707                   <1>       ja     short pix_op_sub_w_1
                                <1>
                                <1>       ; 256 colors (8bpp)
0000E99D BD[2DF30000]           <1>       mov    ebp, pix_op_sub_8
0000E9A2 EB1E                   <1>       jmp    short pix_op_sub_w_4
                                <1>
                                <1> pix_op_sub_w_1:
0000E9A4 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000E9AB 7710                   <1>       ja     short pix_op_sub_w_3 ; 32bpp
0000E9AD 7207                   <1>       jb     short pix_op_sub_w_2 ; 16bpp
                                <1>
                                <1>       ; 24 bit true colors
0000E9AF BD[50F30000]           <1>       mov    ebp, pix_op_sub_24
0000E9B4 EB0C                   <1>       jmp    short pix_op_sub_w_4
                                <1>
                                <1>       ; 65536 colors (16bpp)
                                <1> pix_op_sub_w_2:
0000E9B6 BD[3DF30000]           <1>       mov    ebp, pix_op_sub_16
0000E9BB EB05                   <1>       jmp    short pix_op_sub_w_4
                                <1>
                                <1>       ; 32 bit true colors
                                <1> pix_op_sub_w_3:
0000E9BD BD[6BF30000]           <1>       mov    ebp, pix_op_sub_32
                                <1> pix_op_sub_w_4:
0000E9C2 E9AD000000             <1>       jmp    pix_op_sub_w_x
                                <1>
                                <1> pix_op_mix:
                                <1>       ; 31/01/2021
                                <1>       ; MIX COLOR
                                <1>       ;
                                <1>       ; INPUT:
                                <1>       ;   CL = color (8 bit, 256 colors)
                                <1>       ;  ECX = color (16 bit and true colors)
                                <1>       ;  EDX = start position (row, column)
                                <1>       ;        (HW = row, DX = column)
                                <1>       ;  ESI = size (rows, colums)
                                <1>       ;        (HW = rows, SI = columns)
                                <1>       ;
                                <1>       ;  [maskcolor] = mask color (to be excluded)
                                <1>       ;
                                <1>       ; OUTPUT:
                                <1>       ;      [u.r0] will be > 0 if succesful
                                <1>
0000E9C7 F605[88120300]10       <1>       test   byte [v_ops], 10h ; display page or window ?
0000E9CE 7555                   <1>       jnz    short pix_op_mix_w ; window
                                <1>
0000E9D0 8B3D[8A120300]         <1>       mov    edi, [v_mem]
0000E9D6 89FE                   <1>       mov    esi, edi
                                <1>       ; ecx = color (CL, CX, ECX)
0000E9D8 89C8                   <1>       mov    eax, ecx
0000E9DA 8B0D[8E120300]         <1>       mov    ecx, [v_siz] ; display page pixel count
                                <1>
0000E9E0 F605[88120300]20       <1>       test   byte [v_ops], 20h ; masked color mix ?
0000E9E7 7405                   <1>       jz     short pix_op_mix_0 ; no
0000E9E9 E918110000             <1>       jmp    m_pix_op_mix ; mix colors except mask color
                                <1> pix_op_mix_0:
0000E9EE 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
```

```
4572 0000E9F5 7707               <1>       ja     short pix_op_mix_1
4573                             <1>
4574                             <1>       ; 256 colors (8bpp)
4575 0000E9F7 E8EE090000         <1>       call   pix_op_mix_8
4576 0000E9FC EB1E               <1>       jmp    short pix_op_mix_4
4577                             <1>
4578                             <1> pix_op_mix_1:
4579 0000E9FE 803D[89120300]18   <1>       cmp    byte [v_bpp], 24 ; 24bpp
4580 0000EA05 7710               <1>       ja     short pix_op_mix_3 ; 32bpp
4581 0000EA07 7207               <1>       jb     short pix_op_mix_2 ; 16bpp
4582                             <1>
4583                             <1>       ; 24 bit true colors
4584 0000EA09 E8F7090000         <1>       call   pix_op_mix_24
4585 0000EA0E EB0C               <1>       jmp    short pix_op_mix_4
4586                             <1>
4587                             <1>       ; 65536 colors (16bpp)
4588                             <1> pix_op_mix_2:
4589 0000EA10 E8E1090000         <1>       call   pix_op_mix_16
4590 0000EA15 EB05               <1>       jmp    short pix_op_mix_4
4591                             <1>
4592                             <1>       ; 32 bit true colors
4593                             <1> pix_op_mix_3:
4594 0000EA17 E8050A0000         <1>       call   pix_op_mix_32
4595                             <1> pix_op_mix_4:
4596 0000EA1C 29F7               <1>       sub    edi, esi
4597 0000EA1E 893D[64030300]     <1>       mov    [u.r0], edi
4598                             <1> pix_op_mix_5:
4599 0000EA24 C3                 <1>       retn
4600                             <1>
4601                             <1> pix_op_mix_w:
4602                             <1>       ; 31/01/2021
4603 0000EA25 51                 <1>       push   ecx ; * ; color
4604 0000EA26 89D1               <1>       mov    ecx, edx ; win start pos
4605 0000EA28 89F2               <1>       mov    edx, esi ; size (rows, cols)
4606 0000EA2A E8C9FCFFFF         <1>       call   sysvideo_15_12 ; window preparations
4607 0000EA2F 58                 <1>       pop    eax ; * ; color
4608 0000EA30 72F2               <1>       jc     short pix_op_mix_5
4609                             <1>
4610 0000EA32 F605[88120300]20   <1>       test   byte [v_ops], 20h ; masked color mix ?
4611 0000EA39 7405               <1>       jz     short pix_op_mix_w_0 ; no
4612 0000EA3B E965110000         <1>       jmp    m_pix_op_mix_w
4613                             <1>                     ; window mix colors except mask color
4614                             <1> pix_op_mix_w_0:
4615                             <1>       ; ecx = bytes per row (to be applied)
4616                             <1>       ; edx = screen width in bytes
4617                             <1>       ; ebx = row count
4618                             <1>       ; eax = color
4619                             <1>
4620 0000EA40 8B3D[92120300]     <1>       mov    edi, [v_str]
4621 0000EA46 803D[89120300]08   <1>       cmp    byte [v_bpp], 8 ; 8bpp
4622 0000EA4D 7707               <1>       ja     short pix_op_mix_w_1
4623                             <1>
4624                             <1>       ; 256 colors (8bpp)
4625 0000EA4F BD[EAF30000]       <1>       mov    ebp, pix_op_mix_8
4626 0000EA54 EB1E               <1>       jmp    short pix_op_mix_w_x
4627                             <1>
4628                             <1> pix_op_mix_w_1:
4629 0000EA56 803D[89120300]18   <1>       cmp    byte [v_bpp], 24 ; 24bpp
4630 0000EA5D 7710               <1>       ja     short pix_op_mix_w_3 ; 32bpp
4631 0000EA5F 7207               <1>       jb     short pix_op_mix_w_2 ; 16bpp
4632                             <1>
4633                             <1>       ; 24 bit true colors
4634 0000EA61 BD[05F40000]       <1>       mov    ebp, pix_op_mix_24
4635 0000EA66 EB0C               <1>       jmp    short pix_op_mix_w_x
4636                             <1>
4637                             <1>       ; 65536 colors (16bpp)
4638                             <1> pix_op_mix_w_2:
4639 0000EA68 BD[F6F30000]       <1>       mov    ebp, pix_op_mix_16
4640 0000EA6D EB05               <1>       jmp    short pix_op_mix_w_x
4641                             <1>
4642                             <1>       ; 32 bit true colors
4643                             <1> pix_op_mix_w_3:
4644 0000EA6F BD[21F40000]       <1>       mov    ebp, pix_op_mix_32
4645                             <1>       ;jmp   short pix_op_mix_w_x
4646                             <1>
4647                             <1> pix_op_mix_w_x:
4648                             <1> pix_op_add_w_x:
4649                             <1> pix_op_sub_w_x:
4650                             <1> pix_op_rpl_w_x:
4651                             <1> pix_op_orc_w_x:
4652                             <1> pix_op_and_w_x:
4653                             <1> pix_op_xor_w_x:
4654                             <1>       ; 31/01/2021
4655                             <1>       ; ecx = bytes per row (to be applied)
4656                             <1>       ; edx = windows (screen) width in bytes
4657                             <1>       ; ebx = row count
4658                             <1>       ; eax = color
4659                             <1>       ; ebp = pixel operation subroutine address
4660 0000EA74 52                 <1>       push   edx
4661 0000EA75 51                 <1>       push   ecx
4662 0000EA76 57                 <1>       push   edi
4663 0000EA77 FFD5               <1>       call   ebp ; call pixel-row operation
4664 0000EA79 5F                 <1>       pop    edi
4665 0000EA7A 59                 <1>       pop    ecx ; bytes per row
4666 0000EA7B 010D[64030300]     <1>       add    [u.r0], ecx
4667 0000EA81 5A                 <1>       pop    edx
4668 0000EA82 01D7               <1>       add    edi, edx ; next row
4669 0000EA84 4B                 <1>       dec    ebx
4670 0000EA85 75ED               <1>       jnz    short pix_op_mix_w_x
4671 0000EA87 C3                 <1>       retn
4672                             <1>
4673                             <1> pix_op_rpl:
4674                             <1>       ; 01/02/2021
4675                             <1>       ; REPLACE COLOR
4676                             <1>       ;
```

```
                                <1>       ; INPUT:
                                <1>       ;   CL = old/current color (8 bit, 256 colors)
                                <1>       ;  ECX = old/current color (16 bit and true colors)
                                <1>       ;   DL = new color (8 bit, 256 colors)
                                <1>       ;  EDX = new color (16 bit and true colors)
                                <1>       ;  ESI = start position (row, column)
                                <1>       ;        (HW = row, DX = column)
                                <1>       ;  EDI = size (rows, colums)
                                <1>       ;        (HW = rows, SI = columns)
                                <1>       ; OUTPUT:
                                <1>       ;      [u.r0] will be > 0 if succesful
                                <1>
0000EA88 F605[88120300]10       <1>       test   byte [v_ops], 10h ; display page or window ?
0000EA8F 754D                   <1>       jnz    short pix_op_rpl_w ; window
                                <1>
0000EA91 8B3D[8A120300]         <1>       mov    edi, [v_mem]
0000EA97 89FE                   <1>       mov    esi, edi
                                <1>       ; ecx = old color (CL, CX, ECX) -to be replaced with-
                                <1>       ; edx = new color (CL, CX, ECX) -new one-
0000EA99 89D0                   <1>       mov    eax, edx ; new color
0000EA9B 890D[9A120300]         <1>       mov    [maskcolor], ecx ; old color
0000EAA1 8B0D[8E120300]         <1>       mov    ecx, [v_siz] ; display page pixel count
                                <1> pix_op_rpl_0:
0000EAA7 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000EAAE 7707                   <1>       ja     short pix_op_rpl_1
                                <1>
                                <1>       ; 256 colors (8bpp)
0000EAB0 E8300A0000             <1>       call   pix_op_rpl_8
0000EAB5 EB1E                   <1>       jmp    short pix_op_rpl_4
                                <1>
                                <1> pix_op_rpl_1:
0000EAB7 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000EABE 7710                   <1>       ja     short pix_op_rpl_3 ; 32bpp
0000EAC0 7207                   <1>       jb     short pix_op_rpl_2 ; 16bpp
                                <1>
                                <1>       ; 24 bit true colors
0000EAC2 E8410A0000             <1>       call   pix_op_rpl_24
0000EAC7 EB0C                   <1>       jmp    short pix_op_rpl_4
                                <1>
                                <1>       ; 65536 colors (16bpp)
                                <1> pix_op_rpl_2:
0000EAC9 E8270A0000             <1>       call   pix_op_rpl_16
0000EACE EB05                   <1>       jmp    short pix_op_rpl_4
                                <1>
                                <1>       ; 32 bit true colors
                                <1> pix_op_rpl_3:
0000EAD0 E8550A0000             <1>       call   pix_op_rpl_32
                                <1> pix_op_rpl_4:
0000EAD5 29F7                   <1>       sub    edi, esi
0000EAD7 893D[64030300]         <1>       mov    [u.r0], edi
                                <1> pix_op_rpl_5:
0000EADD C3                     <1>       retn
                                <1>
                                <1> pix_op_rpl_w:
                                <1>       ; 01/02/2021
0000EADE 890D[9A120300]         <1>       mov    [maskcolor], ecx ; old color
0000EAE4 52                     <1>       push   edx ; * ; new color
0000EAE5 89F1                   <1>       mov    ecx, esi ; win start pos
0000EAE7 89FA                   <1>       mov    edx, edi ; size (rows, cols)
0000EAE9 E80AFCFFFF             <1>       call   sysvideo_15_12 ; window preparations
0000EAEE 58                     <1>       pop    eax ; * ; new color
0000EAEF 72EC                   <1>       jc     short pix_op_rpl_5
                                <1>
                                <1>       ; replace window color
                                <1> pix_op_rpl_w_0:
                                <1>       ; ecx = bytes per row (to be applied)
                                <1>       ; edx = screen width in bytes
                                <1>       ; ebx = row count
                                <1>       ; eax = new color
                                <1>       ; [maskcolor] = old color
                                <1>
0000EAF1 8B3D[92120300]         <1>       mov    edi, [v_str]
                                <1>
0000EAF7 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000EAFE 7707                   <1>       ja     short pix_op_rpl_w_1
                                <1>
                                <1>       ; 256 colors (8bpp)
0000EB00 BD[E5F40000]           <1>       mov    ebp, pix_op_rpl_8
0000EB05 EB1E                   <1>       jmp    short pix_op_rpl_w_4
                                <1>
                                <1> pix_op_rpl_w_1:
0000EB07 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000EB0E 7710                   <1>       ja     short pix_op_rpl_w_3 ; 32bpp
0000EB10 7207                   <1>       jb     short pix_op_rpl_w_2 ; 16bpp
                                <1>
                                <1>       ; 24 bit true colors
0000EB12 BD[08F50000]           <1>       mov    ebp, pix_op_rpl_24
0000EB17 EB0C                   <1>       jmp    short pix_op_rpl_w_4
                                <1>
                                <1>       ; 65536 colors (16bpp)
                                <1> pix_op_rpl_w_2:
0000EB19 BD[F5F40000]           <1>       mov    ebp, pix_op_rpl_16
0000EB1E EB05                   <1>       jmp    short pix_op_rpl_w_4
                                <1>
                                <1>       ; 32 bit true colors
                                <1> pix_op_rpl_w_3:
0000EB20 BD[2AF50000]           <1>       mov    ebp, pix_op_rpl_32
                                <1> pix_op_rpl_w_4:
0000EB25 E94AFFFFFF             <1>       jmp    pix_op_rpl_w_x
                                <1>
                                <1> pix_op_orc:
                                <1>       ; 31/01/2021
                                <1>       ; OR COLOR
                                <1>       ;
                                <1>       ; INPUT:
```

```
                                <1>       ;   CL = color (8 bit, 256 colors)
                                <1>       ;  ECX = color (16 bit and true colors)
                                <1>       ;  EDX = start position (row, column)
                                <1>       ;         (HW = row, DX = column)
                                <1>       ;  ESI = size (rows, colums)
                                <1>       ;         (HW = rows, SI = columns)
                                <1>       ;
                                <1>       ;  [maskcolor] = mask color (to be excluded)
                                <1>       ;
                                <1>       ; OUTPUT:
                                <1>       ;      [u.r0] will be > 0 if succesful
                                <1>
0000EB2A F605[88120300]10       <1>       test   byte [v_ops], 10h ; display page or window ?
0000EB31 7555                   <1>       jnz    short pix_op_or_w ; window
                                <1>
0000EB33 8B3D[8A120300]         <1>       mov    edi, [v_mem]
0000EB39 89FE                   <1>       mov    esi, edi
                                <1>       ; ecx = color (CL, CX, ECX)
0000EB3B 89C8                   <1>       mov    eax, ecx
0000EB3D 8B0D[8E120300]         <1>       mov    ecx, [v_siz] ; display page pixel count
                                <1>
0000EB43 F605[88120300]20       <1>       test   byte [v_ops], 20h ; masked color 'or' ?
0000EB4A 7405                   <1>       jz     short pix_op_or_0 ; no
0000EB4C E947110000             <1>       jmp    m_pix_op_or ; 'or' color except mask color
                                <1> pix_op_or_0:
0000EB51 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000EB58 7707                   <1>       ja     short pix_op_or_1
                                <1>
                                <1>       ; 256 colors (8bpp)
0000EB5A E81C080000             <1>       call   pix_op_or_8
0000EB5F EB1E                   <1>       jmp    short pix_op_or_4
                                <1>
                                <1> pix_op_or_1:
0000EB61 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000EB68 7710                   <1>       ja     short pix_op_or_3 ; 32bpp
0000EB6A 7207                   <1>       jb     short pix_op_or_2 ; 16bpp
                                <1>
                                <1>       ; 24 bit true colors
0000EB6C E818080000             <1>       call   pix_op_or_24
0000EB71 EB0C                   <1>       jmp    short pix_op_or_4
                                <1>
                                <1>       ; 65536 colors (16bpp)
                                <1> pix_op_or_2:
0000EB73 E809080000             <1>       call   pix_op_or_16
0000EB78 EB05                   <1>       jmp    short pix_op_or_4
                                <1>
                                <1>       ; 32 bit true colors
                                <1> pix_op_or_3:
0000EB7A E819080000             <1>       call   pix_op_or_32
                                <1> pix_op_or_4:
0000EB7F 29F7                   <1>       sub    edi, esi
0000EB81 893D[64030300]         <1>       mov    [u.r0], edi
                                <1> pix_op_or_5:
0000EB87 C3                     <1>       retn
                                <1>
                                <1> pix_op_or_w:
                                <1>       ; 31/01/2021
0000EB88 51                     <1>       push   ecx ; * ; color
0000EB89 89D1                   <1>       mov    ecx, edx ; win start pos
0000EB8B 89F2                   <1>       mov    edx, esi ; size (rows, cols)
0000EB8D E866FBFFFF             <1>       call   sysvideo_15_12 ; window preparations
0000EB92 58                     <1>       pop    eax ; * ; color
0000EB93 72F2                   <1>       jc     short pix_op_or_5
                                <1>
0000EB95 F605[88120300]20       <1>       test   byte [v_ops], 20h ; masked color 'or' ?
0000EB9C 7405                   <1>       jz     short pix_op_or_w_0 ; no
0000EB9E E982110000             <1>       jmp    m_pix_op_or_w
                                <1>                      ; window 'or' color except mask color
                                <1> pix_op_or_w_0:
                                <1>       ; ecx = bytes per row (to be applied)
                                <1>       ; edx = screen width in bytes
                                <1>       ; ebx = row count
                                <1>       ; eax = color
                                <1>
0000EBA3 8B3D[92120300]         <1>       mov    edi, [v_str]
0000EBA9 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000EBB0 7707                   <1>       ja     short pix_op_or_w_1
                                <1>
                                <1>       ; 256 colors (8bpp)
0000EBB2 BD[7BF30000]           <1>       mov    ebp, pix_op_or_8
0000EBB7 EB1E                   <1>       jmp    short pix_op_or_w_4
                                <1>
                                <1> pix_op_or_w_1:
0000EBB9 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000EBC0 7710                   <1>       ja     short pix_op_or_w_3 ; 32bpp
0000EBC2 7207                   <1>       jb     short pix_op_or_w_2 ; 16bpp
                                <1>
                                <1>       ; 24 bit true colors
0000EBC4 BD[89F30000]           <1>       mov    ebp, pix_op_or_24
0000EBC9 EB0C                   <1>       jmp    short pix_op_or_w_4
                                <1>
                                <1>       ; 65536 colors (16bpp)
                                <1> pix_op_or_w_2:
0000EBCB BD[81F30000]           <1>       mov    ebp, pix_op_or_16
0000EBD0 EB05                   <1>       jmp    short pix_op_or_w_4
                                <1>
                                <1>       ; 32 bit true colors
                                <1> pix_op_or_w_3:
0000EBD2 BD[98F30000]           <1>       mov    ebp, pix_op_or_32
                                <1> pix_op_or_w_4:
0000EBD7 E998FEFFFF             <1>       jmp    pix_op_orc_w_x
                                <1>
                                <1> pix_op_and:
                                <1>       ; 31/01/2021
                                <1>       ; AND COLOR
```

```
                                <1>       ;
                                <1>       ; INPUT:
                                <1>       ;   CL = color (8 bit, 256 colors)
                                <1>       ;  ECX = color (16 bit and true colors)
                                <1>       ;  EDX = start position (row, column)
                                <1>       ;         (HW = row, DX = column)
                                <1>       ;  ESI = size (rows, colums)
                                <1>       ;         (HW = rows, SI = columns)
                                <1>       ;
                                <1>       ;  [maskcolor] = mask color (to be excluded)
                                <1>       ;
                                <1>       ; OUTPUT:
                                <1>       ;      [u.r0] will be > 0 if succesful
                                <1>
0000EBDC F605[88120300]10       <1>       test   byte [v_ops], 10h ; display page or window ?
0000EBE3 7555                   <1>       jnz    short pix_op_and_w ; window
                                <1>
0000EBE5 8B3D[8A120300]         <1>       mov    edi, [v_mem]
0000EBEB 89FE                   <1>       mov    esi, edi
                                <1>       ; ecx = color (CL, CX, ECX)
0000EBED 89C8                   <1>       mov    eax, ecx
0000EBEF 8B0D[8E120300]         <1>       mov    ecx, [v_siz] ; display page pixel count
                                <1>
0000EBF5 F605[88120300]20       <1>       test   byte [v_ops], 20h ; masked color 'and' ?
0000EBFC 7405                   <1>       jz     short pix_op_and_0 ; no
0000EBFE E9D50F0000             <1>       jmp    m_pix_op_and ; 'and' color except mask color
                                <1> pix_op_and_0:
0000EC03 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000EC0A 7707                   <1>       ja     short pix_op_and_1
                                <1>
                                <1>       ; 256 colors (8bpp)
0000EC0C E88F070000             <1>       call   pix_op_and_8
0000EC11 EB1E                   <1>       jmp    short pix_op_and_4
                                <1>
                                <1> pix_op_and_1:
0000EC13 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000EC1A 7710                   <1>       ja     short pix_op_and_3 ; 32bpp
0000EC1C 7207                   <1>       jb     short pix_op_and_2 ; 16bpp
                                <1>
                                <1>       ; 24 bit true colors
0000EC1E E88B070000             <1>       call   pix_op_and_24
0000EC23 EB0C                   <1>       jmp    short pix_op_and_4
                                <1>
                                <1>       ; 65536 colors (16bpp)
                                <1> pix_op_and_2:
0000EC25 E87C070000             <1>       call   pix_op_and_16
0000EC2A EB05                   <1>       jmp    short pix_op_and_4
                                <1>
                                <1>       ; 32 bit true colors
                                <1> pix_op_and_3:
0000EC2C E88C070000             <1>       call   pix_op_and_32
                                <1> pix_op_and_4:
0000EC31 29F7                   <1>       sub    edi, esi
0000EC33 893D[64030300]         <1>       mov    [u.r0], edi
                                <1> pix_op_and_5:
0000EC39 C3                     <1>       retn
                                <1>
                                <1> pix_op_and_w:
                                <1>       ; 31/01/2021
0000EC3A 51                     <1>       push   ecx ; * ; color
0000EC3B 89D1                   <1>       mov    ecx, edx ; win start pos
0000EC3D 89F2                   <1>       mov    edx, esi ; size (rows, cols)
0000EC3F E8B4FAFFFF             <1>       call   sysvideo_15_12 ; window preparations
0000EC44 58                     <1>       pop    eax ; * ; color
0000EC45 72F2                   <1>       jc     short pix_op_and_5
                                <1>
0000EC47 F605[88120300]20       <1>       test   byte [v_ops], 20h ; masked color 'and' ?
0000EC4E 7405                   <1>       jz     short pix_op_and_w_0 ; no
0000EC50 E910100000             <1>       jmp    m_pix_op_and_w
                                <1>                    ; window 'and' color except mask color
                                <1> pix_op_and_w_0:
                                <1>       ; ecx = bytes per row (to be applied)
                                <1>       ; edx = screen width in bytes
                                <1>       ; ebx = row count
                                <1>       ; eax = color
                                <1>
0000EC55 8B3D[92120300]         <1>       mov    edi, [v_str]
0000EC5B 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000EC62 7707                   <1>       ja     short pix_op_and_w_1
                                <1>
                                <1>       ; 256 colors (8bpp)
0000EC64 BD[A0F30000]           <1>       mov    ebp, pix_op_and_8
0000EC69 EB1E                   <1>       jmp    short pix_op_and_w_4
                                <1>
                                <1> pix_op_and_w_1:
0000EC6B 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000EC72 7710                   <1>       ja     short pix_op_and_w_3 ; 32bpp
0000EC74 7207                   <1>       jb     short pix_op_and_w_2 ; 16bpp
                                <1>
                                <1>       ; 24 bit true colors
0000EC76 BD[AEF30000]           <1>       mov    ebp, pix_op_and_24
0000EC7B EB0C                   <1>       jmp    short pix_op_and_w_4
                                <1>
                                <1>       ; 65536 colors (16bpp)
                                <1> pix_op_and_w_2:
0000EC7D BD[A6F30000]           <1>       mov    ebp, pix_op_and_16
0000EC82 EB05                   <1>       jmp    short pix_op_and_w_4
                                <1>
                                <1>       ; 32 bit true colors
                                <1> pix_op_and_w_3:
0000EC84 BD[BDF30000]           <1>       mov    ebp, pix_op_and_32
                                <1> pix_op_and_w_4:
0000EC89 E9E6FDFFFF             <1>       jmp    pix_op_and_w_x
                                <1>
                                <1> pix_op_xor:
```

```
                                <1>       ; 31/01/2021
                                <1>       ; XOR COLOR
                                <1>       ;
                                <1>       ; INPUT:
                                <1>       ;   CL = color (8 bit, 256 colors)
                                <1>       ;  ECX = color (16 bit and true colors)
                                <1>       ;  EDX = start position (row, column)
                                <1>       ;        (HW = row, DX = column)
                                <1>       ;  ESI = size (rows, colums)
                                <1>       ;        (HW = rows, SI = columns)
                                <1>       ;
                                <1>       ;  [maskcolor] = mask color (to be excluded)
                                <1>       ;
                                <1>       ; OUTPUT:
                                <1>       ;      [u.r0] will be > 0 if succesful
                                <1>
0000EC8E F605[88120300]10       <1>       test   byte [v_ops], 10h ; display page or window ?
0000EC95 7555                   <1>       jnz    short pix_op_xor_w ; window
                                <1>
0000EC97 8B3D[8A120300]         <1>       mov    edi, [v_mem]
0000EC9D 89FE                   <1>       mov    esi, edi
                                <1>       ; ecx = color (CL, CX, ECX)
0000EC9F 89C8                   <1>       mov    eax, ecx
0000ECA1 8B0D[8E120300]         <1>       mov    ecx, [v_siz] ; display page pixel count
                                <1>
0000ECA7 F605[88120300]20       <1>       test   byte [v_ops], 20h ; masked color 'xor' ?
0000ECAE 7405                   <1>       jz     short pix_op_xor_0 ; no
0000ECB0 E9A3100000             <1>       jmp    m_pix_op_xor ; 'xor' color except mask color
                                <1> pix_op_xor_0:
0000ECB5 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000ECBC 7707                   <1>       ja     short pix_op_xor_1
                                <1>
                                <1>       ; 256 colors (8bpp)
0000ECBE E802070000             <1>       call   pix_op_xor_8
0000ECC3 EB1E                   <1>       jmp    short pix_op_xor_4
                                <1>
                                <1> pix_op_xor_1:
0000ECC5 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000ECCC 7710                   <1>       ja     short pix_op_xor_3 ; 32bpp
0000ECCE 7207                   <1>       jb     short pix_op_xor_2 ; 16bpp
                                <1>
                                <1>       ; 24 bit true colors
0000ECD0 E8FE060000             <1>       call   pix_op_xor_24
0000ECD5 EB0C                   <1>       jmp    short pix_op_xor_4
                                <1>
                                <1>       ; 65536 colors (16bpp)
                                <1> pix_op_xor_2:
0000ECD7 E8EF060000             <1>       call   pix_op_xor_16
0000ECDC EB05                   <1>       jmp    short pix_op_xor_4
                                <1>
                                <1>       ; 32 bit true colors
                                <1> pix_op_xor_3:
0000ECDE E8FF060000             <1>       call   pix_op_xor_32
                                <1> pix_op_xor_4:
0000ECE3 29F7                   <1>       sub    edi, esi
0000ECE5 893D[64030300]         <1>       mov    [u.r0], edi
                                <1> pix_op_xor_5:
0000ECEB C3                     <1>       retn
                                <1>
                                <1> pix_op_xor_w:
                                <1>       ; 31/01/2021
0000ECEC 51                     <1>       push   ecx ; * ; color
0000ECED 89D1                   <1>       mov    ecx, edx ; win start pos
0000ECEF 89F2                   <1>       mov    edx, esi ; size (rows, cols)
0000ECF1 E802FAFFFF             <1>       call   sysvideo_15_12 ; window preparations
0000ECF6 58                     <1>       pop    eax ; * ; color
0000ECF7 72F2                   <1>       jc     short pix_op_xor_5
                                <1>
0000ECF9 F605[88120300]20       <1>       test   byte [v_ops], 20h ; masked color 'xor' ?
0000ED00 7405                   <1>       jz     short pix_op_xor_w_0 ; no
0000ED02 E9DE100000             <1>       jmp    m_pix_op_xor_w
                                <1>                     ; window 'xor' color except mask color
                                <1> pix_op_xor_w_0:
                                <1>       ; ecx = bytes per row (to be applied)
                                <1>       ; edx = screen width in bytes
                                <1>       ; ebx = row count
                                <1>       ; eax = color
                                <1>
0000ED07 8B3D[92120300]         <1>       mov    edi, [v_str]
0000ED0D 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000ED14 7707                   <1>       ja     short pix_op_xor_w_1
                                <1>
                                <1>       ; 256 colors (8bpp)
0000ED16 BD[C5F30000]           <1>       mov    ebp, pix_op_xor_8
0000ED1B EB1E                   <1>       jmp    short pix_op_xor_w_4
                                <1>
                                <1> pix_op_xor_w_1:
0000ED1D 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000ED24 7710                   <1>       ja     short pix_op_xor_w_3 ; 32bpp
0000ED26 7207                   <1>       jb     short pix_op_xor_w_2 ; 16bpp
                                <1>
                                <1>       ; 24 bit true colors
0000ED28 BD[D3F30000]           <1>       mov    ebp, pix_op_xor_24
0000ED2D EB0C                   <1>       jmp    short pix_op_xor_w_4
                                <1>
                                <1>       ; 65536 colors (16bpp)
                                <1> pix_op_xor_w_2:
0000ED2F BD[CBF30000]           <1>       mov    ebp, pix_op_xor_16
0000ED34 EB05                   <1>       jmp    short pix_op_xor_w_4
                                <1>
                                <1>       ; 32 bit true colors
                                <1> pix_op_xor_w_3:
0000ED36 BD[E2F30000]           <1>       mov    ebp, pix_op_xor_32
                                <1> pix_op_xor_w_4:
0000ED3B E934FDFFFF             <1>       jmp    pix_op_xor_w_x
```

```
                                <1>
                                <1> pix_op_new:
                                <1>       ; 31/01/2021
                                <1>       ; 30/01/2021
                                <1>       ; CHANGE COLOR
                                <1>       ;
                                <1>       ; INPUT:
                                <1>       ;   CL = color (8 bit, 256 colors)
                                <1>       ;  ECX = color (16 bit and true colors)
                                <1>       ;  EDX = start position (row, column)
                                <1>       ;        (HW = row, DX = column)
                                <1>       ;  ESI = size (rows, colums)
                                <1>       ;        (HW = rows, SI = columns)
                                <1>       ;
                                <1>       ;  [maskcolor] = mask color (to be excluded)
                                <1>       ;
                                <1>       ; OUTPUT:
                                <1>       ;      [u.r0] will be > 0 if succesful
                                <1>
0000ED40 F605[88120300]10       <1>       test   byte [v_ops], 10h ; display page or window ?
0000ED47 7554                   <1>       jnz    short pix_op_new_w ; window
                                <1>
0000ED49 8B3D[8A120300]         <1>       mov    edi, [v_mem]
0000ED4F 89FE                   <1>       mov    esi, edi
                                <1>       ; ecx = color (CL, CX, ECX)
0000ED51 89C8                   <1>       mov    eax, ecx
0000ED53 8B0D[8E120300]         <1>       mov    ecx, [v_siz] ; display page pixel count
                                <1>
0000ED59 F605[88120300]20       <1>       test   byte [v_ops], 20h ; masked color change ?
0000ED60 7405                   <1>       jz     short pix_op_new_0 ; no
0000ED62 E90C0B0000             <1>       jmp    m_pix_op_new ; change color except mask color
                                <1> pix_op_new_0:
0000ED67 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000ED6E 7706                   <1>       ja     short pix_op_new_2
                                <1>
                                <1>       ; 256 colors (8bpp)
                                <1> pix_op_new_1:
0000ED70 88C4                   <1>       mov    ah, al
0000ED72 D1E9                   <1>       shr    ecx, 1
0000ED74 EB12                   <1>       jmp    short pix_op_new_3
                                <1>
                                <1> pix_op_new_2:
0000ED76 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000ED7D 7713                   <1>       ja     short pix_op_new_4 ; 32bpp
0000ED7F 7207                   <1>       jb     short pix_op_new_3 ; 16bpp
                                <1>
                                <1>       ; 31/01/2021
                                <1>
                                <1>       ; 24 bit true colors
0000ED81 E849050000             <1>       call   pix_op_new_24
                                <1>
0000ED86 EB0C                   <1>       jmp    short pix_op_new_5
                                <1>
                                <1>       ; 65536 colors (16bpp)
                                <1> pix_op_new_3:
0000ED88 89C2                   <1>       mov    edx, eax
0000ED8A C1E010                 <1>       shl    eax, 16
0000ED8D 6689D0                 <1>       mov    ax, dx
0000ED90 D1E9                   <1>       shr    ecx, 1 ; dword counts
                                <1>       ; 32 bit true colors
                                <1> pix_op_new_4:
0000ED92 F3AB                   <1>       rep    stosd
                                <1> pix_op_new_5:
0000ED94 29F7                   <1>       sub    edi, esi
0000ED96 893D[64030300]         <1>       mov    [u.r0], edi
                                <1> pix_op_new_6:
0000ED9C C3                     <1>       retn
                                <1>
                                <1> pix_op_new_w:
                                <1>       ; 31/01/2021
                                <1>       ; 30/01/2021
0000ED9D 51                     <1>       push   ecx ; * ; color
0000ED9E 89D1                   <1>       mov    ecx, edx ; win start pos
0000EDA0 89F2                   <1>       mov    edx, esi ; size (rows, cols)
0000EDA2 E851F9FFFF             <1>       call   sysvideo_15_12 ; window preparations
0000EDA7 58                     <1>       pop    eax ; * ; color
0000EDA8 72F2                   <1>       jc     short pix_op_new_6
                                <1>
0000EDAA F605[88120300]20       <1>       test   byte [v_ops], 20h ; masked color change ?
0000EDB1 7405                   <1>       jz     short pix_op_new_w_0 ; no
0000EDB3 E9490B0000             <1>       jmp    m_pix_op_new_w
                                <1>                      ; window chg color except mask color
                                <1> pix_op_new_w_0:
                                <1>       ; ecx = bytes per row (to be applied)
                                <1>       ; edx = screen width in bytes
                                <1>       ; ebx = row count
                                <1>       ; eax = color
                                <1>
0000EDB8 8B3D[92120300]         <1>       mov    edi, [v_str]
                                <1>
0000EDBE 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000EDC5 7707                   <1>       ja     short pix_op_new_w_1
                                <1>
                                <1>       ; 256 colors (8bpp)
0000EDC7 BD[C8F20000]           <1>       mov    ebp, pix_op_new_8
0000EDCC EB1E                   <1>       jmp    short pix_op_new_w_x
                                <1>
                                <1> pix_op_new_w_1:
0000EDCE 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000EDD5 7710                   <1>       ja     short pix_op_new_w_3 ; 32bpp
0000EDD7 7207                   <1>       jb     short pix_op_new_w_2 ; 16bpp
                                <1>
                                <1>       ; 24 bit true colors
0000EDD9 BD[CFF20000]           <1>       mov    ebp, pix_op_new_24
0000EDDE EB0C                   <1>       jmp    short pix_op_new_w_x
```

```
5202                                    <1>
5203                                    <1>     ; 65536 colors (16bpp)
5204                                    <1> pix_op_new_w_2:
5205 0000EDE0 BD[CBF20000]              <1>     mov   ebp, pix_op_new_16
5206 0000EDE5 EB05                      <1>     jmp   short pix_op_new_w_x
5207                                    <1>
5208                                    <1>     ; 32 bit true colors
5209                                    <1> pix_op_new_w_3:
5210 0000EDE7 BD[DBF20000]              <1>     mov   ebp, pix_op_new_32
5211                                    <1>     ;jmp  short pix_op_new_w_x
5212                                    <1>
5213                                    <1> pix_op_new_w_x:
5214                                    <1> pix_op_not_w_x:
5215                                    <1> pix_op_neg_w_x:
5216                                    <1> pix_op_inc_w_x:
5217                                    <1> pix_op_dec_w_x:
5218                                    <1>     ; 01/02/2021
5219                                    <1>     ; 31/01/2021
5220                                    <1>     ; ecx = bytes per row (to be applied)
5221                                    <1>     ; edx = windows (screen) width in bytes
5222                                    <1>     ; ebx = row count
5223                                    <1>     ; eax = color
5224                                    <1>     ; ebp = pixel operation subroutine address
5225                                    <1>     ;push  edx ; 01/02/2021
5226 0000EDEC 51                       <1>     push  ecx
5227 0000EDED 57                       <1>     push  edi
5228 0000EDEE FFD5                     <1>     call  ebp ; call pixel-row operation
5229 0000EDF0 5F                       <1>     pop   edi
5230 0000EDF1 59                       <1>     pop   ecx ; bytes per row
5231 0000EDF2 010D[64030300]           <1>     add   [u.r0], ecx
5232                                    <1>     ;pop  edx ; 01/02/2021
5233 0000EDF8 01D7                     <1>     add   edi, edx ; next row
5234 0000EDFA 4B                       <1>     dec   ebx
5235 0000EDFB 75EF                     <1>     jnz   short pix_op_new_w_x
5236 0000EDFD C3                       <1>     retn
5237                                    <1>
5238                                    <1> pix_op_not:
5239                                    <1>     ; 31/01/2021
5240                                    <1>     ; NOT COLOR
5241                                    <1>     ;
5242                                    <1>     ; INPUT:
5243                                    <1>     ;  ECX = start position (row, column)
5244                                    <1>     ;       (HW = row, CX = column)
5245                                    <1>     ;  EDX = size (rows, colums)
5246                                    <1>     ;       (HW = rows, DX = columns)
5247                                    <1>     ;      (0 -> invalid
5248                                    <1>     ;       (1 -> horizontal or vertical line)
5249                                    <1>     ;  [maskcolor] = mask color (to be excluded)
5250                                    <1>     ;
5251                                    <1>     ; OUTPUT:
5252                                    <1>     ;     [u.r0] will be > 0 if succesful
5253                                    <1>
5254 0000EDFE F605[88120300]10         <1>     test  byte [v_ops], 10h ; display page or window ?
5255 0000EE05 7553                     <1>     jnz   short pix_op_not_w ; window
5256                                    <1>
5257 0000EE07 8B3D[8A120300]           <1>     mov   edi, [v_mem]
5258 0000EE0D 89FE                     <1>     mov   esi, edi
5259 0000EE0F 8B0D[8E120300]           <1>     mov   ecx, [v_siz] ; display page pixel count
5260                                    <1>
5261 0000EE15 F605[88120300]20         <1>     test  byte [v_ops], 20h ; masked color 'not' ?
5262 0000EE1C 7405                     <1>     jz    short pix_op_not_0 ; no
5263 0000EE1E E9F50F0000               <1>     jmp   m_pix_op_not ; 'not' color except mask color
5264                                    <1> pix_op_not_0:
5265 0000EE23 803D[89120300]08         <1>     cmp   byte [v_bpp], 8 ; 8bpp
5266 0000EE2A 7707                     <1>     ja    short pix_op_not_1
5267                                    <1>
5268                                    <1>     ; 256 colors (8bpp)
5269 0000EE2C E8FC050000               <1>     call  pix_op_not_8
5270 0000EE31 EB1E                     <1>     jmp   short pix_op_not_4
5271                                    <1>
5272                                    <1> pix_op_not_1:
5273 0000EE33 803D[89120300]18         <1>     cmp   byte [v_bpp], 24 ; 24bpp
5274 0000EE3A 7710                     <1>     ja    short pix_op_not_3 ; 32bpp
5275 0000EE3C 7207                     <1>     jb    short pix_op_not_2 ; 16bpp
5276                                    <1>
5277                                    <1>     ; 24 bit true colors
5278 0000EE3E E8F8050000               <1>     call  pix_op_not_24
5279 0000EE43 EB0C                     <1>     jmp   short pix_op_not_4
5280                                    <1>
5281                                    <1>     ; 65536 colors (16bpp)
5282                                    <1> pix_op_not_2:
5283 0000EE45 E8E9050000               <1>     call  pix_op_not_16
5284 0000EE4A EB05                     <1>     jmp   short pix_op_not_4
5285                                    <1>
5286                                    <1>     ; 32 bit true colors
5287                                    <1> pix_op_not_3:
5288 0000EE4C E8F5050000               <1>     call  pix_op_not_32
5289                                    <1> pix_op_not_4:
5290 0000EE51 29F7                     <1>     sub   edi, esi
5291 0000EE53 893D[64030300]           <1>     mov   [u.r0], edi
5292                                    <1> pix_op_not_5:
5293 0000EE59 C3                       <1>     retn
5294                                    <1>
5295                                    <1> pix_op_not_w:
5296                                    <1>     ; 31/01/2021
5297                                    <1>     ; ecx = win start pos (row, column)
5298                                    <1>     ; edx = size (rows, columns)
5299 0000EE5A E899F8FFFF               <1>     call  sysvideo_15_12 ; window preparations
5300 0000EE5F 72F8                     <1>     jc    short pix_op_not_5
5301                                    <1>
5302 0000EE61 F605[88120300]20         <1>     test  byte [v_ops], 20h ; masked color 'not' ?
5303 0000EE68 7405                     <1>     jz    short pix_op_not_w_0 ; no
5304 0000EE6A E92E100000               <1>     jmp   m_pix_op_not_w
5305                                    <1>                   ; window 'not' color except mask color
5306                                    <1> pix_op_not_w_0:
```

```
                            <1>       ; ecx = bytes per row (to be applied)
                            <1>       ; edx = screen width in bytes
                            <1>       ; ebx = row count
                            <1>
0000EE6F 8B3D[92120300]     <1>       mov    edi, [v_str]
                            <1>
0000EE75 803D[89120300]08   <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000EE7C 7707               <1>       ja     short pix_op_not_w_1
                            <1>
                            <1>       ; 256 colors (8bpp)
0000EE7E BD[2DF40000]       <1>       mov    ebp, pix_op_not_8
0000EE83 EB1E               <1>       jmp    short pix_op_not_w_4
                            <1>
                            <1> pix_op_not_w_1:
0000EE85 803D[89120300]18   <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000EE8C 7710               <1>       ja     short pix_op_not_w_3 ; 32bpp
0000EE8E 7207               <1>       jb     short pix_op_not_w_2 ; 16bpp
                            <1>
                            <1>       ; 24 bit true colors
0000EE90 BD[3BF40000]       <1>       mov    ebp, pix_op_not_24
0000EE95 EB0C               <1>       jmp    short pix_op_not_w_4
                            <1>
                            <1>       ; 65536 colors (16bpp)
                            <1> pix_op_not_w_2:
0000EE97 BD[33F40000]       <1>       mov    ebp, pix_op_not_16
0000EE9C EB05               <1>       jmp    short pix_op_not_w_4
                            <1>
                            <1>       ; 32 bit true colors
                            <1> pix_op_not_w_3:
0000EE9E BD[46F40000]       <1>       mov    ebp, pix_op_not_32
                            <1> pix_op_not_w_4:
0000EEA3 E944FFFFFF         <1>       jmp    pix_op_not_w_x
                            <1>
                            <1> pix_op_neg:
                            <1>       ; 31/01/2021
                            <1>       ; NEGATE COLOR
                            <1>       ;
                            <1>       ; INPUT:
                            <1>       ;  ECX = start position (row, column)
                            <1>       ;         (HW = row, CX = column)
                            <1>       ;  EDX = size (rows, colums)
                            <1>       ;         (HW = rows, DX = columns)
                            <1>       ;       (0 -> invalid
                            <1>       ;         (1 -> horizontal or vertical line)
                            <1>       ;  [maskcolor] = mask color (to be excluded)
                            <1>       ;
                            <1>       ; OUTPUT:
                            <1>       ;      [u.r0] will be > 0 if succesful
                            <1>
0000EEA8 F605[88120300]10   <1>       test   byte [v_ops], 10h ; display page or window ?
0000EEAF 7553               <1>       jnz    short pix_op_neg_w ; window
                            <1>
0000EEB1 8B3D[8A120300]     <1>       mov    edi, [v_mem]
0000EEB7 89FE               <1>       mov    esi, edi
0000EEB9 8B0D[8E120300]     <1>       mov    ecx, [v_siz] ; display page pixel count
                            <1>
0000EEBF F605[88120300]20   <1>       test   byte [v_ops], 20h ; masked negate color ?
0000EEC6 7405               <1>       jz     short pix_op_neg_0 ; no
0000EEC8 E903100000         <1>       jmp    m_pix_op_neg ; 'neg' color except mask color
                            <1> pix_op_neg_0:
0000EECD 803D[89120300]08   <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000EED4 7707               <1>       ja     short pix_op_neg_1
                            <1>
                            <1>       ; 256 colors (8bpp)
0000EED6 E873050000         <1>       call   pix_op_neg_8
0000EEDB EB1E               <1>       jmp    short pix_op_neg_4
                            <1>
                            <1> pix_op_neg_1:
0000EEDD 803D[89120300]18   <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000EEE4 7710               <1>       ja     short pix_op_neg_3 ; 32bpp
0000EEE6 7207               <1>       jb     short pix_op_neg_2 ; 16bpp
                            <1>
                            <1>       ; 24 bit true colors
0000EEE8 E86F050000         <1>       call   pix_op_neg_24
0000EEED EB0C               <1>       jmp    short pix_op_neg_4
                            <1>
                            <1>       ; 65536 colors (16bpp)
                            <1> pix_op_neg_2:
0000EEEF E860050000         <1>       call   pix_op_neg_16
0000EEF4 EB05               <1>       jmp    short pix_op_neg_4
                            <1>
                            <1>       ; 32 bit true colors
                            <1> pix_op_neg_3:
0000EEF6 E873050000         <1>       call   pix_op_neg_32
                            <1> pix_op_neg_4:
0000EEFB 29F7               <1>       sub    edi, esi
0000EEFD 893D[64030300]     <1>       mov    [u.r0], edi
                            <1> pix_op_neg_5:
0000EF03 C3                 <1>       retn
                            <1>
                            <1> pix_op_neg_w:
                            <1>       ; 31/01/2021
                            <1>       ; ecx = win start pos (row, column)
                            <1>       ; edx = size (rows, columns)
0000EF04 E8EFF7FFFF         <1>       call   sysvideo_15_12 ; window preparations
0000EF09 72F8               <1>       jc     short pix_op_neg_5
                            <1>
0000EF0B F605[88120300]20   <1>       test   byte [v_ops], 20h ; masked negate color ?
0000EF12 7405               <1>       jz     short pix_op_neg_w_0 ; no
0000EF14 E93C100000         <1>       jmp    m_pix_op_neg_w
                            <1>                      ; window 'neg' color except mask color
                            <1> pix_op_neg_w_0:
                            <1>       ; ecx = bytes per row (to be applied)
                            <1>       ; edx = screen width in bytes
                            <1>       ; ebx = row count
```

```
                                <1>
0000EF19 8B3D[92120300]         <1>       mov    edi, [v_str]
                                <1>
0000EF1F 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000EF26 7707                   <1>       ja     short pix_op_neg_w_1
                                <1>
                                <1>       ; 256 colors (8bpp)
0000EF28 BD[4EF40000]           <1>       mov    ebp, pix_op_neg_8
0000EF2D EB1E                   <1>       jmp    short pix_op_neg_w_4
                                <1>
                                <1> pix_op_neg_w_1:
0000EF2F 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000EF36 7710                   <1>       ja     short pix_op_neg_w_3 ; 32bpp
0000EF38 7207                   <1>       jb     short pix_op_neg_w_2 ; 16bpp
                                <1>
                                <1>       ; 24 bit true colors
0000EF3A BD[5CF40000]           <1>       mov    ebp, pix_op_neg_24
0000EF3F EB0C                   <1>       jmp    short pix_op_neg_w_4
                                <1>
                                <1>       ; 65536 colors (16bpp)
                                <1> pix_op_neg_w_2:
0000EF41 BD[54F40000]           <1>       mov    ebp, pix_op_neg_16
0000EF46 EB05                   <1>       jmp    short pix_op_neg_w_4
                                <1>
                                <1>       ; 32 bit true colors
                                <1> pix_op_neg_w_3:
0000EF48 BD[6EF40000]           <1>       mov    ebp, pix_op_neg_32
                                <1> pix_op_neg_w_4:
0000EF4D E99AFEFFFF             <1>       jmp    pix_op_neg_w_x
                                <1>
                                <1> pix_op_inc:
                                <1>       ; 31/01/2021
                                <1>       ; INCREASE COLOR
                                <1>       ;
                                <1>       ; INPUT:
                                <1>       ;  ECX = start position (row, column)
                                <1>       ;         (HW = row, CX = column)
                                <1>       ;  EDX = size (rows, colums)
                                <1>       ;         (HW = rows, DX = columns)
                                <1>       ;       (0 -> invalid
                                <1>       ;         (1 -> horizontal or vertical line)
                                <1>       ;  [maskcolor] = mask color (to be excluded)
                                <1>       ;
                                <1>       ; OUTPUT:
                                <1>       ;      [u.r0] will be > 0 if succesful
                                <1>
0000EF52 F605[88120300]10       <1>       test   byte [v_ops], 10h ; display page or window ?
0000EF59 7553                   <1>       jnz    short pix_op_inc_w ; window
                                <1>
0000EF5B 8B3D[8A120300]         <1>       mov    edi, [v_mem]
0000EF61 89FE                   <1>       mov    esi, edi
0000EF63 8B0D[8E120300]         <1>       mov    ecx, [v_siz] ; display page pixel count
                                <1>
0000EF69 F605[88120300]20       <1>       test   byte [v_ops], 20h ; masked increase color ?
0000EF70 7405                   <1>       jz     short pix_op_inc_0 ; no
0000EF72 E911100000             <1>       jmp    m_pix_op_inc ; 'inc' color except mask color
                                <1> pix_op_inc_0:
0000EF77 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000EF7E 7707                   <1>       ja     short pix_op_inc_1
                                <1>
                                <1>       ; 256 colors (8bpp)
0000EF80 E8F1040000             <1>       call   pix_op_inc_8
0000EF85 EB1E                   <1>       jmp    short pix_op_inc_4
                                <1>
                                <1> pix_op_inc_1:
0000EF87 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000EF8E 7710                   <1>       ja     short pix_op_inc_3 ; 32bpp
0000EF90 7207                   <1>       jb     short pix_op_inc_2 ; 16bpp
                                <1>
                                <1>       ; 24 bit true colors
0000EF92 E8F6040000             <1>       call   pix_op_inc_24
0000EF97 EB0C                   <1>       jmp    short pix_op_inc_4
                                <1>
                                <1>       ; 65536 colors (16bpp)
                                <1> pix_op_inc_2:
0000EF99 E8E2040000             <1>       call   pix_op_inc_16
0000EF9E EB05                   <1>       jmp    short pix_op_inc_4
                                <1>
                                <1>       ; 32 bit true colors
                                <1> pix_op_inc_3:
0000EFA0 E8FC040000             <1>       call   pix_op_inc_32
                                <1> pix_op_inc_4:
0000EFA5 29F7                   <1>       sub    edi, esi
0000EFA7 893D[64030300]         <1>       mov    [u.r0], edi
                                <1> pix_op_inc_5:
0000EFAD C3                     <1>       retn
                                <1>
                                <1> pix_op_inc_w:
                                <1>       ; 31/01/2021
                                <1>       ; ecx = win start pos (row, column)
                                <1>       ; edx = size (rows, columns)
0000EFAE E845F7FFFF             <1>       call   sysvideo_15_12 ; window preparations
0000EFB3 72F8                   <1>       jc     short pix_op_inc_5
                                <1>
0000EFB5 F605[88120300]20       <1>       test   byte [v_ops], 20h ; masked increase color ?
0000EFBC 7405                   <1>       jz     short pix_op_inc_w_0 ; no
0000EFBE E95E100000             <1>       jmp    m_pix_op_inc_w
                                <1>                      ; window 'inc' color except mask color
                                <1> pix_op_inc_w_0:
                                <1>       ; ecx = bytes per row (to be applied)
                                <1>       ; edx = screen width in bytes
                                <1>       ; ebx = row count
                                <1>
0000EFC3 8B3D[92120300]         <1>       mov    edi, [v_str]
                                <1>
```

```
0000EFC9 803D[89120300]08    <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000EFD0 7707                <1>       ja     short pix_op_inc_w_1
                             <1>
                             <1>       ; 256 colors (8bpp)
0000EFD2 BD[76F40000]        <1>       mov    ebp, pix_op_inc_8
0000EFD7 EB1E                <1>       jmp    short pix_op_inc_w_4
                             <1>
                             <1> pix_op_inc_w_1:
0000EFD9 803D[89120300]18    <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000EFE0 7710                <1>       ja     short pix_op_inc_w_3 ; 32bpp
0000EFE2 7207                <1>       jb     short pix_op_inc_w_2 ; 16bpp
                             <1>
                             <1>       ; 24 bit true colors
0000EFE4 BD[8DF40000]        <1>       mov    ebp, pix_op_inc_24
0000EFE9 EB0C                <1>       jmp    short pix_op_inc_w_4
                             <1>
                             <1>       ; 65536 colors (16bpp)
                             <1> pix_op_inc_w_2:
0000EFEB BD[80F40000]        <1>       mov    ebp, pix_op_inc_16
0000EFF0 EB05                <1>       jmp    short pix_op_inc_w_4
                             <1>
                             <1>       ; 32 bit true colors
                             <1> pix_op_inc_w_3:
0000EFF2 BD[A1F40000]        <1>       mov    ebp, pix_op_inc_32
                             <1> pix_op_inc_w_4:
0000EFF7 E9F0FDFFFF          <1>       jmp    pix_op_inc_w_x
                             <1>
                             <1> pix_op_dec:
                             <1>       ; 31/01/2021
                             <1>       ; DECREASE COLOR
                             <1>       ;
                             <1>       ; INPUT:
                             <1>       ;  ECX = start position (row, column)
                             <1>       ;         (HW = row, CX = column)
                             <1>       ;  EDX = size (rows, colums)
                             <1>       ;         (HW = rows, DX = columns)
                             <1>       ;       (0 -> invalid
                             <1>       ;         (1 -> horizontal or vertical line)
                             <1>       ;  [maskcolor] = mask color (to be excluded)
                             <1>       ;
                             <1>       ; OUTPUT:
                             <1>       ;      [u.r0] will be > 0 if succesful
                             <1>
0000EFFC F605[88120300]10    <1>       test   byte [v_ops], 10h ; display page or window ?
0000F003 7553                <1>       jnz    short pix_op_dec_w ; window
                             <1>
0000F005 8B3D[8A120300]      <1>       mov    edi, [v_mem]
0000F00B 89FE                <1>       mov    esi, edi
0000F00D 8B0D[8E120300]      <1>       mov    ecx, [v_siz] ; display page pixel count
                             <1>
0000F013 F605[88120300]20    <1>       test   byte [v_ops], 20h ; masked decrease color ?
0000F01A 7405                <1>       jz     short pix_op_dec_0 ; no
0000F01C E933100000          <1>       jmp    m_pix_op_dec ; 'dec' color except mask color
                             <1> pix_op_dec_0:
0000F021 803D[89120300]08    <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000F028 7707                <1>       ja     short pix_op_dec_1
                             <1>
                             <1>       ; 256 colors (8bpp)
0000F02A E87E040000          <1>       call   pix_op_dec_8
0000F02F EB1E                <1>       jmp    short pix_op_dec_4
                             <1>
                             <1> pix_op_dec_1:
0000F031 803D[89120300]18    <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000F038 7710                <1>       ja     short pix_op_dec_3 ; 32bpp
0000F03A 7207                <1>       jb     short pix_op_dec_2 ; 16bpp
                             <1>
                             <1>       ; 24 bit true colors
0000F03C E883040000          <1>       call   pix_op_dec_24
0000F041 EB0C                <1>       jmp    short pix_op_dec_4
                             <1>
                             <1>       ; 65536 colors (16bpp)
                             <1> pix_op_dec_2:
0000F043 E86F040000          <1>       call   pix_op_dec_16
0000F048 EB05                <1>       jmp    short pix_op_dec_4
                             <1>
                             <1>       ; 32 bit true colors
                             <1> pix_op_dec_3:
0000F04A E888040000          <1>       call   pix_op_dec_32
                             <1> pix_op_dec_4:
0000F04F 29F7                <1>       sub    edi, esi
0000F051 893D[64030300]      <1>       mov    [u.r0], edi
                             <1> pix_op_dec_5:
0000F057 C3                  <1>       retn
                             <1>
                             <1> pix_op_dec_w:
                             <1>       ; 31/01/2021
                             <1>       ; ecx = win start pos (row, column)
                             <1>       ; edx = size (rows, columns)
0000F058 E89BF6FFFF          <1>       call   sysvideo_15_12 ; window preparations
0000F05D 72F8                <1>       jc     short pix_op_dec_5
                             <1>
0000F05F F605[88120300]20    <1>       test   byte [v_ops], 20h ; masked decrease color ?
0000F066 7405                <1>       jz     short pix_op_dec_w_0 ; no
0000F068 E97B100000          <1>       jmp    m_pix_op_dec_w
                             <1>                   ; window 'dec' color except mask color
                             <1> pix_op_dec_w_0:
                             <1>       ; ecx = bytes per row (to be applied)
                             <1>       ; edx = screen width in bytes
                             <1>       ; ebx = row count
                             <1>
0000F06D 8B3D[92120300]      <1>       mov    edi, [v_str]
                             <1>
0000F073 803D[89120300]08    <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000F07A 7707                <1>       ja     short pix_op_dec_w_1
                             <1>
```

```
                                <1>       ; 256 colors (8bpp)
0000F07C BD[ADF40000]           <1>       mov    ebp, pix_op_dec_8
0000F081 EB1E                   <1>       jmp    short pix_op_dec_w_4
                                <1>
                                <1> pix_op_dec_w_1:
0000F083 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000F08A 7710                   <1>       ja     short pix_op_dec_w_3 ; 32bpp
0000F08C 7207                   <1>       jb     short pix_op_dec_w_2 ; 16bpp
                                <1>
                                <1>       ; 24 bit true colors
0000F08E BD[C4F40000]           <1>       mov    ebp, pix_op_dec_24
0000F093 EB0C                   <1>       jmp    short pix_op_dec_w_4
                                <1>
                                <1>       ; 65536 colors (16bpp)
                                <1> pix_op_dec_w_2:
0000F095 BD[B7F40000]           <1>       mov    ebp, pix_op_dec_16
0000F09A EB05                   <1>       jmp    short pix_op_dec_w_4
                                <1>
                                <1>       ; 32 bit true colors
                                <1> pix_op_dec_w_3:
0000F09C BD[D7F40000]           <1>       mov    ebp, pix_op_dec_32
                                <1> pix_op_dec_w_4:
0000F0A1 E946FDFFFF             <1>       jmp    pix_op_dec_w_x
                                <1>
                                <1> pix_op_blk:
                                <1>       ; 02/02/2021
                                <1>       ; COPY PIXEL BLOCK -system to system-
                                <1>       ; WRITE PIXEL BLOCKS -user to system-
                                <1>       ;
                                <1>       ; INPUT:
                                <1>       ;   -If BL bit 5 is 0-
                                <1>       ;    ECX = start position (row, column) (*)
                                <1>       ;    (HW = row, CX = column)
                                <1>       ;    EDX = size (rows, colums) (*)
                                <1>       ;    (HW = rows, DX = columns)
                                <1>       ;         (0 -> invalid)
                                <1>       ;         (1 -> horizontal or vertical line)
                                <1>       ;    ESI = destination (row, column) (***)
                                <1>       ;   -If BL bit 5 is 1-
                                <1>       ;     CL = color (8 bit, 256 colors)
                                <1>       ;    ECX = color (16 bit and true colors)
                                <1>       ;    EDX = count of blocks (not bytes)
                                <1>       ;       (limit: 2048 blocks/windows)
                                <1>       ;    ESI = user's buffer address
                                <1>       ;        contains 64 bit block data
                                <1>       ;        BLOCK ADDRESS - (row, col), dword
                                <1>       ;        (first 32 bits)
                                <1>       ;        BLOCK SIZE - (rows, cols), dword
                                <1>       ;        (second 32 bits)
                                <1>       ; OUTPUT:
                                <1>       ;     [u.r0] will be > 0 if succesful
                                <1>
                                <1>       ; Window option ([v_ops] bit 4) will be ignored
                                <1>       ; (Function is used for display page coordinates)
                                <1>
0000F0A6 F605[88120300]20       <1>       test   byte [v_ops], 20h ; masked or direct ?
0000F0AD 7561                   <1>       jnz    short pix_op_blk_u ; blocks from user's buffer
                                <1>
0000F0AF 89F0                   <1>       mov    eax, esi ; destination position (row, col)
0000F0B1 E8EAF6FFFF             <1>       call   calc_pixel_offset
0000F0B6 3B05[8E120300]         <1>       cmp    eax, [v_siz]
0000F0BC 7351                   <1>       jnb    short pix_op_blk_retn ; out of display page
0000F0BE 89C6                   <1>       mov    esi, eax
0000F0C0 E8B8F6FFFF             <1>       call   pixels_to_byte_count
0000F0C5 89C7                   <1>       mov    edi, eax
0000F0C7 89D0                   <1>       mov    eax, edx  ; size
0000F0C9 E8D2F6FFFF             <1>       call   calc_pixel_offset
0000F0CE 3B05[8E120300]         <1>       cmp    eax, [v_siz]
0000F0D4 7339                   <1>       jnb    short pix_op_blk_retn ; out of display page
0000F0D6 01C6                   <1>       add    esi, eax
0000F0D8 3B35[8E120300]         <1>       cmp    esi, [v_siz]
0000F0DE 732F                   <1>       jnb    short pix_op_blk_retn ; out of display page
0000F0E0 E898F6FFFF             <1>       call   pixels_to_byte_count
0000F0E5 01C7                   <1>       add    edi, eax
0000F0E7 033D[8A120300]         <1>       add    edi, [v_mem] ; destination address
                                <1>       ;cmp   edi, [v_end]
                                <1>       ;jnb   short pix_op_blk_retn ; out of display page
                                <1>
0000F0ED E806F6FFFF             <1>       call   sysvideo_15_12 ; window preparations
0000F0F2 721B                   <1>       jc     short pix_op_blk_retn ; something wrong !?
                                <1>       ; ecx = bytes per row (to be applied)
                                <1>       ; edx = screen width in bytes
                                <1>       ; ebx = row count
                                <1>
0000F0F4 8B35[92120300]         <1>       mov    esi, [v_str] ; source address
                                <1>
                                <1>       ; Note:
                                <1>       ; ecx & edx are already adjusted for pixel sizes
                                <1>       ; so, following code is proper all pixel sizes
                                <1>
0000F0FA 29CA                   <1>       sub    edx, ecx ; screen width - window width
                                <1> pix_op_blk_0:
0000F0FC 89C8                   <1>       mov    eax, ecx
0000F0FE 0105[64030300]         <1>       add    [u.r0], eax
0000F104 F3A4                   <1>       rep    movsb
0000F106 89C1                   <1>       mov    ecx, eax
0000F108 01D6                   <1>       add    esi, edx ; next row
0000F10A 01D7                   <1>       add    edi, edx ; next row
0000F10C 4B                     <1>       dec    ebx
0000F10D 75ED                   <1>       jnz    short pix_op_blk_0
                                <1> pix_op_blk_retn:
0000F10F C3                     <1>       retn
                                <1>
                                <1> pix_op_blk_u:
                                <1>       ; fill blocks (windows) with desired color
```

```
                                <1>       ; according to block definitions in user's buffer
0000F110 81FA00080000           <1>       cmp    edx, 2048
0000F116 7605                   <1>       jna    short pix_op_blk_u_0
                                <1>       ; Maximum 2048 blocks
0000F118 BA00080000             <1>       mov    edx, 2048
                                <1> pix_op_blk_u_0:
0000F11D 8025[88120300]DF       <1>       and    byte [v_ops], ~20h ; clear masked bit
0000F124 890D[9A120300]         <1>       mov    [maskcolor], ecx ; save pixel color
0000F12A 89D5                   <1>       mov    ebp, edx ; save blocks count
                                <1> pix_op_blk_u_next:
0000F12C B908000000             <1>       mov    ecx, 8
0000F131 BF[9E120300]           <1>       mov    edi, buffer8 ; 8 bytes small buffer
                                <1>       ; esi = user's buffer address
0000F136 E8012A0000             <1>       call   transfer_from_user_buffer
0000F13B 72D2                   <1>       jc     short pix_op_blk_retn
0000F13D 56                     <1>       push   esi
0000F13E 8B15[9E120300]         <1>       mov    edx, [buffer8] ; block start pos (row,col)
0000F144 8B35[A2120300]         <1>       mov    esi, [buffer8+4] ; block size (rows,cols)
0000F14A 8B0D[9A120300]         <1>       mov    ecx, [maskcolor]
0000F150 E848FCFFFF             <1>       call   pix_op_new_w ; new (change) color (window)
0000F155 5E                     <1>       pop    esi
0000F156 4D                     <1>       dec    ebp
0000F157 75D3                   <1>       jnz    short pix_op_blk_u_next
0000F159 C3                     <1>       retn
                                <1>
                                <1> pix_op_lin:
                                <1>       ; 12/02/2021
                                <1>       ; 11/02/2021
                                <1>       ; 10/02/2021
                                <1>       ; 05/02/2021
                                <1>       ; 02/02/2021
                                <1>       ; WRITE LINE -direct-
                                <1>       ; WRITE LINE(S) -via user's buffer-
                                <1>       ;
                                <1>       ; INPUT:
                                <1>       ;   -If BL bit 5 is 0-
                                <1>       ;     CL = color (8 bit, 256 colors)
                                <1>       ;    ECX = color (16 bit and true colors)
                                <1>       ;     DX = low 12 bits - size (length)
                                <1>       ;          high 4 bits - direction or type
                                <1>       ;            0 - Horizontal line
                                <1>       ;            1 - Vertical line
                                <1>       ;          > 1 - undefined, invalid
                                <1>       ;    ESI = start position (row, column)
                                <1>       ;         (HW = row, SI = column)
                                <1>       ;   -If BL bit 5 is 1-
                                <1>       ;     CL = color (8 bit, 256 colors)
                                <1>       ;    ECX = color (16 bit and true colors)
                                <1>       ;     DX = number of lines (in user buffer)
                                <1>       ;          (limit: 2048 lines)
                                <1>       ;    ESI = user's buffer
                                <1>       ;           contains 64 bit data for lines
                                <1>       ;          START POINT: 32 bit (row, col)
                                <1>       ;          LENGTH: 32 bit
                                <1>       ;                high 16 bits - 0
                                <1>       ;                bit 0-11 - length
                                <1>       ;                bit 12-15 - type (length)
                                <1>       ; OUTPUT:
                                <1>       ;      [u.r0] will be > 0 if succesful
                                <1>
                                <1>       ; Window option ([v_ops] bit 4) will be ignored
                                <1>       ; (Function is used for display page coordinates)
                                <1>
                                <1>       ; 10/02/2021
0000F15A F605[88120300]20       <1>       test   byte [v_ops], 20h ; masked or direct ?
0000F161 7445                   <1>       jz     short pix_op_lin_vh ; direct (v/h lines)
                                <1>
                                <1>       ; lines from user's buffer
                                <1> pix_op_lin_u:
                                <1>       ; draw lines with desired color
                                <1>       ; according to line definitions in user's buffer
0000F163 81FA00080000           <1>       cmp    edx, 2048
0000F169 7605                   <1>       jna    short pix_op_lin_u_0
                                <1>       ; Maximum 2048 lines
0000F16B BA00080000             <1>       mov    edx, 2048
                                <1> pix_op_lin_u_0:
0000F170 890D[9A120300]         <1>       mov    [maskcolor], ecx ; save pixel color
0000F176 89D5                   <1>       mov    ebp, edx ; save line count
                                <1> pix_op_lin_u_next:
0000F178 B908000000             <1>       mov    ecx, 8
0000F17D BF[9E120300]           <1>       mov    edi, buffer8 ; 8 bytes small buffer
                                <1>       ; esi = user's buffer address
0000F182 E8B5290000             <1>       call   transfer_from_user_buffer
0000F187 721E                   <1>       jc     short pix_op_lin_retn
0000F189 01CE                   <1>       add    esi, ecx ; 11/02/2021
0000F18B 56                     <1>       push   esi
0000F18C 8B35[9E120300]         <1>       mov    esi, [buffer8] ; line start pos (row,col)
0000F192 8B15[A2120300]         <1>       mov    edx, [buffer8+4] ; line length
0000F198 8B0D[9A120300]         <1>       mov    ecx, [maskcolor]
0000F19E E805000000             <1>       call   pix_op_lin_vh ; new (change) color (window)
0000F1A3 5E                     <1>       pop    esi
0000F1A4 4D                     <1>       dec    ebp
0000F1A5 75D1                   <1>       jnz    short pix_op_lin_u_next
                                <1> pix_op_lin_retn:
0000F1A7 C3                     <1>       retn
                                <1>
                                <1> pix_op_lin_vh:
0000F1A8 81FA38140000           <1>       cmp    edx, 1438h ; 1920*1080 (780hx438h) limit
0000F1AE 7761                   <1>       ja     short pix_op_lin_err1 ; invalid type
                                <1>                           ; (for current version)
0000F1B0 66F7C2FF0F             <1>       test   dx, 0FFFh
0000F1B5 745A                   <1>       jz     short pix_op_lin_err1 ; zero length!
                                <1>
0000F1B7 89F0                   <1>       mov    eax, esi ; start point (row, col)
0000F1B9 E8E2F5FFFF             <1>       call   calc_pixel_offset
```

```
5832 0000F1BE 3B05[8E120300]      <1>       cmp    eax, [v_siz]
5833 0000F1C4 734B                <1>       jnb    short pix_op_lin_err1 ; out of display page!
5834 0000F1C6 E8B2F5FFFF          <1>       call   pixels_to_byte_count
5835 0000F1CB 89C7                <1>       mov    edi, eax ; start point offset
5836 0000F1CD 033D[8A120300]      <1>       add    edi, [v_mem] ; LFB start address
5837 0000F1D3 89C8                <1>       mov    eax, ecx ; color
5838                              <1>
5839 0000F1D5 F6C610              <1>       test   dh, 10h
5840 0000F1D8 0F848A000000        <1>       jz     pix_op_lin_h ; Horizontal line
5841                              <1>
5842                              <1> pix_op_lin_v:
5843                              <1>       ; Vertical line
5844 0000F1DE 80E60F              <1>       and    dh, 0Fh ; low 12 bits
5845 0000F1E1 51                  <1>       push   ecx ; color
5846 0000F1E2 89D1                <1>       mov    ecx, edx
5847 0000F1E4 0FB705[86120300]    <1>       movzx  eax, word [v_width]
5848 0000F1EB 89C3                <1>       mov    ebx, eax
5849                              <1>       ; 12/02/2021
5850 0000F1ED F7E2                <1>       mul    edx ; rows * [v_width]
5851 0000F1EF 01F8                <1>       add    eax, edi
5852 0000F1F1 3B05[96120300]      <1>       cmp    eax, [v_end]
5853 0000F1F7 58                  <1>       pop    eax ; color
5854 0000F1F8 7717                <1>       ja     short pix_op_lin_err1 ; out of display page
5855                              <1>       ; ecx = rows
5856 0000F1FA 89CA                <1>       mov    edx, ecx
5857                              <1>
5858 0000F1FC 803D[89120300]08    <1>       cmp    byte [v_bpp], 8 ; 8bpp
5859 0000F203 770D                <1>       ja     short pix_op_lin_v_2
5860                              <1>       ; 256 colors (1 byte per pixel)
5861 0000F205 010D[64030300]      <1>       add    [u.r0], ecx ; byte count
5862                              <1> pix_op_lin_v_1:
5863 0000F20B 8807                <1>       mov    [edi], al
5864 0000F20D 01DF                <1>       add    edi, ebx  ; next row
5865 0000F20F E2FA                <1>       loop   pix_op_lin_v_1
5866                              <1> pix_op_lin_err1:
5867 0000F211 C3                  <1>       retn
5868                              <1>
5869                              <1> pix_op_lin_v_2:
5870 0000F212 803D[89120300]18    <1>       cmp    byte [v_bpp], 24 ; 24bpp
5871 0000F219 773A                <1>       ja     short pix_op_lin_v_6 ; 32bpp
5872 0000F21B 7226                <1>       jb     short pix_op_lin_v_4 ; 16bpp
5873                              <1>
5874                              <1>       ; 24 bit true colors
5875                              <1>       ; * 3
5876 0000F21D 53                  <1>       push   ebx ; screen width in pixels
5877 0000F21E D1E3                <1>       shl    ebx, 1
5878 0000F220 011C24              <1>       add    [esp], ebx
5879 0000F223 5B                  <1>       pop    ebx ; screen width in bytes
5880 0000F224 010D[64030300]      <1>       add    [u.r0], ecx
5881 0000F22A D1E2                <1>       shl    edx, 1
5882 0000F22C 0115[64030300]      <1>       add    [u.r0], edx ; byte count
5883                              <1> pix_op_lin_v_3:
5884 0000F232 668907              <1>       mov    [edi], ax
5885 0000F235 C1C810              <1>       ror    eax, 16
5886 0000F238 884702              <1>       mov    [edi+2], al
5887 0000F23B C1C010              <1>       rol    eax, 16
5888 0000F23E 01DF                <1>       add    edi, ebx  ; next row
5889 0000F240 E2F0                <1>       loop   pix_op_lin_v_3
5890 0000F242 C3                  <1>       retn
5891                              <1>
5892                              <1> pix_op_lin_v_4:
5893                              <1>       ; 16 bit (65536) colors
5894 0000F243 D1E3                <1>       shl    ebx, 1
5895 0000F245 D1E2                <1>       shl    edx, 1
5896 0000F247 0115[64030300]      <1>       add    [u.r0], edx
5897                              <1> pix_op_lin_v_5:
5898 0000F24D 668907              <1>       mov    [edi], ax
5899 0000F250 01DF                <1>       add    edi, ebx  ; next row
5900 0000F252 E2F9                <1>       loop   pix_op_lin_v_5
5901 0000F254 C3                  <1>       retn
5902                              <1>
5903                              <1> pix_op_lin_v_6:
5904                              <1>       ; 32 bit true colors
5905 0000F255 C1E302              <1>       shl    ebx, 2
5906 0000F258 C1E202              <1>       shl    edx, 2
5907 0000F25B 0115[64030300]      <1>       add    [u.r0], edx ; byte count
5908                              <1> pix_op_lin_v_7:
5909 0000F261 8907                <1>       mov    [edi], eax
5910 0000F263 01DF                <1>       add    edi, ebx  ; next row
5911 0000F265 E2FA                <1>       loop   pix_op_lin_v_7
5912 0000F267 C3                  <1>       retn
5913                              <1>
5914                              <1> pix_op_lin_h:
5915                              <1>       ; Horizontal line
5916 0000F268 80E60F              <1>       and    dh, 0Fh ; low 12 bits
5917 0000F26B 89D1                <1>       mov    ecx, edx
5918 0000F26D 6601D6              <1>       add    si, dx ; start column + columns
5919 0000F270 663B35[86120300]    <1>       cmp    si, [v_width] ; screen width
5920 0000F277 7711                <1>       ja     short pix_op_lin_err2 ; out of columns limit
5921                              <1>
5922 0000F279 803D[89120300]08    <1>       cmp    byte [v_bpp], 8 ; 8bpp
5923 0000F280 7709                <1>       ja     short pix_op_lin_h_1
5924                              <1>       ; 256 colors (1 byte per pixel)
5925 0000F282 010D[64030300]      <1>       add    [u.r0], ecx
5926 0000F288 F3AA                <1>       rep    stosb
5927                              <1> pix_op_lin_err2:
5928 0000F28A C3                  <1>       retn
5929                              <1>
5930                              <1> pix_op_lin_h_1:
5931 0000F28B 803D[89120300]18    <1>       cmp    byte [v_bpp], 24 ; 24bpp
5932 0000F292 7728                <1>       ja     short pix_op_lin_h_4 ; 32bpp
5933 0000F294 721A                <1>       jb     short pix_op_lin_h_3 ; 16bpp
5934                              <1>
5935                              <1>       ; 24 bit true colors
5936                              <1>       ; * 3
```

```
5937 0000F296 0115[64030300]      <1>       add    [u.r0], edx
5938 0000F29C D1E2                <1>       shl    edx, 1
5939 0000F29E 0115[64030300]      <1>       add    [u.r0], edx
5940                              <1> pix_op_lin_h_2:
5941 0000F2A4 66AB                <1>       stosw
5942 0000F2A6 C1C810              <1>       ror    eax, 16
5943 0000F2A9 AA                  <1>       stosb
5944 0000F2AA C1C010              <1>       rol    eax, 16
5945 0000F2AD E2F5                <1>       loop   pix_op_lin_h_2
5946 0000F2AF C3                  <1>       retn
5947                              <1>
5948                              <1> pix_op_lin_h_3:
5949                              <1>       ; 16 bit (65536) colors
5950 0000F2B0 D1E2                <1>       shl    edx, 1
5951 0000F2B2 0115[64030300]      <1>       add    [u.r0], edx
5952 0000F2B8 F366AB              <1>       rep    stosw
5953 0000F2BB C3                  <1>       retn
5954                              <1>
5955                              <1> pix_op_lin_h_4:
5956                              <1>       ; 32 bit true colors
5957 0000F2BC C1E202              <1>       shl    edx, 2
5958 0000F2BF 0115[64030300]      <1>       add    [u.r0], edx
5959 0000F2C5 F3AB                <1>       rep    stosd
5960 0000F2C7 C3                  <1>       retn
5961                              <1>
5962                              <1> pix_op_new_8:
5963                              <1>       ; 8 bit colors (256 colors)
5964                              <1>       ; CHANGE PIXEL COLOR
5965                              <1>       ; ecx = pixel count per row
5966                              <1>       ;  al = color
5967                              <1>       ; edi = start pixel address
5968                              <1>
5969 0000F2C8 F3AA                <1>       rep    stosb
5970 0000F2CA C3                  <1>       retn
5971                              <1>
5972                              <1> pix_op_new_16:
5973                              <1>       ; 16 bit colors (65536 colors)
5974                              <1>       ; CHANGE PIXEL COLOR
5975                              <1>       ; ecx = pixel count per row
5976                              <1>       ;  ax = color
5977                              <1>       ; edi = start pixel address
5978                              <1>
5979 0000F2CB F366AB              <1>       rep    stosw
5980 0000F2CE C3                  <1>       retn
5981                              <1>
5982                              <1> pix_op_new_24:
5983                              <1>       ; 24 bit true colors
5984                              <1>       ; CHANGE PIXEL COLOR
5985                              <1>       ; ecx = pixel count per row
5986                              <1>       ; eax = color
5987                              <1>       ; edi = start pixel address
5988                              <1>
5989 0000F2CF 66AB                <1>       stosw
5990 0000F2D1 C1C810              <1>       ror    eax, 16
5991 0000F2D4 AA                  <1>       stosb
5992 0000F2D5 C1C010              <1>       rol    eax, 16
5993 0000F2D8 E2F5                <1>       loop   pix_op_new_24
5994 0000F2DA C3                  <1>       retn
5995                              <1>
5996                              <1> pix_op_new_32:
5997                              <1>       ; 32 bit true colors
5998                              <1>       ; CHANGE PIXEL COLOR
5999                              <1>       ; ecx = pixel count per row
6000                              <1>       ; eax = color
6001                              <1>       ; edi = start pixel address
6002                              <1>
6003 0000F2DB F3AB                <1>       rep    stosd
6004 0000F2DD C3                  <1>       retn
6005                              <1>
6006                              <1> pix_op_add_8:
6007                              <1>       ; 8 bit colors (256 colors)
6008                              <1>       ; ADD PIXEL COLOR
6009                              <1>       ; ecx = pixel count per row
6010                              <1>       ;  al = color
6011                              <1>       ; edi = start pixel address
6012                              <1>
6013 0000F2DE 88C4                <1>       mov    ah, al
6014                              <1> pix_op_add_8_0:
6015 0000F2E0 0207                <1>       add    al, [edi]
6016 0000F2E2 7302                <1>       jnc    short pix_op_add_8_1
6017 0000F2E4 B0FF                <1>       mov    al, 0FFh ; Max. value
6018                              <1> pix_op_add_8_1:
6019 0000F2E6 AA                  <1>       stosb
6020 0000F2E7 88E0                <1>       mov    al, ah
6021 0000F2E9 E2F5                <1>       loop   pix_op_add_8_0
6022 0000F2EB C3                  <1>       retn
6023                              <1>
6024                              <1> pix_op_add_16:
6025                              <1>       ; 16 bit colors (65536 colors)
6026                              <1>       ; ADD PIXEL COLOR
6027                              <1>       ; ecx = pixel count per row
6028                              <1>       ;  ax = color
6029                              <1>       ; edi = start pixel address
6030                              <1>
6031 0000F2EC 89C2                <1>       mov    edx, eax
6032                              <1> pix_op_add_16_0:
6033 0000F2EE 660307              <1>       add    ax, [edi]
6034 0000F2F1 7304                <1>       jnc    short pix_op_add_16_1
6035 0000F2F3 66B8FFFF            <1>       mov    ax, 0FFFFh ; Max. value
6036                              <1> pix_op_add_16_1:
6037 0000F2F7 66AB                <1>       stosw
6038 0000F2F9 89D0                <1>       mov    eax, edx
6039 0000F2FB E2F1                <1>       loop   pix_op_add_16_0
6040 0000F2FD C3                  <1>       retn
6041                              <1>
```

```
                        <1> pix_op_add_24:
                        <1>       ; 24 bit true colors
                        <1>       ; ADD PIXEL COLOR
                        <1>       ; ecx = pixel count per row
                        <1>       ; eax = color
                        <1>       ; edi = start pixel address
                        <1>
0000F2FE 53             <1>       push   ebx
0000F2FF BBFFFFFF00     <1>       mov    ebx, 0FFFFFFh
                        <1>       ;and   eax, ebx ; 0FFFFFFh
0000F304 89C2           <1>       mov    edx, eax
                        <1> pix_op_add_24_0:
0000F306 8B07           <1>       mov    eax, [edi]
0000F308 21D8           <1>       and    eax, ebx ; 0FFFFFFh
0000F30A 01D0           <1>       add    eax, edx
0000F30C 39D8           <1>       cmp    eax, ebx
0000F30E 7602           <1>       jna    short pix_op_add_24_1
0000F310 89D8           <1>       mov    eax, ebx ; 0FFFFFFh  ; Max. value
                        <1> pix_op_add_24_1:
0000F312 66AB           <1>       stosw
0000F314 C1E810         <1>       shr    eax, 16
0000F317 AA             <1>       stosb
0000F318 E2EC           <1>       loop   pix_op_add_24_0
0000F31A 89D0           <1>       mov    eax, edx
0000F31C 5B             <1>       pop    ebx
0000F31D C3             <1>       retn
                        <1>
                        <1> pix_op_add_32:
                        <1>       ; 32 bit true colors
                        <1>       ; ADD PIXEL COLOR
                        <1>       ; ecx = pixel count per row
                        <1>       ; eax = color
                        <1>       ; edi = start pixel address
                        <1>
0000F31E 89C2           <1>       mov    edx, eax
                        <1> pix_op_add_32_0:
0000F320 0307           <1>       add    eax, [edi]
0000F322 7303           <1>       jnc    short pix_op_add_32_1
                        <1>       ;mov   eax, 0FFFFFFFFh ; Max. value
0000F324 29C0           <1>       sub    eax, eax
0000F326 48             <1>       dec    eax
                        <1> pix_op_add_32_1:
0000F327 AB             <1>       stosd
0000F328 89D0           <1>       mov    eax, edx
0000F32A E2F4           <1>       loop   pix_op_add_32_0
0000F32C C3             <1>       retn
                        <1>
                        <1> pix_op_sub_8:
                        <1>       ; 8 bit colors (256 colors)
                        <1>       ; SUBTRACT PIXEL COLOR
                        <1>       ; ecx = pixel count per row
                        <1>       ;  al = color
                        <1>       ; edi = start pixel address
                        <1>
0000F32D 88C4           <1>       mov    ah, al
                        <1> pix_op_sub_8_0:
0000F32F 8A07           <1>       mov    al, [edi]
0000F331 28E0           <1>       sub    al, ah
0000F333 7302           <1>       jnb    short pix_op_sub_8_1
0000F335 30C0           <1>       xor    al, al ; 0 ; Min. value
                        <1> pix_op_sub_8_1:
0000F337 AA             <1>       stosb
0000F338 E2F5           <1>       loop   pix_op_sub_8_0
0000F33A 88E0           <1>       mov    al, ah
0000F33C C3             <1>       retn
                        <1>
                        <1> pix_op_sub_16:
                        <1>       ; 16 bit colors (65536 colors)
                        <1>       ; SUBTRACT PIXEL COLOR
                        <1>       ; ecx = pixel count per row
                        <1>       ;  ax = color
                        <1>       ; edi = start pixel address
                        <1>
0000F33D 89C2           <1>       mov    edx, eax
                        <1> pix_op_sub_16_0:
0000F33F 668B07         <1>       mov    ax, [edi]
0000F342 6629D0         <1>       sub    ax, dx
0000F345 7302           <1>       jnb    short pix_op_sub_16_1
0000F347 31C0           <1>       xor    eax, eax ; 0 ; Min. value
                        <1> pix_op_sub_16_1:
0000F349 66AB           <1>       stosw
0000F34B E2F2           <1>       loop   pix_op_sub_16_0
0000F34D 89D0           <1>       mov    eax, edx
0000F34F C3             <1>       retn
                        <1>
                        <1> pix_op_sub_24:
                        <1>       ; 24 bit true colors
                        <1>       ; SUBTRACT PIXEL COLOR
                        <1>       ; ecx = pixel count per row
                        <1>       ; eax = color
                        <1>       ; edi = start pixel address
                        <1>
                        <1>       ;and   eax, 0FFFFFFh
0000F350 89C2           <1>       mov    edx, eax
                        <1> pix_op_sub_24_0:
0000F352 8B07           <1>       mov    eax, [edi]
0000F354 50             <1>       push   eax
0000F355 25FFFFFF00     <1>       and    eax, 0FFFFFFh
0000F35A 29D0           <1>       sub    eax, edx
0000F35C 7302           <1>       jnb    short pix_op_sub_24_1
0000F35E 31C0           <1>       xor    eax, eax ; 0 ; Min. value
                        <1> pix_op_sub_24_1:
0000F360 66AB           <1>       stosw
0000F362 C1E810         <1>       shr    eax, 16
0000F365 AA             <1>       stosb
```

```
6147 0000F366 E2EA              <1>       loop   pix_op_sub_24_0
6148 0000F368 89D0              <1>       mov    eax, edx
6149 0000F36A C3                <1>       retn
6150                            <1>
6151                            <1> pix_op_sub_32:
6152                            <1>       ; 32 bit true colors
6153                            <1>       ; SUBTRACT PIXEL COLOR
6154                            <1>       ; ecx = pixel count per row
6155                            <1>       ; eax = color
6156                            <1>       ; edi = start pixel address
6157                            <1>
6158 0000F36B 89C2              <1>       mov    edx, eax
6159                            <1> pix_op_sub_32_0:
6160 0000F36D 8B07              <1>       mov    eax, [edi]
6161 0000F36F 29D0              <1>       sub    eax, edx
6162 0000F371 7302              <1>       jnb    short pix_op_sub_32_1
6163 0000F373 31C0              <1>       xor    eax, eax ; 0 ; Min. value
6164                            <1> pix_op_sub_32_1:
6165 0000F375 AB                <1>       stosd
6166 0000F376 E2F5              <1>       loop   pix_op_sub_32_0
6167 0000F378 89D0              <1>       mov    eax, edx
6168 0000F37A C3                <1>       retn
6169                            <1>
6170                            <1> pix_op_or_8:
6171                            <1>       ; 8 bit colors (256 colors)
6172                            <1>       ; OR PIXEL COLOR
6173                            <1>       ; ecx = pixel count per row
6174                            <1>       ;  al = color
6175                            <1>       ; edi = start pixel address
6176                            <1>
6177                            <1> pix_op_or_8_0:
6178 0000F37B 0807              <1>       or     [edi], al
6179 0000F37D 47                <1>       inc    edi
6180 0000F37E E2FB              <1>       loop   pix_op_or_8_0
6181 0000F380 C3                <1>       retn
6182                            <1>
6183                            <1> pix_op_or_16:
6184                            <1>       ; 16 bit colors (65536 colors)
6185                            <1>       ; OR PIXEL COLOR
6186                            <1>       ; ecx = pixel count per row
6187                            <1>       ;  ax = color
6188                            <1>       ; edi = start pixel address
6189                            <1>
6190                            <1> pix_op_or_16_0:
6191 0000F381 660907            <1>       or     [edi], ax
6192 0000F384 47                <1>       inc    edi
6193 0000F385 47                <1>       inc    edi
6194 0000F386 E2F9              <1>       loop   pix_op_or_16_0
6195 0000F388 C3                <1>       retn
6196                            <1>
6197                            <1> pix_op_or_24:
6198                            <1>       ; 24 bit true colors
6199                            <1>       ; OR PIXEL COLOR
6200                            <1>       ; ecx = pixel count per row
6201                            <1>       ; eax = color
6202                            <1>       ; edi = start pixel address
6203                            <1>
6204 0000F389 89C2              <1>       mov    edx, eax
6205                            <1> pix_op_or_24_0:
6206 0000F38B 0B07              <1>       or     eax, [edi]
6207 0000F38D 66AB              <1>       stosw
6208 0000F38F C1E810            <1>       shr    eax, 16
6209 0000F392 AA                <1>       stosb
6210 0000F393 89D0              <1>       mov    eax, edx
6211 0000F395 E2F4              <1>       loop   pix_op_or_24_0
6212 0000F397 C3                <1>       retn
6213                            <1>
6214                            <1> pix_op_or_32:
6215                            <1>       ; 32 bit true colors
6216                            <1>       ; OR PIXEL COLOR
6217                            <1>       ; ecx = pixel count per row
6218                            <1>       ; eax = color
6219                            <1>       ; edi = start pixel address
6220                            <1>
6221                            <1>       ;mov   edx, eax
6222                            <1> pix_op_or_32_0:
6223                            <1>       ;or    eax, [edi]
6224                            <1>       ;stosd
6225                            <1>       ;mov   eax, edx
6226 0000F398 0907              <1>       or     [edi], eax
6227 0000F39A 83C704            <1>       add    edi, 4
6228 0000F39D E2F9              <1>       loop   pix_op_or_32_0
6229 0000F39F C3                <1>       retn
6230                            <1>
6231                            <1> pix_op_and_8:
6232                            <1>       ; 8 bit colors (256 colors)
6233                            <1>       ; AND PIXEL COLOR
6234                            <1>       ; ecx = pixel count per row
6235                            <1>       ;  al = color
6236                            <1>       ; edi = start pixel address
6237                            <1>
6238                            <1> pix_op_and_8_0:
6239 0000F3A0 2007              <1>       and    [edi], al
6240 0000F3A2 47                <1>       inc    edi
6241 0000F3A3 E2FB              <1>       loop   pix_op_and_8_0
6242 0000F3A5 C3                <1>       retn
6243                            <1>
6244                            <1> pix_op_and_16:
6245                            <1>       ; 16 bit colors (65536 colors)
6246                            <1>       ; AND PIXEL COLOR
6247                            <1>       ; ecx = pixel count per row
6248                            <1>       ;  ax = color
6249                            <1>       ; edi = start pixel address
6250                            <1>
6251                            <1> pix_op_and_16_0:
```

```
6252 0000F3A6 662107            <1>       and   [edi], ax
6253 0000F3A9 47                <1>       inc   edi
6254 0000F3AA 47                <1>       inc   edi
6255 0000F3AB E2F9              <1>       loop  pix_op_and_16_0
6256 0000F3AD C3                <1>       retn
6257                            <1>
6258                            <1> pix_op_and_24:
6259                            <1>       ; 24 bit true colors
6260                            <1>       ; AND PIXEL COLOR
6261                            <1>       ; ecx = pixel count per row
6262                            <1>       ; eax = color
6263                            <1>       ; edi = start pixel address
6264                            <1>
6265 0000F3AE 89C2              <1>       mov   edx, eax
6266                            <1> pix_op_and_24_0:
6267 0000F3B0 2307              <1>       and   eax, [edi]
6268 0000F3B2 66AB              <1>       stosw
6269 0000F3B4 C1E810            <1>       shr   eax, 16
6270 0000F3B7 AA                <1>       stosb
6271 0000F3B8 89D0              <1>       mov   eax, edx
6272 0000F3BA E2F4              <1>       loop  pix_op_and_24_0
6273 0000F3BC C3                <1>       retn
6274                            <1>
6275                            <1> pix_op_and_32:
6276                            <1>       ; 32 bit true colors
6277                            <1>       ; AND PIXEL COLOR
6278                            <1>       ; ecx = pixel count per row
6279                            <1>       ; eax = color
6280                            <1>       ; edi = start pixel address
6281                            <1>
6282                            <1>       ;mov  edx, eax
6283                            <1> pix_op_and_32_0:
6284                            <1>       ;and  eax, [edi]
6285                            <1>       ;stosd
6286                            <1>       ;mov  eax, edx
6287 0000F3BD 2107              <1>       and   [edi], eax
6288 0000F3BF 83C704            <1>       add   edi, 4
6289 0000F3C2 E2F9              <1>       loop  pix_op_and_32_0
6290 0000F3C4 C3                <1>       retn
6291                            <1>
6292                            <1> pix_op_xor_8:
6293                            <1>       ; 8 bit colors (256 colors)
6294                            <1>       ; XOR PIXEL COLOR
6295                            <1>       ; ecx = pixel count per row
6296                            <1>       ;  al = color
6297                            <1>       ; edi = start pixel address
6298                            <1>
6299                            <1> pix_op_xor_8_0:
6300 0000F3C5 3007              <1>       xor   [edi], al
6301 0000F3C7 47                <1>       inc   edi
6302 0000F3C8 E2FB              <1>       loop  pix_op_xor_8_0
6303 0000F3CA C3                <1>       retn
6304                            <1>
6305                            <1> pix_op_xor_16:
6306                            <1>       ; 16 bit colors (65536 colors)
6307                            <1>       ; XOR PIXEL COLOR
6308                            <1>       ; ecx = pixel count per row
6309                            <1>       ;  ax = color
6310                            <1>       ; edi = start pixel address
6311                            <1>
6312                            <1> pix_op_xor_16_0:
6313 0000F3CB 663107            <1>       xor   [edi], ax
6314 0000F3CE 47                <1>       inc   edi
6315 0000F3CF 47                <1>       inc   edi
6316 0000F3D0 E2F9              <1>       loop  pix_op_xor_16_0
6317 0000F3D2 C3                <1>       retn
6318                            <1>
6319                            <1> pix_op_xor_24:
6320                            <1>       ; 24 bit true colors
6321                            <1>       ; XOR PIXEL COLOR
6322                            <1>       ; ecx = pixel count per row
6323                            <1>       ; eax = color
6324                            <1>       ; edi = start pixel address
6325                            <1>
6326 0000F3D3 89C2              <1>       mov   edx, eax
6327                            <1> pix_op_xor_24_0:
6328 0000F3D5 3307              <1>       xor   eax, [edi]
6329 0000F3D7 66AB              <1>       stosw
6330 0000F3D9 C1E810            <1>       shr   eax, 16
6331 0000F3DC AA                <1>       stosb
6332 0000F3DD 89D0              <1>       mov   eax, edx
6333 0000F3DF E2F4              <1>       loop  pix_op_xor_24_0
6334 0000F3E1 C3                <1>       retn
6335                            <1>
6336                            <1> pix_op_xor_32:
6337                            <1>       ; 32 bit true colors
6338                            <1>       ; XOR PIXEL COLOR
6339                            <1>       ; ecx = pixel count per row
6340                            <1>       ; eax = color
6341                            <1>       ; edi = start pixel address
6342                            <1>
6343                            <1>       ;mov  edx, eax
6344                            <1> pix_op_xor_32_0:
6345                            <1>       ;xor  eax, [edi]
6346                            <1>       ;stosd
6347                            <1>       ;mov  eax, edx
6348 0000F3E2 3107              <1>       xor   [edi], eax
6349 0000F3E4 83C704            <1>       add   edi, 4
6350 0000F3E7 E2F9              <1>       loop  pix_op_xor_32_0
6351 0000F3E9 C3                <1>       retn
6352                            <1>
6353                            <1> pix_op_mix_8:
6354                            <1>       ; 8 bit colors (256 colors)
6355                            <1>       ; MIX (AVERAGE) PIXEL COLORS
6356                            <1>       ; ecx = pixel count per row
```

```
                        <1>       ;  al = color
                        <1>       ; edi = start pixel address
                        <1>
0000F3EA 88C4           <1>       mov   ah, al
                        <1> pix_op_mix_8_0:
0000F3EC 0207           <1>       add   al, [edi]
0000F3EE D0D8           <1>       rcr   al, 1
0000F3F0 AA             <1>       stosb
0000F3F1 88E0           <1>       mov   al, ah
0000F3F3 E2F7           <1>       loop  pix_op_mix_8_0
0000F3F5 C3             <1>       retn
                        <1>
                        <1> pix_op_mix_16:
                        <1>       ; 16 bit colors (65536 colors)
                        <1>       ; MIX (AVERAGE) PIXEL COLORS
                        <1>       ; ecx = pixel count per row
                        <1>       ;  ax = color
                        <1>       ; edi = start pixel address
                        <1>
0000F3F6 89C2           <1>       mov   edx, eax
                        <1> pix_op_mix_16_0:
0000F3F8 660307         <1>       add   ax, [edi]
0000F3FB 66D1D8         <1>       rcr   ax, 1
0000F3FE 66AB           <1>       stosw
0000F400 89D0           <1>       mov   eax, edx
0000F402 E2F4           <1>       loop  pix_op_mix_16_0
0000F404 C3             <1>       retn
                        <1>
                        <1> pix_op_mix_24:
                        <1>       ; 24 bit true colors
                        <1>       ; MIX (AVERAGE) PIXEL COLORS
                        <1>       ; ecx = pixel count per row
                        <1>       ; eax = color
                        <1>       ; edi = start pixel address
                        <1>
0000F405 53             <1>       push  ebx
0000F406 BBFFFFFF00     <1>       mov   ebx, 0FFFFFFh
                        <1>       ;and  eax, ebx ; 0FFFFFFh
0000F40B 89C2           <1>       mov   edx, eax
                        <1> pix_op_mix_24_0:
0000F40D 8B07           <1>       mov   eax, [edi]
0000F40F 21D8           <1>       and   eax, ebx ; 0FFFFFFh
0000F411 01D0           <1>       add   eax, edx
0000F413 D1E8           <1>       shr   eax, 1
                        <1>       ;rcr  eax, 1
0000F415 66AB           <1>       stosw
0000F417 C1E810         <1>       shr   eax, 16
0000F41A AA             <1>       stosb
0000F41B E2F0           <1>       loop  pix_op_mix_24_0
0000F41D 89D0           <1>       mov   eax, edx
0000F41F 5B             <1>       pop   ebx
0000F420 C3             <1>       retn
                        <1>
                        <1> pix_op_mix_32:
                        <1>       ; 32 bit true colors
                        <1>       ; MIX (AVERAGE) PIXEL COLORS
                        <1>       ; ecx = pixel count per row
                        <1>       ; eax = color
                        <1>       ; edi = start pixel address
                        <1>
0000F421 89C2           <1>       mov   edx, eax
                        <1> pix_op_mix_32_0:
0000F423 0307           <1>       add   eax, [edi]
0000F425 D1D8           <1>       rcr   eax, 1
0000F427 AB             <1>       stosd
0000F428 89D0           <1>       mov   eax, edx
0000F42A E2F7           <1>       loop  pix_op_mix_32_0
0000F42C C3             <1>       retn
                        <1>
                        <1> pix_op_not_8:
                        <1>       ; 8 bit colors (256 colors)
                        <1>       ; NOT PIXEL COLOR
                        <1>       ; ecx = pixel count per row
                        <1>       ; edi = start pixel address
                        <1>
                        <1> pix_op_not_8_0:
0000F42D F617           <1>       not   byte [edi]
0000F42F 47             <1>       inc   edi
0000F430 E2FB           <1>       loop  pix_op_not_8_0
0000F432 C3             <1>       retn
                        <1>
                        <1> pix_op_not_16:
                        <1>       ; 16 bit colors (65536 colors)
                        <1>       ; NOT PIXEL COLOR
                        <1>       ; ecx = pixel count per row
                        <1>       ; edi = start pixel address
                        <1>
                        <1> pix_op_not_16_0:
0000F433 66F717         <1>       not   word [edi]
0000F436 47             <1>       inc   edi
0000F437 47             <1>       inc   edi
0000F438 E2F9           <1>       loop  pix_op_not_16_0
0000F43A C3             <1>       retn
                        <1>
                        <1> pix_op_not_24:
                        <1>       ; 24 bit true colors
                        <1>       ; NOT PIXEL COLOR
                        <1>       ; ecx = pixel count per row
                        <1>       ; edi = start pixel address
                        <1>
                        <1> pix_op_not_24_0:
0000F43B 66F717         <1>       not   word [edi]
0000F43E 47             <1>       inc   edi
0000F43F 47             <1>       inc   edi
0000F440 F617           <1>       not   byte [edi]
```

```
6462 0000F442 47                <1>       inc    edi
6463 0000F443 E2F6              <1>       loop   pix_op_not_24_0
6464 0000F445 C3                <1>       retn
6465                            <1>
6466                            <1> pix_op_not_32:
6467                            <1>       ; 32 bit true colors
6468                            <1>       ; NOT PIXEL COLOR
6469                            <1>       ; ecx = pixel count per row
6470                            <1>       ; eax = color
6471                            <1>       ; edi = start pixel address
6472                            <1> pix_op_not_32_0:
6473 0000F446 F717              <1>       not    dword [edi]
6474 0000F448 83C704            <1>       add    edi, 4
6475 0000F44B E2F9              <1>       loop   pix_op_not_32_0
6476 0000F44D C3                <1>       retn
6477                            <1>
6478                            <1> pix_op_neg_8:
6479                            <1>       ; 8 bit colors (256 colors)
6480                            <1>       ; NEG PIXEL COLOR
6481                            <1>       ; ecx = pixel count per row
6482                            <1>       ; edi = start pixel address
6483                            <1>
6484                            <1> pix_op_neg_8_0:
6485 0000F44E F61F              <1>       neg    byte [edi]
6486 0000F450 47                <1>       inc    edi
6487 0000F451 E2FB              <1>       loop   pix_op_neg_8_0
6488 0000F453 C3                <1>       retn
6489                            <1>
6490                            <1> pix_op_neg_16:
6491                            <1>       ; 16 bit colors (65536 colors)
6492                            <1>       ; NEG PIXEL COLOR
6493                            <1>       ; ecx = pixel count per row
6494                            <1>       ; edi = start pixel address
6495                            <1>
6496                            <1> pix_op_neg_16_0:
6497 0000F454 66F71F            <1>       neg    word [edi]
6498 0000F457 47                <1>       inc    edi
6499 0000F458 47                <1>       inc    edi
6500 0000F459 E2F9              <1>       loop   pix_op_neg_16_0
6501 0000F45B C3                <1>       retn
6502                            <1>
6503                            <1> pix_op_neg_24:
6504                            <1>       ; 24 bit true colors
6505                            <1>       ; NEG PIXEL COLOR
6506                            <1>       ; ecx = pixel count per row
6507                            <1>       ; edi = start pixel address
6508                            <1>
6509                            <1> pix_op_neg_24_0:
6510 0000F45C 8B07              <1>       mov    eax, [edi]
6511 0000F45E 25FFFFFF00        <1>       and    eax, 0FFFFFFh
6512 0000F463 F7D8              <1>       neg    eax
6513 0000F465 66AB              <1>       stosw
6514 0000F467 C1E810            <1>       shr    eax, 16
6515 0000F46A AA                <1>       stosb
6516 0000F46B E2EF              <1>       loop   pix_op_neg_24_0
6517 0000F46D C3                <1>       retn
6518                            <1>
6519                            <1> pix_op_neg_32:
6520                            <1>       ; 32 bit true colors
6521                            <1>       ; NEG PIXEL COLOR
6522                            <1>       ; ecx = pixel count per row
6523                            <1>       ; eax = color
6524                            <1>       ; edi = start pixel address
6525                            <1> pix_op_neg_32_0:
6526 0000F46E F71F              <1>       neg    dword [edi]
6527 0000F470 83C704            <1>       add    edi, 4
6528 0000F473 E2F9              <1>       loop   pix_op_neg_32_0
6529 0000F475 C3                <1>       retn
6530                            <1>
6531                            <1> pix_op_inc_8:
6532                            <1>       ; 8 bit colors (256 colors)
6533                            <1>       ; INCREASE PIXEL COLOR
6534                            <1>       ; ecx = pixel count per row
6535                            <1>       ; edi = start pixel address
6536                            <1>
6537                            <1> pix_op_inc_8_0:
6538 0000F476 FE07              <1>       inc    byte [edi]
6539 0000F478 7502              <1>       jnz    short pix_op_inc_8_1
6540                            <1>       ;mov   [edi], 0FFh ; Max. value
6541 0000F47A FE0F              <1>       dec    byte [edi]
6542                            <1> pix_op_inc_8_1:
6543 0000F47C 47                <1>       inc    edi
6544 0000F47D E2F7              <1>       loop   pix_op_inc_8_0
6545 0000F47F C3                <1>       retn
6546                            <1>
6547                            <1> pix_op_inc_16:
6548                            <1>       ; 16 bit colors (65536 colors)
6549                            <1>       ; INCREASE PIXEL COLOR
6550                            <1>       ; ecx = pixel count per row
6551                            <1>       ; edi = start pixel address
6552                            <1>
6553                            <1> pix_op_inc_16_0:
6554 0000F480 66FF07            <1>       inc    word [edi]
6555 0000F483 7503              <1>       jnz    short pix_op_inc_16_1
6556                            <1>       ;mov   word [edi], 0FFFFh ; Max. value
6557 0000F485 66FF0F            <1>       dec    word [edi]
6558                            <1> pix_op_inc_16_1:
6559 0000F488 47                <1>       inc    edi
6560 0000F489 47                <1>       inc    edi
6561 0000F48A E2F4              <1>       loop   pix_op_inc_16_0
6562 0000F48C C3                <1>       retn
6563                            <1>
6564                            <1> pix_op_inc_24:
6565                            <1>       ; 24 bit true colors
6566                            <1>       ; INCREASE PIXEL COLOR
```

```
                                <1>       ; ecx = pixel count per row
                                <1>       ; edi = start pixel address
                                <1>
                                <1> pix_op_inc_24_0:
0000F48D 8B07                   <1>       mov   eax, [edi]
0000F48F 40                     <1>       inc   eax
0000F490 25FFFFFF00             <1>       and   eax, 0FFFFFFh
0000F495 7501                   <1>       jnz   short pix_op_inc_24_1
                                <1>       ;mov  eax, 0FFFFFFh  ; Max. value
0000F497 48                     <1>       dec   eax ; 0FFFFFFFFh
                                <1> pix_op_inc_24_1:
0000F498 66AB                   <1>       stosw
0000F49A C1E810                 <1>       shr   eax, 16
0000F49D AA                     <1>       stosb
0000F49E E2ED                   <1>       loop  pix_op_inc_24_0
0000F4A0 C3                     <1>       retn
                                <1>
                                <1> pix_op_inc_32:
                                <1>       ; 32 bit true colors
                                <1>       ; INCREASE PIXEL COLOR
                                <1>       ; ecx = pixel count per row
                                <1>       ; edi = start pixel address
                                <1>
                                <1> pix_op_inc_32_0:
0000F4A1 FF07                   <1>       inc   dword [edi]
0000F4A3 7502                   <1>       jnz   short pix_op_inc_32_1
                                <1>       ;mov  dword [edi], 0FFFFFFFFh ; Max. value
0000F4A5 FF0F                   <1>       dec   dword [edi]
                                <1> pix_op_inc_32_1:
0000F4A7 83C704                 <1>       add   edi, 4
0000F4AA E2F5                   <1>       loop  pix_op_inc_32_0
0000F4AC C3                     <1>       retn
                                <1>
                                <1> pix_op_dec_8:
                                <1>       ; 8 bit colors (256 colors)
                                <1>       ; DECREASE PIXEL COLOR
                                <1>       ; ecx = pixel count per row
                                <1>       ; edi = start pixel address
                                <1>
                                <1> pix_op_dec_8_0:
0000F4AD FE0F                   <1>       dec   byte [edi]
0000F4AF 7902                   <1>       jns   short pix_op_dec_8_1
0000F4B1 FE07                   <1>       inc   byte [edi] ; 0 ; Min. value
                                <1> pix_op_dec_8_1:
0000F4B3 47                     <1>       inc   edi
0000F4B4 E2F7                   <1>       loop  pix_op_dec_8_0
0000F4B6 C3                     <1>       retn
                                <1>
                                <1> pix_op_dec_16:
                                <1>       ; 16 bit colors (65536 colors)
                                <1>       ; DECREASE PIXEL COLOR
                                <1>       ; ecx = pixel count per row
                                <1>       ; edi = start pixel address
                                <1>
                                <1> pix_op_dec_16_0:
0000F4B7 66FF0F                 <1>       dec   word [edi]
0000F4BA 7903                   <1>       jns   short pix_op_dec_16_1
0000F4BC 66FF07                 <1>       inc   word [edi] ; 0 ; Min. value
                                <1> pix_op_dec_16_1:
0000F4BF 47                     <1>       inc edi
0000F4C0 47                     <1>       inc edi
0000F4C1 E2F4                   <1>       loop  pix_op_dec_16_0
0000F4C3 C3                     <1>       retn
                                <1>
                                <1> pix_op_dec_24:
                                <1>       ; 24 bit true colors
                                <1>       ; DECREASE PIXEL COLOR
                                <1>       ; ecx = pixel count per row
                                <1>       ; edi = start pixel address
                                <1>
                                <1> pix_op_dec_24_0:
0000F4C4 8B07                   <1>       mov   eax, [edi]
0000F4C6 25FFFFFF00             <1>       and   eax, 0FFFFFFh
0000F4CB 7401                   <1>       jz    short pix_op_dec_24_1
                                <1>                   ; 0 ; Min. value
0000F4CD 48                     <1>       dec   eax
                                <1> pix_op_dec_24_1:
0000F4CE 66AB                   <1>       stosw
0000F4D0 C1E810                 <1>       shr   eax, 16
0000F4D3 AA                     <1>       stosb
0000F4D4 E2B7                   <1>       loop  pix_op_inc_24_0
0000F4D6 C3                     <1>       retn
                                <1>
                                <1> pix_op_dec_32:
                                <1>       ; 32 bit true colors
                                <1>       ; DECREASE PIXEL COLOR
                                <1>       ; ecx = pixel count per row
                                <1>       ; edi = start pixel address
                                <1>
                                <1> pix_op_dec_32_0:
0000F4D7 FF0F                   <1>       dec   dword [edi]
0000F4D9 7902                   <1>       jns   short pix_op_dec_32_1
0000F4DB FF07                   <1>       inc   dword [edi] ; 0 ; Min. value
                                <1> pix_op_dec_32_1:
0000F4DD 83C704                 <1>       add   edi, 4
0000F4E0 E2F5                   <1>       loop  pix_op_dec_32_0
0000F4E2 89D0                   <1>       mov   eax, edx
0000F4E4 C3                     <1>       retn
                                <1>
                                <1> pix_op_rpl_8:
                                <1>       ; 8 bit colors (256 colors)
                                <1>       ; REPLACE PIXEL COLORS
                                <1>       ; ecx = pixel count per row
                                <1>       ;  al = new color
                                <1>       ; byte [maskcolor] = old color
```

```
                                <1>       ; edi = start pixel address
                                <1>
0000F4E5 8A25[9A120300]         <1>       mov   ah, [maskcolor]
                                <1> pix_op_rpl_8_0:
0000F4EB 3A27                   <1>       cmp   ah, [edi]
0000F4ED 7502                   <1>       jne   short pix_op_rpl_8_1
0000F4EF 8807                   <1>       mov   [edi], al
                                <1> pix_op_rpl_8_1:
0000F4F1 47                     <1>       inc   edi
0000F4F2 E2F7                   <1>       loop  pix_op_rpl_8_0
0000F4F4 C3                     <1>       retn
                                <1>
                                <1> pix_op_rpl_16:
                                <1>       ; 16 bit colors (65536 colors)
                                <1>       ; REPLACE PIXEL COLORS
                                <1>       ; ecx = pixel count per row
                                <1>       ;  ax = new color
                                <1>       ; word [maskcolor] = old color
                                <1>       ; edi = start pixel address
                                <1>
0000F4F5 8B15[9A120300]         <1>       mov   edx, [maskcolor]
                                <1> pix_op_rpl_16_0:
0000F4FB 663B17                 <1>       cmp   dx, [edi]
0000F4FE 7503                   <1>       jne   short pix_op_rpl_16_1
0000F500 668907                 <1>       mov   [edi], ax
                                <1> pix_op_rpl_16_1:
0000F503 47                     <1>       inc   edi
0000F504 47                     <1>       inc   edi
0000F505 E2F4                   <1>       loop  pix_op_rpl_16_0
0000F507 C3                     <1>       retn
                                <1>
                                <1> pix_op_rpl_24:
                                <1>       ; 24 bit true colors
                                <1>       ; REPLACE PIXEL COLORS
                                <1>       ; ecx = pixel count per row
                                <1>       ; eax = new color
                                <1>       ; [maskcolor] = old color
                                <1>       ; edi = start pixel address
                                <1>
                                <1> pix_op_rpl_24_0:
0000F508 8B17                   <1>       mov   edx, [edi]
0000F50A 81E2FFFFFF00           <1>       and   edx, 0FFFFFFh
0000F510 3B15[9A120300]         <1>       cmp   edx, [maskcolor]
0000F516 7406                   <1>       je    short pix_op_rpl_24_1
0000F518 83C703                 <1>       add   edi, 3
0000F51B E2EB                   <1>       loop  pix_op_rpl_24_0
0000F51D C3                     <1>       retn
                                <1> pix_op_rpl_24_1:
0000F51E AA                     <1>       stosb
0000F51F C1C808                 <1>       ror   eax, 8
0000F522 66AB                   <1>       stosw
0000F524 C1C008                 <1>       rol   eax, 8
0000F527 E2DF                   <1>       loop  pix_op_rpl_24_0
0000F529 C3                     <1>       retn
                                <1>
                                <1> pix_op_rpl_32:
                                <1>       ; 32 bit true colors
                                <1>       ; REPLACE PIXEL COLORS
                                <1>       ; ecx = pixel count per row
                                <1>       ; eax = new color
                                <1>       ; [maskcolor] = old color
                                <1>       ; edi = start pixel address
                                <1>
0000F52A 8B15[9A120300]         <1>       mov   edx, [maskcolor]
                                <1> pix_op_rpl_32_0:
0000F530 3B17                   <1>       cmp   edx, [edi]
0000F532 7504                   <1>       jne   short pix_op_rpl_32_2
0000F534 AB                     <1>       stosd
0000F535 E2F9                   <1>       loop  pix_op_rpl_32_0
0000F537 C3                     <1>       retn
                                <1> pix_op_rpl_32_2:
0000F538 83C704                 <1>       add   edi, 4
0000F53B E2F3                   <1>       loop  pix_op_rpl_32_0
0000F53D C3                     <1>       retn
                                <1>
                                <1> pix_op_chr:
                                <1>       ; 15/02/2021
                                <1>       ; 05/02/2021
                                <1>       ; WRITE CHARACTER (FONT)
                                <1>       ; 05/01/2021 ([ufont])
                                <1>       ; 01/01/2021
                                <1>       ;     CL = char's color (8 bit, 256 colors)
                                <1>       ;    ECX = char's color (16 bit and true colors)
                                <1>       ;     DL = Character's ASCII code
                                <1>       ;     DH bit 0 -> font height
                                <1>       ;        0 -> 8x16 character font
                                <1>       ;        1 -> 8x8 character font
                                <1>       ;     DH bit 1 & 2 -> scale
                                <1>       ;        0 = 1/1 (8 pixels per char row)
                                <1>       ;        1 = 2/1 (16 pixels per char row)
                                <1>       ;        2 = 3/1 (24 pixels per char row)
                                <1>       ;        3 = 4/1 (32 pixels per char row)
                                <1>       ;     DH bit 6 -> [ufont] option (1 = use [ufont])
                                <1>       ;     If DH bit 7 = 1
                                <1>       ;      USER FONT (from user buffer)
                                <1>       ;         DL = 0 -> 8x8 (width: 1 byte per row)
                                <1>       ;         DL = 1 -> 8x16
                                <1>       ;         DL = 2 -> 16x16 (width: 2 bytes)
                                <1>       ;         DL = 3 -> 16x32
                                <1>       ;         DL = 4 -> 24x24 (width: 3 bytes)
                                <1>       ;         DL = 5 -> 24x48
                                <1>       ;         DL = 6 -> 32x32 (width: 4 bytes)
                                <1>       ;         DL = 7 -> 32x64
                                <1>       ;         DL > 7 -> invalid (unused)
                                <1>       ;      EDI = user's font buffer address
```

```
                        <1>      ;          (NOTE: byte order is as row0,row1,row2..)
                        <1>      ;     ESI = start position (row, column) (*)
                        <1>      ;           (HW = row, SI = column)
                        <1>
0000F53E 89F0           <1>      mov    eax, esi ; start position
0000F540 E85BF2FFFF     <1>      call   calc_pixel_offset
0000F545 3B05[8E120300] <1>      cmp    eax, [v_siz]
0000F54B 736D           <1>      jnb    short pix_op_chr_err ; out of display page!
0000F54D E82BF2FFFF     <1>      call   pixels_to_byte_count
                        <1>      ; eax = font start offset
0000F552 0305[8A120300] <1>      add    eax, [v_mem] ; LFB start address
0000F558 A3[92120300]   <1>      mov    [v_str], eax ; font start address
                        <1>
0000F55D 890D[9A120300] <1>      mov    [maskcolor], ecx ; save char's color
                        <1>
0000F563 8835[88120300] <1>      mov    [v_ops], dh
                        <1>
0000F569 81E6FFFF0000   <1>      and    esi, 0FFFFh
0000F56F 8935[9E120300] <1>      mov    [buffer8], esi ; start column
                        <1>
0000F575 31DB           <1>      xor    ebx, ebx ; 0
0000F577 31C0           <1>      xor    eax, eax ; 15/02/2021
                        <1>
0000F579 F6C680         <1>      test   dh, 80h
0000F57C 7577           <1>      jnz    short pix_op_chr_u ; user font
                        <1>
0000F57E 80E63F         <1>      and    dh, 3Fh ; clear bit 6, [UFONT] option bit
0000F581 7409           <1>      jz     short pix_op_chr_0
                        <1>
0000F583 80FE07         <1>      cmp    dh, 7
0000F586 7732           <1>      ja     short pix_op_chr_err
                        <1>                    ; invalid (undefined) option
0000F588 88F4           <1>      mov    ah, dh
0000F58A D0EC           <1>      shr    ah, 1
                        <1>      ; ah = 0 to 3, scale
                        <1>      ;jmp   short pix_op_chr_font_pixels
                        <1>
                        <1> pix_op_chr_font_pixels:
                        <1>      ; 05/02/2021
                        <1>      ; write scaled font to buffer
                        <1>
                        <1>      ; DL = ASCII code of character
                        <1>      ; AH = scale
                        <1>      ; EDI = buffer address (kernel)
                        <1>
                        <1> pix_op_chr_0:
0000F58C 88D3           <1>      mov    bl, dl ; 15/02/2021
0000F58E 31C9           <1>      xor    ecx, ecx
0000F590 B610           <1>      mov    dh, 16
0000F592 F605[88120300]01 <1>    test   byte [v_ops], 1 ; 8x8 font ?
0000F599 7428           <1>      jz     short pix_op_chr_2 ; 8x16 font
0000F59B B608           <1>      mov    dh, 8
0000F59D C1E303         <1>      shl    ebx, 3 ; * 8
0000F5A0 F605[88120300]40 <1>    test   byte [v_ops], 40h ; [ufont] option
0000F5A7 7412           <1>      jz     short pix_op_chr_1  ; no
                        <1>      ; test 8x8 user font is ready flag
0000F5A9 F605[7E120300]01 <1>    test   byte [ufont], 1
0000F5B0 7409           <1>      jz     short pix_op_chr_1 ; no
0000F5B2 81C300500900   <1>      add    ebx, VGAFONT8USER
0000F5B8 EB2C           <1>      jmp    short pix_op_chr_fpos_0
                        <1> pix_op_chr_err:
0000F5BA C3             <1>      retn
                        <1> pix_op_chr_1:
0000F5BB 81C3[54600100] <1>      add    ebx, vgafont8 ; system font (8x8)
0000F5C1 EB23           <1>      jmp    short pix_op_chr_fpos_0
                        <1> pix_op_chr_2:
0000F5C3 C1E304         <1>      shl    ebx, 4 ; * 16
0000F5C6 F605[88120300]40 <1>    test   byte [v_ops], 40h ; [ufont] option
0000F5CD 7411           <1>      jz     short pix_op_chr_3  ; no
                        <1>      ; test 8x16 user font is ready flag
0000F5CF F605[7E120300]02 <1>    test   byte [ufont], 2
0000F5D6 7408           <1>      jz     short pix_op_chr_3 ; no
0000F5D8 81C300400900   <1>      add    ebx, VGAFONT16USER
0000F5DE EB06           <1>      jmp    short pix_op_chr_fpos_0
                        <1> pix_op_chr_3:
0000F5E0 81C3[54760100] <1>      add    ebx, vgafont16 ; system font (8x16)
                        <1> pix_op_chr_fpos_0:
0000F5E6 20E4           <1>      and    ah, ah
0000F5E8 7549           <1>      jnz    short pix_op_chr_fpos_1 ; scale > 1
                        <1>      ; no scale (scale = 1)
0000F5EA 89DE           <1>      mov    esi, ebx ; 15/02/2021
0000F5EC 88F1           <1>      mov    cl, dh ; rows/height (16 or 8)
0000F5EE B608           <1>      mov    dh, 8  ; columns/width
0000F5F0 E9C4000000     <1>      jmp    pix_op_chr_f2p
                        <1> pix_op_chr_u:
                        <1>      ; write user defined font
0000F5F5 80FE80         <1>      cmp    dh, 80h
0000F5F8 75C0           <1>      jne    short pix_op_chr_err
0000F5FA 80FA07         <1>      cmp    dl, 7
0000F5FD 77BB           <1>      ja     short pix_op_chr_err
                        <1>
                        <1>      ;xor   eax, eax
                        <1>      ; eax = 0 ; 15/02/2021
0000F5FF 88D4           <1>      mov    ah, dl
0000F601 D0EC           <1>      shr    ah, 1
0000F603 FEC4           <1>      inc    ah
                        <1>      ; ah =  1 to 4
0000F605 88E0           <1>      mov    al, ah
0000F607 C0E003         <1>      shl    al, 3 ; * 8
                        <1>      ; al = 8,16,24,32
0000F60A 88C3           <1>      mov    bl, al
0000F60C 88C7           <1>      mov    bh, al
0000F60E F6E4           <1>      mul    ah
                        <1>      ; ax = 8,32,72,128 bytes
0000F610 F6C201         <1>      test   dl, 1
```

```
6882 0000F613 7405              <1>       jz     short pix_op_chr_u_0
6883 0000F615 66D1E0            <1>       shl    ax, 1 ; *2
6884                            <1>       ; ax = 16,32,144,256 bytes
6885 0000F618 D0E7              <1>       shl    bh, 1
6886                            <1> pix_op_chr_u_0:
6887                            <1>       ; eax = byte count
6888 0000F61A 89C1              <1>       mov    ecx, eax
6889 0000F61C BF00760900        <1>       mov    edi, VBE3SAVERESTOREBLOCK
6890                            <1>       ; esi = user buffer
6891 0000F621 E816250000        <1>       call   transfer_from_user_buffer
6892 0000F626 7292              <1>       jc     short pix_op_chr_err
6893                            <1>
6894 0000F628 88F9              <1>       mov    cl, bh ; rows/height
6895 0000F62A 88DE              <1>       mov    dh, bl ; columns (width)
6896 0000F62C 89FE              <1>       mov    esi, edi ; VBE3SAVERESTOREBLOCK
6897 0000F62E E986000000        <1>       jmp    pix_op_chr_f2p
6898                            <1>
6899                            <1> pix_op_chr_fpos_1:
6900                            <1>       ; scale > 1
6901 0000F633 88F5              <1>       mov    ch, dh ; 16 or 8
6902 0000F635 BF00760900        <1>       mov    edi, VBE3SAVERESTOREBLOCK
6903 0000F63A 89FE              <1>       mov    esi, edi
6904 0000F63C FECC              <1>       dec    ah
6905 0000F63E 7525              <1>       jnz    short pix_op_chr_fpos_5 ; scale > 2
6906                            <1>       ; scale = 2
6907                            <1> pix_op_chr_fpos_2:
6908 0000F640 B108              <1>       mov    cl, 8
6909 0000F642 8A13              <1>       mov    dl, [ebx]
6910                            <1> pix_op_chr_fpos_3:
6911 0000F644 66C1E002          <1>       shl    ax, 2
6912 0000F648 D0EA              <1>       shr    dl, 1
6913 0000F64A 7302              <1>       jnc    short pix_op_chr_fpos_4
6914 0000F64C 0C03              <1>       or     al, 3
6915                            <1> pix_op_chr_fpos_4:
6916 0000F64E FEC9              <1>       dec    cl
6917 0000F650 75F2              <1>       jnz    short pix_op_chr_fpos_3
6918 0000F652 66AB              <1>       stosw
6919 0000F654 43                <1>       inc    ebx
6920 0000F655 66894720          <1>       mov    [edi+32],ax
6921 0000F659 FECD              <1>       dec    ch
6922 0000F65B 75E3              <1>       jnz    short pix_op_chr_fpos_2
6923                            <1>       ; scale = 2
6924 0000F65D 88F1              <1>       mov    cl, dh ; 16 or 8 (height/rows)
6925 0000F65F D0E1              <1>       shl    cl, 1 ; 32 or 16 rows
6926 0000F661 B610              <1>       mov    dh, 16 ; columns (width)
6927 0000F663 EB54              <1>       jmp    short pix_op_chr_f2p
6928                            <1> pix_op_chr_fpos_5:
6929 0000F665 FECC              <1>       dec    ah
6930 0000F667 752D              <1>       jnz    short pix_op_chr_fpos_9 ; scale = 4
6931                            <1>       ; scale = 3
6932                            <1> pix_op_chr_fpos_6:
6933 0000F669 B108              <1>       mov    cl, 8
6934 0000F66B 8A13              <1>       mov    dl, [ebx]
6935                            <1> pix_op_chr_fpos_7:
6936 0000F66D C1E003            <1>       shl    eax, 3
6937 0000F670 D0EA              <1>       shr    dl, 1
6938 0000F672 7302              <1>       jnc    short pix_op_chr_fpos_8
6939 0000F674 0C07              <1>       or     al, 7
6940                            <1> pix_op_chr_fpos_8:
6941 0000F676 FEC9              <1>       dec    cl
6942 0000F678 75F3              <1>       jnz    short pix_op_chr_fpos_7
6943 0000F67A 66AB              <1>       stosw
6944 0000F67C 66894730          <1>       mov    [edi+48], ax
6945 0000F680 C1E810            <1>       shr    eax, 16
6946 0000F683 AA                <1>       stosb
6947 0000F684 884730            <1>       mov    [edi+48], al
6948 0000F687 43                <1>       inc    ebx
6949 0000F688 FECD              <1>       dec    ch
6950 0000F68A 75DD              <1>       jnz    short pix_op_chr_fpos_6
6951                            <1>       ; scale = 3
6952 0000F68C 88F1              <1>       mov    cl, dh ; 16 or 8 (height/rows)
6953 0000F68E D0E1              <1>       shl    cl, 1
6954 0000F690 00F1              <1>       add    cl, dh ; 48 or 24 rows
6955 0000F692 B618              <1>       mov    dh, 24 ; columns (width)
6956 0000F694 EB23              <1>       jmp    short pix_op_chr_f2p
6957                            <1>
6958                            <1> pix_op_chr_fpos_9:
6959                            <1>       ; scale = 4
6960 0000F696 B108              <1>       mov    cl, 8
6961 0000F698 8A13              <1>       mov    dl, [ebx]
6962                            <1> pix_op_chr_fpos_10:
6963 0000F69A 66C1E004          <1>       shl    ax, 4
6964 0000F69E D0EA              <1>       shr    dl, 1
6965 0000F6A0 7302              <1>       jnc    short pix_op_chr_fpos_11
6966 0000F6A2 0C07              <1>       or     al, 7
6967                            <1> pix_op_chr_fpos_11:
6968 0000F6A4 FEC9              <1>       dec    cl
6969 0000F6A6 75F2              <1>       jnz    short pix_op_chr_fpos_10
6970 0000F6A8 AB                <1>       stosd
6971 0000F6A9 43                <1>       inc    ebx
6972 0000F6AA 66894740          <1>       mov    [edi+64],ax
6973 0000F6AE FECD              <1>       dec    ch
6974 0000F6B0 75E4              <1>       jnz    short pix_op_chr_fpos_9
6975                            <1>       ; scale = 4
6976 0000F6B2 88F1              <1>       mov    cl, dh ; 16 or 8 (height/rows)
6977 0000F6B4 C0E102            <1>       shl    cl, 2 ; 64 or 32 rows
6978 0000F6B7 B620              <1>       mov    dh, 32 ; columns (width)
6979                            <1>       ;jmp   short pix_op_chr_f2p
6980                            <1>
6981                            <1> pix_op_chr_f2p:
6982                            <1>       ; write font pixels
6983 0000F6B9 8B3D[92120300]    <1>       mov    edi, [v_str]
6984                            <1>       ; 15/02/2021
6985                            <1> pix_op_chr_f2p_next:
6986 0000F6BF 80FE08            <1>       cmp    dh, 8
```

```
6987 0000F6C2 7706              <1>       ja     short pix_op_chr_f2p_24
6988                            <1> pix_op_chr_f2p_8:
6989 0000F6C4 AC                <1>       lodsb
6990 0000F6C5 C1E018            <1>       shl    eax, 24 ; 15/02/2021
6991 0000F6C8 EB1F              <1>       jmp    short pix_op_chr_f2p_0
6992                            <1> pix_op_chr_f2p_24:
6993 0000F6CA 80FE18            <1>       cmp    dh, 24
6994 0000F6CD 7719              <1>       ja     short pix_op_chr_f2p_32
6995 0000F6CF 7210              <1>       jb     short pix_op_chr_f2p_16
6996 0000F6D1 668B06            <1>       mov    ax, [esi]
6997 0000F6D4 C1E010            <1>       shl    eax, 16
6998 0000F6D7 46                <1>       inc    esi
6999 0000F6D8 46                <1>       inc    esi
7000                            <1>       ; 15/02/2021
7001 0000F6D9 8A06              <1>       mov    al, [esi]
7002 0000F6DB 46                <1>       inc    esi
7003 0000F6DC C1C808            <1>       ror    eax, 8
7004 0000F6DF EB08              <1>       jmp    short pix_op_chr_f2p_0
7005                            <1> pix_op_chr_f2p_16:
7006 0000F6E1 66AD              <1>       lodsw
7007 0000F6E3 C1E010            <1>       shl    eax, 16 ; 15/02/2021
7008 0000F6E6 EB01              <1>       jmp    short pix_op_chr_f2p_0
7009                            <1> pix_op_chr_f2p_32:
7010 0000F6E8 AD                <1>       lodsd
7011                            <1> pix_op_chr_f2p_0:
7012                            <1>       ; EAX = font row (8,16,24,32 pixels)
7013                            <1>       ;      (bits are shifted to left)
7014                            <1>       ; CL = rows
7015                            <1>       ; DH = bits per row (8,16,24,32)
7016 0000F6E9 8B1D[9E120300]    <1>       mov    ebx, [buffer8] ; start column
7017 0000F6EF 57                <1>       push   edi ; *
7018 0000F6F0 52                <1>       push   edx ; **
7019                            <1> pix_op_chr_f2p_1:
7020 0000F6F1 E827000000        <1>       call   pix_op_chr_w_pixel
7021                            <1> pix_op_chr_f2p_2:
7022 0000F6F6 663B1D[86120300]  <1>       cmp    bx, [v_width] ; current column
7023 0000F6FD 7304              <1>       jnb    short pix_op_chr_f2p_3
7024 0000F6FF FECE              <1>       dec    dh
7025 0000F701 75EE              <1>       jnz    short pix_op_chr_f2p_1 ; next bit
7026                            <1> pix_op_chr_f2p_3:
7027                            <1>       ;mov   ebx, [buffer8]
7028 0000F703 5A                <1>       pop    edx ; **
7029 0000F704 58                <1>       pop    eax ; *
7030 0000F705 3B3D[96120300]    <1>       cmp    edi, [v_end]
7031 0000F70B 730F              <1>       jnb    short pix_op_chr_f2p_4
7032 0000F70D FEC9              <1>       dec    cl
7033 0000F70F 740B              <1>       jz     short pix_op_chr_f2p_4
7034 0000F711 0FB73D[86120300]  <1>       movzx  edi, word [v_width]
7035 0000F718 01C7              <1>       add    edi, eax ; next position
7036 0000F71A EBA3              <1>       jmp    short pix_op_chr_f2p_next
7037                            <1> pix_op_chr_f2p_4:
7038 0000F71C C3                <1>       retn
7039                            <1>
7040                            <1> pix_op_chr_w_pixel:
7041                            <1>       ; 15/02/2021
7042 0000F71D 89C5              <1>       mov    ebp, eax
7043 0000F71F A1[9A120300]      <1>       mov    eax, [maskcolor]
7044 0000F724 803D[89120300]08  <1>       cmp    byte [v_bpp], 8 ; 8bpp
7045 0000F72B 7711              <1>       ja     short pix_op_chr_wp_2
7046                            <1>       ; 256 colors (1 byte per pixel)
7047 0000F72D D1E5              <1>       shl    ebp, 1
7048 0000F72F 7302              <1>       jnc    short pix_op_chr_wp_0
7049 0000F731 8807              <1>       mov    [edi], al
7050                            <1> pix_op_chr_wp_0:
7051 0000F733 47                <1>       inc    edi
7052 0000F734 FF05[64030300]    <1>       inc    dword [u.r0] ; +1
7053                            <1> pix_op_chr_wp_1:
7054 0000F73A 43                <1>       inc    ebx
7055 0000F73B 89E8              <1>       mov    eax, ebp
7056 0000F73D C3                <1>       retn
7057                            <1> pix_op_chr_wp_2:
7058 0000F73E 803D[89120300]18  <1>       cmp    byte [v_bpp], 24 ; 24bpp
7059 0000F745 772D              <1>       ja     short pix_op_chr_wp_6 ; 32bpp
7060 0000F747 721B              <1>       jb     short pix_op_chr_wp_4 ; 16bpp
7061                            <1>       ; 24 bit true colors
7062                            <1>       ; * 3
7063 0000F749 D1E5              <1>       shl    ebp, 1
7064 0000F74B 7309              <1>       jnc    short pix_op_chr_wp_3
7065 0000F74D 668907            <1>       mov    [edi], ax
7066 0000F750 C1E810            <1>       shr    eax, 16
7067 0000F753 884702            <1>       mov    [edi+2], al
7068                            <1> pix_op_chr_wp_3:
7069 0000F756 66B80300          <1>       mov    ax, 3
7070 0000F75A 01C7              <1>       add    edi, eax  ; add edi, 3
7071 0000F75C 0105[64030300]    <1>       add    [u.r0], eax ; +3
7072                            <1>
7073 0000F762 EBD6              <1>       jmp    short pix_op_chr_wp_1
7074                            <1>
7075                            <1> pix_op_chr_wp_4:
7076                            <1>       ; 16 bit (65536) colors
7077 0000F764 D1E5              <1>       shl    ebp, 1
7078 0000F766 7303              <1>       jnc    short pix_op_chr_wp_5
7079 0000F768 668907            <1>       mov    [edi], ax
7080                            <1> pix_op_chr_wp_5:
7081 0000F76B 47                <1>       inc    edi
7082 0000F76C FF05[64030300]    <1>       inc    dword [u.r0] ; +1
7083 0000F772 EBBF              <1>       jmp    short pix_op_chr_wp_0
7084                            <1>
7085                            <1> pix_op_chr_wp_6:
7086                            <1>       ; 32 bit true colors
7087 0000F774 D1E5              <1>       shl    ebp, 1
7088 0000F776 7302              <1>       jnc    short pix_op_chr_wp_7
7089 0000F778 8907              <1>       mov    [edi], eax
7090                            <1> pix_op_chr_wp_7:
7091 0000F77A 31C0              <1>       xor    eax, eax
```

```
7092 0000F77C B004                <1>       mov    al, 4
7093 0000F77E 01C7                <1>       add    edi, eax ; add edi, 4
7094 0000F780 0105[64030300]      <1>       add    [u.r0], eax ; +4
7095 0000F786 EBB2                <1>       jmp    short pix_op_chr_wp_1
7096                              <1>
7097                              <1> m_pix_op_cpy:
7098                              <1>       ; 06/02/2021
7099                              <1>       ; MASKED COPY PIXELS (full screen)
7100                              <1>       ;
7101                              <1>       ; jump from pix_op_cpy
7102                              <1>       ;
7103                              <1>       ; INPUT:
7104                              <1>       ;   ecx = transfer count (bytes)
7105                              <1>       ;   edi = [v_mem] = start address of LFB
7106                              <1>       ;   esi = user's buffer address (virtual)
7107                              <1>       ;
7108                              <1>       ; OUTPUT:
7109                              <1>       ;   [u.r0] will be > 0 if succesful
7110                              <1>
7111                              <1>       ; Full screen masked copy
7112                              <1>
7113                              <1> m_pix_op_cpy_0:
7114 0000F788 57                  <1>       push   edi ; **
7115 0000F789 51                  <1>       push   ecx ; *
7116 0000F78A 81F9F8070000        <1>       cmp    ecx, 2040
7117 0000F790 7605                <1>       jna    short m_pix_op_cpy_1
7118 0000F792 B9F8070000          <1>       mov    ecx, 2040
7119                              <1> m_pix_op_cpy_1:
7120 0000F797 BF00760900          <1>       mov    edi, VBE3SAVERESTOREBLOCK ; temporary buff
7121 0000F79C E89B230000          <1>       call   transfer_from_user_buffer
7122 0000F7A1 726D                <1>       jc     short m_pix_op_cpy_3
7123 0000F7A3 01CE                <1>       add    esi, ecx
7124 0000F7A5 89F5                <1>       mov    ebp, esi  ; save user's buffer address
7125 0000F7A7 89FE                <1>       mov    esi, edi
7126 0000F7A9 89CB                <1>       mov    ebx, ecx
7127 0000F7AB 59                  <1>       pop    ecx ; *
7128 0000F7AC 29D9                <1>       sub    ecx, ebx
7129 0000F7AE 5F                  <1>       pop    edi ; **
7130 0000F7AF 803D[89120300]08    <1>       cmp    byte [v_bpp], 8
7131 0000F7B6 7438                <1>       je     short m_pix_op_cpy_1_8
7132 0000F7B8 803D[89120300]18    <1>       cmp    byte [v_bpp], 24
7133 0000F7BF 7773                <1>       ja     short m_pix_op_cpy_1_32
7134 0000F7C1 7250                <1>       jb     short m_pix_op_cpy_1_16
7135                              <1> m_pix_op_cpy_1_24:
7136                              <1>       ; 24 bit masked copy
7137 0000F7C3 BA03000000          <1>       mov    edx, 3
7138                              <1> m_pix_op_cpy_1_24_0:
7139 0000F7C8 66AD                <1>       lodsw
7140 0000F7CA C1E010              <1>       shl    eax, 16
7141 0000F7CD AC                  <1>       lodsb
7142 0000F7CE C1C010              <1>       rol    eax, 16
7143 0000F7D1 3B05[9A120300]      <1>       cmp    eax, [maskcolor]
7144 0000F7D7 740F                <1>       je     short m_pix_op_cpy_1_24_1 ; exclude
7145 0000F7D9 668907              <1>       mov    [edi], ax
7146 0000F7DC C1E810              <1>       shr    eax, 16
7147 0000F7DF 884702              <1>       mov    [edi+2], al
7148 0000F7E2 0115[64030300]      <1>       add    [u.r0], edx ; +3
7149                              <1> m_pix_op_cpy_1_24_1:
7150 0000F7E8 01D7                <1>       add    edi, edx ; +3
7151 0000F7EA 29D3                <1>       sub    ebx, edx ; sub ebx, 3
7152 0000F7EC 77DA                <1>       ja     short m_pix_op_cpy_1_24_0
7153 0000F7EE EB15                <1>       jmp    short m_pix_op_cpy_2
7154                              <1>
7155                              <1> m_pix_op_cpy_1_8:
7156                              <1>       ; 8 bit masked copy
7157 0000F7F0 AC                  <1>       lodsb
7158 0000F7F1 3A05[9A120300]      <1>       cmp    al, [maskcolor]
7159 0000F7F7 7408                <1>       je     short m_pix_op_cpy_1_8_1 ; exclude
7160 0000F7F9 8807                <1>       mov    [edi], al
7161 0000F7FB FF05[64030300]      <1>       inc    dword [u.r0] ; +1
7162                              <1> m_pix_op_cpy_1_8_1:
7163 0000F801 47                  <1>       inc    edi ; +1
7164 0000F802 4B                  <1>       dec    ebx
7165 0000F803 75EB                <1>       jnz    short m_pix_op_cpy_1_8
7166                              <1> m_pix_op_cpy_2:
7167 0000F805 89EE                <1>       mov    esi, ebp ; restore user's buffer addr
7168 0000F807 09C9                <1>       or     ecx, ecx
7169 0000F809 0F8579FFFFFF        <1>       jnz    m_pix_op_cpy_0
7170 0000F80F C3                  <1>       retn
7171                              <1> m_pix_op_cpy_3:
7172 0000F810 59                  <1>       pop    ecx ; *
7173 0000F811 5F                  <1>       pop    edi ; **
7174 0000F812 C3                  <1>       retn
7175                              <1>
7176                              <1> m_pix_op_cpy_1_16:
7177                              <1>       ; 16 bit masked copy
7178 0000F813 BA02000000          <1>       mov    edx, 2
7179                              <1> m_pix_op_cpy_1_16_0:
7180 0000F818 66AD                <1>       lodsw
7181 0000F81A 663B05[9A120300]    <1>       cmp    ax, [maskcolor]
7182 0000F821 7409                <1>       je     short m_pix_op_cpy_1_16_1 ; exclude
7183 0000F823 668907              <1>       mov    [edi], ax
7184 0000F826 0115[64030300]      <1>       add    [u.r0], edx ; +2
7185                              <1> m_pix_op_cpy_1_16_1:
7186 0000F82C 01D7                <1>       add    edi, edx ; +2
7187 0000F82E 29D3                <1>       sub    ebx, edx ; sub ebx, 2
7188 0000F830 77E6                <1>       ja     short m_pix_op_cpy_1_16_0
7189 0000F832 EBD1                <1>       jmp    short m_pix_op_cpy_2
7190                              <1>
7191                              <1> m_pix_op_cpy_1_32:
7192                              <1>       ; 32 bit masked copy
7193 0000F834 BA04000000          <1>       mov    edx, 4
7194                              <1> m_pix_op_cpy_1_32_0:
7195 0000F839 AD                  <1>       lodsd
7196 0000F83A 3B05[9A120300]      <1>       cmp    eax, [maskcolor]
```

```
7197 0000F840 7408              <1>       je     short m_pix_op_cpy_1_32_1 ; exclude
7198 0000F842 8907              <1>       mov    [edi], eax
7199 0000F844 0115[64030300]    <1>       add    [u.r0], edx ; +4
7200                            <1> m_pix_op_cpy_1_32_1:
7201 0000F84A 01D7              <1>       add    edi, edx ; +4
7202 0000F84C 29D3              <1>       sub    ebx, edx ; sub ebx, 4
7203 0000F84E 77E9              <1>       ja     short m_pix_op_cpy_1_32_0
7204 0000F850 EBB3              <1>       jmp    short m_pix_op_cpy_2
7205                            <1>
7206                            <1> m_pix_op_cpy_w:
7207                            <1>       ; 06/02/2021
7208                            <1>       ; MASKED COPY PIXELS (window)
7209                            <1>       ;
7210                            <1>       ; jump from pix_op_cpy_w
7211                            <1>       ;
7212                            <1>       ; INPUT:
7213                            <1>       ;   ecx = bytes per row (to be applied)
7214                            <1>       ;   edx = screen width in bytes
7215                            <1>       ;   ebx = row count
7216                            <1>       ;
7217                            <1>       ; OUTPUT:
7218                            <1>       ;   [u.r0] will be > 0 if succesful
7219                            <1>
7220                            <1>       ; Window masked copy
7221                            <1>
7222                            <1> m_pix_op_cpy_w_0:
7223 0000F852 8B3D[92120300]    <1>       mov    edi, [v_str]
7224                            <1> m_pix_op_cpy_w_1:
7225 0000F858 52                <1>       push   edx
7226 0000F859 57                <1>       push   edi
7227 0000F85A 56                <1>       push   esi
7228 0000F85B 53                <1>       push   ebx
7229 0000F85C 51                <1>       push   ecx
7230 0000F85D E826FFFFFF        <1>       call   m_pix_op_cpy_0
7231 0000F862 59                <1>       pop    ecx
7232 0000F863 5B                <1>       pop    ebx
7233 0000F864 5E                <1>       pop    esi
7234 0000F865 5F                <1>       pop    edi
7235 0000F866 5A                <1>       pop    edx
7236 0000F867 7209              <1>       jc     short m_pix_op_cpy_w_2
7237 0000F869 4B                <1>       dec    ebx
7238 0000F86A 7406              <1>       jz     short m_pix_op_cpy_w_2 ; ok.
7239                            <1>       ; next row
7240 0000F86C 01CE              <1>       add    esi, ecx ; next row in user's buffer
7241 0000F86E 01D7              <1>       add    edi, edx ; next row of window (system)
7242 0000F870 EBE6              <1>       jmp    short m_pix_op_cpy_w_1
7243                            <1> m_pix_op_cpy_w_2:
7244 0000F872 C3                <1>       retn
7245                            <1>
7246                            <1> m_pix_op_new:
7247                            <1>       ; 06/02/2021
7248                            <1>       ; CHANGE COLOR (MASKED, full screen)
7249                            <1>       ;
7250                            <1>       ; jump from pix_op_new
7251                            <1>       ;
7252                            <1>       ; INPUT:
7253                            <1>       ;   eax = color (AL, AX, EAX)
7254                            <1>       ;   ecx = [v_siz] ; display page pixel count
7255                            <1>       ;   esi = edi = [v_mem] ; LFB start address
7256                            <1>       ;
7257                            <1>       ;   [maskcolor] = mask color (to be excluded)
7258                            <1>       ;
7259                            <1>       ; OUTPUT:
7260                            <1>       ;     [u.r0] will be > 0 if succesful
7261                            <1>
7262                            <1>       ; Full screen
7263                            <1> m_pix_op_new_0:
7264 0000F873 803D[89120300]08  <1>       cmp    byte [v_bpp], 8 ; 8bpp
7265 0000F87A 7717              <1>       ja     short m_pix_op_new_1
7266                            <1>       ; 256 colors (8bpp)
7267                            <1>       ;jmp   short m_pix_op_new_8
7268                            <1> m_pix_op_new_8:
7269                            <1>       ; 8 bit colors (256 colors)
7270 0000F87C 88C2              <1>       mov    dl, al ; new color
7271                            <1> m_pix_op_new_8_0:
7272 0000F87E AC                <1>       lodsb
7273 0000F87F 3A05[9A120300]    <1>       cmp    al, [maskcolor]
7274 0000F885 7408              <1>       je     short m_pix_op_new_8_1 ; exclude
7275 0000F887 8817              <1>       mov    [edi], dl
7276 0000F889 FF05[64030300]    <1>       inc    dword [u.r0]
7277                            <1> m_pix_op_new_8_1:
7278 0000F88F 47                <1>       inc    edi
7279 0000F890 E2EC              <1>       loop   m_pix_op_new_8_0
7280 0000F892 C3                <1>       retn
7281                            <1> m_pix_op_new_1:
7282 0000F893 803D[89120300]18  <1>       cmp    byte [v_bpp], 24 ; 24bpp
7283 0000F89A 774B              <1>       ja     short m_pix_op_new_3 ; 32bpp
7284 0000F89C 722C              <1>       jb     short m_pix_op_new_2 ; 16bpp
7285                            <1>       ; 24 bit true colors
7286                            <1>       ;jmp   short m_pix_op_new_24
7287                            <1> m_pix_op_new_24:
7288                            <1>       ; 24 bit true colors
7289 0000F89E 89C2              <1>       mov    edx, eax ; new color
7290                            <1> m_pix_op_new_24_0:
7291 0000F8A0 66AD              <1>       lodsw
7292 0000F8A2 C1E010            <1>       shl    eax, 16
7293 0000F8A5 AC                <1>       lodsb
7294 0000F8A6 C1C010            <1>       rol    eax, 16
7295 0000F8A9 3B05[9A120300]    <1>       cmp    eax, [maskcolor]
7296 0000F8AF 7413              <1>       je     short m_pix_op_new_24_1 ; exclude
7297 0000F8B1 668917            <1>       mov    [edi], dx
7298 0000F8B4 C1CA10            <1>       ror    edx, 16
7299 0000F8B7 885702            <1>       mov    [edi+2], dl
7300 0000F8BA C1C210            <1>       rol    edx, 16
7301 0000F8BD 8305[64030300]03  <1>       add    dword [u.r0], 3
```

```
7302                                <1> m_pix_op_new_24_1:
7303 0000F8C4 83C703               <1>         add    edi, 3
7304 0000F8C7 E2D7                 <1>         loop   m_pix_op_new_24_0
7305 0000F8C9 C3                   <1>         retn
7306                                <1>         ; 65536 colors (16bpp)
7307                                <1> m_pix_op_new_2:
7308                                <1>         ;jmp   short m_pix_op_new_16
7309                                <1> m_pix_op_new_16:
7310                                <1>         ; 16 bit colors (65536 colors)
7311 0000F8CA 89C2                 <1>         mov    edx, eax ; new color
7312                                <1> m_pix_op_new_16_0:
7313 0000F8CC 66AD                 <1>         lodsw
7314 0000F8CE 663B05[9A120300]     <1>         cmp    ax, [maskcolor]
7315 0000F8D5 740A                 <1>         je     short m_pix_op_new_16_1 ; exclude
7316 0000F8D7 668917               <1>         mov    [edi], dx
7317 0000F8DA 8305[64030300]02     <1>         add    dword [u.r0], 2
7318                                <1> m_pix_op_new_16_1:
7319 0000F8E1 83C702               <1>         add    edi, 2
7320 0000F8E4 E2E6                 <1>         loop   m_pix_op_new_16_0
7321 0000F8E6 C3                   <1>         retn
7322                                <1> m_pix_op_new_3:
7323                                <1>         ; 32 bit true colors
7324                                <1>         ;jmp   short m_pix_op_new_32
7325                                <1> m_pix_op_new_32:
7326                                <1>         ; 32 bit true colors
7327 0000F8E7 89C2                 <1>         mov    edx, eax ; new color
7328                                <1> m_pix_op_new_32_0:
7329 0000F8E9 AD                   <1>         lodsd
7330 0000F8EA 3B05[9A120300]       <1>         cmp    eax, [maskcolor]
7331 0000F8F0 7409                 <1>         je     short m_pix_op_new_32_1 ; exclude
7332 0000F8F2 8917                 <1>         mov    [edi], edx
7333 0000F8F4 8305[64030300]04     <1>         add    dword [u.r0], 4
7334                                <1> m_pix_op_new_32_1:
7335 0000F8FB 83C704               <1>         add    edi, 4
7336 0000F8FE E2E9                 <1>         loop   m_pix_op_new_32_0
7337 0000F900 C3                   <1>         retn
7338                                <1>
7339                                <1> m_pix_op_new_w:
7340                                <1>         ; 06/02/2021
7341                                <1>         ; CHANGE COLOR (MASKED, window)
7342                                <1>         ;
7343                                <1>         ; jump from pix_op_new_w
7344                                <1>         ;
7345                                <1>         ; INPUT:
7346                                <1>         ;   ecx = bytes per row (to be applied)
7347                                <1>         ;   edx = screen width in bytes
7348                                <1>         ;   ebx = row count
7349                                <1>         ;   eax = color
7350                                <1>         ;
7351                                <1>         ;   [maskcolor] = mask color (to be excluded)
7352                                <1>         ;
7353                                <1>         ; OUTPUT:
7354                                <1>         ;      [u.r0] will be > 0 if succesful
7355                                <1>
7356                                <1>         ; Window
7357                                <1>         ;mov   edi, [v_str] ; LFB start address
7358                                <1>         ;mov   esi, edi
7359                                <1>
7360 0000F901 803D[89120300]08     <1>         cmp    byte [v_bpp], 8 ; 8bpp
7361 0000F908 7707                 <1>         ja     short m_pix_op_new_w_1
7362                                <1>
7363                                <1>         ; 256 colors (8bpp)
7364 0000F90A BD[7CF80000]         <1>         mov    ebp, m_pix_op_new_8
7365 0000F90F EB1E                 <1>         jmp    short m_pix_op_new_w_x
7366                                <1>
7367                                <1> m_pix_op_new_w_1:
7368 0000F911 803D[89120300]18     <1>         cmp    byte [v_bpp], 24 ; 24bpp
7369 0000F918 7710                 <1>         ja     short m_pix_op_new_w_3 ; 32bpp
7370 0000F91A 7207                 <1>         jb     short m_pix_op_new_w_2 ; 16bpp
7371                                <1>
7372                                <1>         ; 24 bit true colors
7373 0000F91C BD[9EF80000]         <1>         mov    ebp, m_pix_op_new_24
7374 0000F921 EB0C                 <1>         jmp    short m_pix_op_new_w_x
7375                                <1>
7376                                <1>         ; 65536 colors (16bpp)
7377                                <1> m_pix_op_new_w_2:
7378 0000F923 BD[CAF80000]         <1>         mov    ebp, m_pix_op_new_16
7379 0000F928 EB05                 <1>         jmp    short m_pix_op_new_w_x
7380                                <1>
7381                                <1>         ; 32 bit true colors
7382                                <1> m_pix_op_new_w_3:
7383 0000F92A BD[E7F80000]         <1>         mov    ebp, m_pix_op_new_32
7384                                <1>         ;jmp   short m_pix_op_new_w_x
7385                                <1>
7386                                <1> m_pix_op_new_w_x:
7387                                <1> m_pix_op_add_w_x:
7388                                <1> m_pix_op_sub_w_x:
7389                                <1> m_pix_op_mix_w_x:
7390                                <1> m_pix_op_and_w_x:
7391                                <1> m_pix_op_orc_w_x:
7392                                <1> m_pix_op_xor_w_x:
7393                                <1> m_pix_op_not_w_x:
7394                                <1> m_pix_op_neg_w_x:
7395                                <1> m_pix_op_inc_w_x:
7396                                <1> m_pix_op_dec_w_x:
7397                                <1>         ; 06/02/2021
7398                                <1>         ; ecx = bytes per row (to be applied)
7399                                <1>         ; edx = windows (screen) width in bytes
7400                                <1>         ; ebx = row count
7401                                <1>         ; eax = color
7402                                <1>         ; ebp = pixel operation subroutine address
7403                                <1>         ; edi = esi = window start address
7404                                <1>
7405 0000F92F 8B3D[92120300]       <1>         mov    edi, [v_str] ;  LFB start address
7406 0000F935 89FE                 <1>         mov    esi, edi
```

```
7407                                <1> m_pix_op_w_x_next:
7408 0000F937 52                   <1>         push   edx
7409 0000F938 51                   <1>         push   ecx
7410 0000F939 56                   <1>         push   esi
7411 0000F93A 57                   <1>         push   edi
7412 0000F93B FFD5                 <1>         call   ebp ; call masked pixel-row operation
7413 0000F93D 5F                   <1>         pop    edi
7414 0000F93E 5E                   <1>         pop    esi
7415 0000F93F 59                   <1>         pop    ecx
7416 0000F940 5A                   <1>         pop    edx
7417 0000F941 01D6                 <1>         add    esi, edx ; next row
7418 0000F943 01D7                 <1>         add    edi, edx ; next row
7419 0000F945 4B                   <1>         dec    ebx
7420 0000F946 75EF                 <1>         jnz    short m_pix_op_w_x_next
7421 0000F948 C3                   <1>         retn
7422                                <1>
7423                                <1> m_pix_op_add:
7424                                <1>         ; 06/02/2021
7425                                <1>         ; ADD COLOR (MASKED, full screen)
7426                                <1>         ;
7427                                <1>         ; jump from pix_op_add
7428                                <1>         ;
7429                                <1>         ; INPUT:
7430                                <1>         ;   eax = color (AL, AX, EAX)
7431                                <1>         ;   ecx = [v_siz] ; display page pixel count
7432                                <1>         ;   esi = edi = [v_mem] ; LFB start address
7433                                <1>         ;
7434                                <1>         ;   [maskcolor] = mask color (to be excluded)
7435                                <1>         ;
7436                                <1>         ; OUTPUT:
7437                                <1>         ;       [u.r0] will be > 0 if succesful
7438                                <1>
7439                                <1>         ; Full screen
7440                                <1> m_pix_op_add_0:
7441 0000F949 803D[89120300]08      <1>         cmp    byte [v_bpp], 8 ; 8bpp
7442 0000F950 771C                 <1>         ja     short m_pix_op_add_1
7443                                <1>         ; 256 colors (8bpp)
7444                                <1>         ;jmp   short m_pix_op_add_8
7445                                <1> m_pix_op_add_8:
7446                                <1>         ; 8 bit colors (256 colors)
7447 0000F952 88C2                 <1>         mov    dl, al ; new color
7448                                <1> m_pix_op_add_8_0:
7449 0000F954 AC                   <1>         lodsb
7450 0000F955 3A05[9A120300]       <1>         cmp    al, [maskcolor]
7451 0000F95B 740D                 <1>         je     short m_pix_op_add_8_1 ; exclude
7452 0000F95D FF05[64030300]       <1>         inc    dword [u.r0] ; +1
7453 0000F963 0017                 <1>         add    [edi], dl
7454 0000F965 7303                 <1>         jnc    short m_pix_op_add_8_1
7455 0000F967 C607FF               <1>         mov    byte [edi], 0FFh
7456                                <1> m_pix_op_add_8_1:
7457 0000F96A 47                   <1>         inc    edi
7458 0000F96B E2E7                 <1>         loop   m_pix_op_add_8_0
7459 0000F96D C3                   <1>         retn
7460                                <1> m_pix_op_add_1:
7461 0000F96E 803D[89120300]18     <1>         cmp    byte [v_bpp], 24 ; 24bpp
7462 0000F975 775E                 <1>         ja     short m_pix_op_add_3 ; 32bpp
7463 0000F977 7238                 <1>         jb     short m_pix_op_add_2 ; 16bpp
7464                                <1>         ; 24 bit true colors
7465                                <1>         ;jmp   short m_pix_op_add_24
7466                                <1> m_pix_op_add_24:
7467                                <1>         ; 24 bit true colors
7468 0000F979 89C2                 <1>         mov    edx, eax ; new color
7469 0000F97B 81CA000000FF         <1>         or     edx, 0FF000000h
7470                                <1> m_pix_op_add_24_0:
7471 0000F981 66AD                 <1>         lodsw
7472 0000F983 C1E010               <1>         shl    eax, 16
7473 0000F986 AC                   <1>         lodsb
7474 0000F987 C1C010               <1>         rol    eax, 16
7475 0000F98A 3B05[9A120300]       <1>         cmp    eax, [maskcolor]
7476 0000F990 7419                 <1>         je     short m_pix_op_add_24_2 ; exclude
7477 0000F992 8305[64030300]03     <1>         add    dword [u.r0], 3 ; +3
7478 0000F999 01D0                 <1>         add    eax, edx
7479 0000F99B 7305                 <1>         jnc    short m_pix_op_add_24_1
7480 0000F99D B8FFFFFF00           <1>         mov    eax, 0FFFFFFh
7481                                <1> m_pix_op_add_24_1:
7482 0000F9A2 668907               <1>         mov    [edi], ax
7483 0000F9A5 C1E810               <1>         shr    eax, 16
7484 0000F9A8 884702               <1>         mov    [edi+2], al
7485                                <1> m_pix_op_add_24_2:
7486 0000F9AB 83C703               <1>         add    edi, 3 ; +3
7487 0000F9AE E2D1                 <1>         loop   m_pix_op_add_24_0
7488 0000F9B0 C3                   <1>         retn
7489                                <1>         ; 65536 colors (16bpp)
7490                                <1> m_pix_op_add_2:
7491                                <1>         ;jmp   short m_pix_op_add_16
7492                                <1> m_pix_op_add_16:
7493                                <1>         ; 16 bit colors (65536 colors)
7494 0000F9B1 89C2                 <1>         mov    edx, eax ; new color
7495                                <1> m_pix_op_add_16_0:
7496 0000F9B3 66AD                 <1>         lodsw
7497 0000F9B5 663B05[9A120300]     <1>         cmp    ax, [maskcolor]
7498 0000F9BC 7411                 <1>         je     short m_pix_op_add_16_1 ; exclude
7499 0000F9BE 8305[64030300]02     <1>         add    dword [u.r0], 2 ; +2
7500 0000F9C5 660117               <1>         add    [edi], dx
7501 0000F9C8 7305                 <1>         jnc    short m_pix_op_add_16_1
7502 0000F9CA 66C707FFFF           <1>         mov    word [edi], 0FFFFh
7503                                <1> m_pix_op_add_16_1:
7504 0000F9CF 83C702               <1>         add    edi, 2 ; +2
7505 0000F9D2 E2DF                 <1>         loop   m_pix_op_add_16_0
7506 0000F9D4 C3                   <1>         retn
7507                                <1> m_pix_op_add_3:
7508                                <1>         ; 32 bit true colors
7509                                <1>         ;jmp   short m_pix_op_add_32
7510                                <1> m_pix_op_add_32:
7511                                <1>         ; 32 bit true colors
```

```
7512 0000F9D5 89C2                <1>       mov    edx, eax ; new color
7513                              <1> m_pix_op_add_32_0:
7514 0000F9D7 AD                  <1>       lodsd
7515 0000F9D8 3B05[9A120300]      <1>       cmp    eax, [maskcolor]
7516 0000F9DE 7411                <1>       je     short m_pix_op_add_32_1 ; exclude
7517 0000F9E0 8305[64030300]04    <1>       add    dword [u.r0], 4 ; +4
7518 0000F9E7 0117                <1>       add    [edi], edx
7519 0000F9E9 7306                <1>       jnc    short m_pix_op_add_32_1
7520 0000F9EB C707FFFFFFFF        <1>       mov    dword [edi], 0FFFFFFFFh
7521                              <1> m_pix_op_add_32_1:
7522 0000F9F1 83C704              <1>       add    edi, 4 ; +4
7523 0000F9F4 E2E1                <1>       loop   m_pix_op_add_32_0
7524 0000F9F6 C3                  <1>       retn
7525                              <1>
7526                              <1> m_pix_op_add_w:
7527                              <1>       ; 06/02/2021
7528                              <1>       ; ADD COLOR (MASKED, window)
7529                              <1>       ;
7530                              <1>       ; jump from pix_op_add_w
7531                              <1>       ;
7532                              <1>       ; INPUT:
7533                              <1>       ;   ecx = bytes per row (to be applied)
7534                              <1>       ;   edx = screen width in bytes
7535                              <1>       ;   ebx = row count
7536                              <1>       ;   eax = color
7537                              <1>       ;
7538                              <1>       ;   [maskcolor] = mask color (to be excluded)
7539                              <1>       ;
7540                              <1>       ; OUTPUT:
7541                              <1>       ;      [u.r0] will be > 0 if succesful
7542                              <1>
7543                              <1>       ; window
7544                              <1>       ;mov   edi, [v_str] ; LFB start address
7545                              <1>       ;mov   esi, edi
7546                              <1>
7547 0000F9F7 803D[89120300]08    <1>       cmp    byte [v_bpp], 8 ; 8bpp
7548 0000F9FE 7707                <1>       ja     short m_pix_op_add_w_1
7549                              <1>
7550                              <1>       ; 256 colors (8bpp)
7551 0000FA00 BD[52F90000]        <1>       mov    ebp, m_pix_op_add_8
7552 0000FA05 EB1E                <1>       jmp    short m_pix_op_add_w_4
7553                              <1>
7554                              <1> m_pix_op_add_w_1:
7555 0000FA07 803D[89120300]18    <1>       cmp    byte [v_bpp], 24 ; 24bpp
7556 0000FA0E 7710                <1>       ja     short m_pix_op_add_w_3 ; 32bpp
7557 0000FA10 7207                <1>       jb     short m_pix_op_add_w_2 ; 16bpp
7558                              <1>
7559                              <1>       ; 24 bit true colors
7560 0000FA12 BD[79F90000]        <1>       mov    ebp, m_pix_op_add_24
7561 0000FA17 EB0C                <1>       jmp    short m_pix_op_add_w_4
7562                              <1>
7563                              <1>       ; 65536 colors (16bpp)
7564                              <1> m_pix_op_add_w_2:
7565 0000FA19 BD[B1F90000]        <1>       mov    ebp, m_pix_op_add_16
7566 0000FA1E EB05                <1>       jmp    short m_pix_op_add_w_4
7567                              <1>
7568                              <1>       ; 32 bit true colors
7569                              <1> m_pix_op_add_w_3:
7570 0000FA20 BD[D5F90000]        <1>       mov    ebp, m_pix_op_add_32
7571                              <1> m_pix_op_add_w_4:
7572 0000FA25 E905FFFFFF          <1>       jmp    m_pix_op_add_w_x
7573                              <1>
7574                              <1> m_pix_op_sub:
7575                              <1>       ; 06/02/2021
7576                              <1>       ; SUBTRACT COLOR (MASKED, full screen)
7577                              <1>       ;
7578                              <1>       ; jump from pix_op_sub
7579                              <1>       ;
7580                              <1>       ; INPUT:
7581                              <1>       ;   eax = color (AL, AX, EAX)
7582                              <1>       ;   ecx = [v_siz] ; display page pixel count
7583                              <1>       ;   esi = edi = [v_mem] ; LFB start address
7584                              <1>       ;
7585                              <1>       ;   [maskcolor] = mask color (to be excluded)
7586                              <1>       ;
7587                              <1>       ; OUTPUT:
7588                              <1>       ;      [u.r0] will be > 0 if succesful
7589                              <1>
7590                              <1>       ; Full screen
7591                              <1> m_pix_op_sub_0:
7592 0000FA2A 803D[89120300]08    <1>       cmp    byte [v_bpp], 8 ; 8bpp
7593 0000FA31 771C                <1>       ja     short m_pix_op_sub_1
7594                              <1>       ; 256 colors (8bpp)
7595                              <1>       ;jmp   short m_pix_op_sub_8
7596                              <1> m_pix_op_sub_8:
7597                              <1>       ; 8 bit colors (256 colors)
7598 0000FA33 88C2                <1>       mov    dl, al ; new color
7599                              <1> m_pix_op_sub_8_0:
7600 0000FA35 AC                  <1>       lodsb
7601 0000FA36 3A05[9A120300]      <1>       cmp    al, [maskcolor]
7602 0000FA3C 740D                <1>       je     short m_pix_op_sub_8_1 ; exclude
7603 0000FA3E FF05[64030300]      <1>       inc    dword [u.r0] ; +1
7604 0000FA44 2817                <1>       sub    [edi], dl
7605 0000FA46 7303                <1>       jnb    short m_pix_op_sub_8_1
7606 0000FA48 C60700              <1>       mov    byte [edi], 0
7607                              <1> m_pix_op_sub_8_1:
7608 0000FA4B 47                  <1>       inc    edi
7609 0000FA4C E2E7                <1>       loop   m_pix_op_sub_8_0
7610 0000FA4E C3                  <1>       retn
7611                              <1> m_pix_op_sub_1:
7612 0000FA4F 803D[89120300]18    <1>       cmp    byte [v_bpp], 24 ; 24bpp
7613 0000FA56 775B                <1>       ja     short m_pix_op_sub_3 ; 32bpp
7614 0000FA58 7235                <1>       jb     short m_pix_op_sub_2 ; 16bpp
7615                              <1>       ; 24 bit true colors
7616                              <1>       ;jmp   short m_pix_op_sub_24
```

```
                            <1> m_pix_op_sub_24:
                            <1>       ; 24 bit true colors
0000FA5A 89C2               <1>       mov    edx, eax ; new color
0000FA5C 81CA000000FF       <1>       or     edx, 0FF000000h
                            <1> m_pix_op_sub_24_0:
0000FA62 66AD               <1>       lodsw
0000FA64 C1E010             <1>       shl    eax, 16
0000FA67 AC                 <1>       lodsb
0000FA68 C1C010             <1>       rol    eax, 16
0000FA6B 3B05[9A120300]     <1>       cmp    eax, [maskcolor]
0000FA71 7416               <1>       je     short m_pix_op_sub_24_2 ; exclude
0000FA73 8305[64030300]03   <1>       add    dword [u.r0], 3 ; +3
0000FA7A 29D0               <1>       sub    eax, edx
0000FA7C 7302               <1>       jnb    short m_pix_op_sub_24_1
0000FA7E 31C0               <1>       xor    eax, eax ; 0
                            <1> m_pix_op_sub_24_1:
0000FA80 668907             <1>       mov    [edi], ax
0000FA83 C1E810             <1>       shr    eax, 16
0000FA86 884702             <1>       mov    [edi+2], al
                            <1> m_pix_op_sub_24_2:
0000FA89 83C703             <1>       add    edi, 3 ; +3
0000FA8C E2D4               <1>       loop   m_pix_op_sub_24_0
0000FA8E C3                 <1>       retn
                            <1>       ; 65536 colors (16bpp)
                            <1> m_pix_op_sub_2:
                            <1>       ;jmp   short m_pix_op_sub_16
                            <1> m_pix_op_sub_16:
                            <1>       ; 16 bit colors (65536 colors)
0000FA8F 89C2               <1>       mov    edx, eax ; new color
                            <1> m_pix_op_sub_16_0:
0000FA91 66AD               <1>       lodsw
0000FA93 663B05[9A120300]   <1>       cmp    ax, [maskcolor]
0000FA9A 7411               <1>       je     short m_pix_op_sub_16_1 ; exclude
0000FA9C 8305[64030300]02   <1>       add    dword [u.r0], 2 ; +2
0000FAA3 662917             <1>       sub    [edi], dx
0000FAA6 7305               <1>       jnb    short m_pix_op_sub_16_1
0000FAA8 31C0               <1>       xor    eax, eax
0000FAAA 668907             <1>       mov    [edi], ax ; 0
                            <1> m_pix_op_sub_16_1:
0000FAAD 83C702             <1>       add    edi, 2 ; +2
0000FAB0 E2DF               <1>       loop   m_pix_op_sub_16_0
0000FAB2 C3                 <1>       retn
                            <1> m_pix_op_sub_3:
                            <1>       ; 32 bit true colors
                            <1>       ;jmp   short m_pix_op_sub_32
                            <1> m_pix_op_sub_32:
                            <1>       ; 32 bit true colors
0000FAB3 89C2               <1>       mov    edx, eax ; new color
                            <1> m_pix_op_sub_32_0:
0000FAB5 AD                 <1>       lodsd
0000FAB6 3B05[9A120300]     <1>       cmp    eax, [maskcolor]
0000FABC 740F               <1>       je     short m_pix_op_sub_32_1 ; exclude
0000FABE 8305[64030300]04   <1>       add    dword [u.r0], 4 ; +4
0000FAC5 2917               <1>       sub    [edi], edx
0000FAC7 7304               <1>       jnb    short m_pix_op_sub_32_1
0000FAC9 31C0               <1>       xor    eax, eax
0000FACB 8907               <1>       mov    [edi], eax ; 0
                            <1> m_pix_op_sub_32_1:
0000FACD 83C704             <1>       add    edi, 4 ; +4
0000FAD0 E2E3               <1>       loop   m_pix_op_sub_32_0
0000FAD2 C3                 <1>       retn
                            <1>
                            <1> m_pix_op_sub_w:
                            <1>       ; 06/02/2021
                            <1>       ; SUBTRACT COLOR (MASKED, window)
                            <1>       ;
                            <1>       ; jump from pix_op_sub_w
                            <1>       ;
                            <1>       ; INPUT:
                            <1>       ;   ecx = bytes per row (to be applied)
                            <1>       ;   edx = screen width in bytes
                            <1>       ;   ebx = row count
                            <1>       ;   eax = color
                            <1>       ;
                            <1>       ;   [maskcolor] = mask color (to be excluded)
                            <1>       ;
                            <1>       ; OUTPUT:
                            <1>       ;      [u.r0] will be > 0 if succesful
                            <1>
                            <1>       ; window
                            <1>       ;mov   edi, [v_str] ; LFB start address
                            <1>       ;mov   esi, edi
                            <1>
0000FAD3 803D[89120300]08   <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000FADA 7707               <1>       ja     short m_pix_op_sub_w_1
                            <1>
                            <1>       ; 256 colors (8bpp)
0000FADC BD[33FA0000]       <1>       mov    ebp, m_pix_op_sub_8
0000FAE1 EB1E               <1>       jmp    short m_pix_op_sub_w_4
                            <1>
                            <1> m_pix_op_sub_w_1:
0000FAE3 803D[89120300]18   <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000FAEA 7710               <1>       ja     short m_pix_op_sub_w_3 ; 32bpp
0000FAEC 7207               <1>       jb     short m_pix_op_sub_w_2 ; 16bpp
                            <1>
                            <1>       ; 24 bit true colors
0000FAEE BD[5AFA0000]       <1>       mov    ebp, m_pix_op_sub_24
0000FAF3 EB0C               <1>       jmp    short m_pix_op_sub_w_4
                            <1>
                            <1>       ; 65536 colors (16bpp)
                            <1> m_pix_op_sub_w_2:
0000FAF5 BD[8FFA0000]       <1>       mov    ebp, m_pix_op_sub_16
0000FAFA EB05               <1>       jmp    short m_pix_op_sub_w_4
                            <1>
                            <1>       ; 32 bit true colors
```

```
                                <1> m_pix_op_sub_w_3:
0000FAFC BD[B3FA0000]           <1>       mov    ebp, m_pix_op_sub_32
                                <1> m_pix_op_sub_w_4:
0000FB01 E929FEFFFF             <1>       jmp    m_pix_op_sub_w_x
                                <1>
                                <1> m_pix_op_mix:
                                <1>       ; 06/02/2021
                                <1>       ; MIX COLOR (MASKED, full screen)
                                <1>       ;
                                <1>       ; jump from pix_op_mix
                                <1>       ;
                                <1>       ; INPUT:
                                <1>       ;   eax = color (AL, AX, EAX)
                                <1>       ;   ecx = [v_siz] ; display page pixel count
                                <1>       ;   esi = edi = [v_mem] ; LFB start address
                                <1>       ;
                                <1>       ;   [maskcolor] = mask color (to be excluded)
                                <1>       ;
                                <1>       ; OUTPUT:
                                <1>       ;     [u.r0] will be > 0 if succesful
                                <1>
                                <1>       ; Full screen
                                <1> m_pix_op_mix_0:
0000FB06 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000FB0D 771B                   <1>       ja     short m_pix_op_mix_1
                                <1>       ; 256 colors (8bpp)
                                <1>       ;jmp   short m_pix_op_mix_8
                                <1> m_pix_op_mix_8:
                                <1>       ; 8 bit colors (256 colors)
0000FB0F 88C2                   <1>       mov    dl, al ; new color
                                <1> m_pix_op_mix_8_0:
0000FB11 AC                     <1>       lodsb
0000FB12 3A05[9A120300]         <1>       cmp    al, [maskcolor]
0000FB18 740C                   <1>       je     short m_pix_op_mix_8_1 ; exclude
0000FB1A 0207                   <1>       add    al, [edi]
0000FB1C D0D8                   <1>       rcr    al, 1
0000FB1E 8807                   <1>       mov    [edi], al
0000FB20 FF05[64030300]         <1>       inc    dword [u.r0] ; +1
                                <1> m_pix_op_mix_8_1:
0000FB26 47                     <1>       inc    edi
0000FB27 E2E8                   <1>       loop   m_pix_op_mix_8_0
0000FB29 C3                     <1>       retn
                                <1> m_pix_op_mix_1:
0000FB2A 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000FB31 7752                   <1>       ja     short m_pix_op_mix_3 ; 32bpp
0000FB33 722D                   <1>       jb     short m_pix_op_mix_2 ; 16bpp
                                <1>       ; 24 bit true colors
                                <1>       ;jmp   short m_pix_op_mix_24
                                <1> m_pix_op_mix_24:
                                <1>       ; 24 bit true colors
0000FB35 89C2                   <1>       mov    edx, eax ; new color
                                <1>       ;and   edx, 0FFFFFFh
                                <1> m_pix_op_mix_24_0:
0000FB37 66AD                   <1>       lodsw
0000FB39 C1E010                 <1>       shl    eax, 16
0000FB3C AC                     <1>       lodsb
0000FB3D C1C010                 <1>       rol    eax, 16
0000FB40 3B05[9A120300]         <1>       cmp    eax, [maskcolor]
0000FB46 7414                   <1>       je     short m_pix_op_mix_24_1 ; exclude
0000FB48 01D0                   <1>       add    eax, edx
0000FB4A D1E8                   <1>       shr    eax, 1
0000FB4C 668907                 <1>       mov    [edi], ax
0000FB4F C1E810                 <1>       shr    eax, 16
0000FB52 884702                 <1>       mov    [edi+2], al
0000FB55 8305[64030300]03       <1>       add    dword [u.r0], 3 ; +3
                                <1> m_pix_op_mix_24_1:
0000FB5C 83C703                 <1>       add    edi, 3 ; +3
0000FB5F E2D6                   <1>       loop   m_pix_op_mix_24_0
0000FB61 C3                     <1>       retn
                                <1>       ; 65536 colors (16bpp)
                                <1> m_pix_op_mix_2:
                                <1>       ;jmp   short m_pix_op_mix_16
                                <1> m_pix_op_mix_16:
                                <1>       ; 16 bit colors (65536 colors)
0000FB62 89C2                   <1>       mov    edx, eax ; new color
                                <1>       ;and   edx, 0FFFFh
                                <1> m_pix_op_mix_16_0:
0000FB64 66AD                   <1>       lodsw
0000FB66 663B05[9A120300]       <1>       cmp    ax, [maskcolor]
0000FB6D 7410                   <1>       je     short m_pix_op_mix_16_1 ; exclude
0000FB6F 6601D0                 <1>       add    ax, dx
0000FB72 66D1D8                 <1>       rcr    ax, 1
0000FB75 668907                 <1>       mov    [edi], ax
0000FB78 8305[64030300]02       <1>       add    dword [u.r0], 2 ; +2
                                <1> m_pix_op_mix_16_1:
0000FB7F 83C702                 <1>       add    edi, 2 ; +2
0000FB82 E2E0                   <1>       loop   m_pix_op_mix_16_0
0000FB84 C3                     <1>       retn
                                <1> m_pix_op_mix_3:
                                <1>       ; 32 bit true colors
                                <1>       ;jmp   short m_pix_op_mix_32
                                <1> m_pix_op_mix_32:
                                <1>       ; 32 bit true colors
0000FB85 89C2                   <1>       mov    edx, eax ; new color
                                <1> m_pix_op_mix_32_0:
0000FB87 AD                     <1>       lodsd
0000FB88 3B05[9A120300]         <1>       cmp    eax, [maskcolor]
0000FB8E 740F                   <1>       je     short m_pix_op_mix_32_1 ; exclude
0000FB90 01D0                   <1>       add    eax, edx
0000FB92 730B                   <1>       jnc    short m_pix_op_mix_32_1
0000FB94 D1D8                   <1>       rcr    eax, 1
0000FB96 8907                   <1>       mov    [edi], eax
0000FB98 8305[64030300]04       <1>       add    dword [u.r0], 4 ; +4
                                <1> m_pix_op_mix_32_1:
0000FB9F 83C704                 <1>       add    edi, 4 ; +4
```

```
7827 0000FBA2 E2E3              <1>       loop   m_pix_op_mix_32_0
7828 0000FBA4 C3                <1>       retn
7829                            <1>
7830                            <1> m_pix_op_mix_w:
7831                            <1>       ; 06/02/2021
7832                            <1>       ; MIX COLOR (MASKED, window)
7833                            <1>       ;
7834                            <1>       ; jump from pix_op_mix_w
7835                            <1>       ;
7836                            <1>       ; INPUT:
7837                            <1>       ;   ecx = bytes per row (to be applied)
7838                            <1>       ;   edx = screen width in bytes
7839                            <1>       ;   ebx = row count
7840                            <1>       ;   eax = color
7841                            <1>       ;
7842                            <1>       ;   [maskcolor] = mask color (to be excluded)
7843                            <1>       ;
7844                            <1>       ; OUTPUT:
7845                            <1>       ;      [u.r0] will be > 0 if succesful
7846                            <1>
7847                            <1>       ; window
7848                            <1>       ;mov   edi, [v_str] ; LFB start address
7849                            <1>       ;mov   esi, edi
7850                            <1>
7851 0000FBA5 803D[89120300]08 <1>       cmp    byte [v_bpp], 8 ; 8bpp
7852 0000FBAC 7707             <1>       ja     short m_pix_op_mix_w_1
7853                            <1>
7854                            <1>       ; 256 colors (8bpp)
7855 0000FBAE BD[0FFB0000]     <1>       mov    ebp, m_pix_op_mix_8
7856 0000FBB3 EB1E             <1>       jmp    short m_pix_op_mix_w_4
7857                            <1>
7858                            <1> m_pix_op_mix_w_1:
7859 0000FBB5 803D[89120300]18 <1>       cmp    byte [v_bpp], 24 ; 24bpp
7860 0000FBBC 7710             <1>       ja     short m_pix_op_mix_w_3 ; 32bpp
7861 0000FBBE 7207             <1>       jb     short m_pix_op_mix_w_2 ; 16bpp
7862                            <1>
7863                            <1>       ; 24 bit true colors
7864 0000FBC0 BD[35FB0000]     <1>       mov    ebp, m_pix_op_mix_24
7865 0000FBC5 EB0C             <1>       jmp    short m_pix_op_mix_w_4
7866                            <1>
7867                            <1>       ; 65536 colors (16bpp)
7868                            <1> m_pix_op_mix_w_2:
7869 0000FBC7 BD[62FB0000]     <1>       mov    ebp, m_pix_op_mix_16
7870 0000FBCC EB05             <1>       jmp    short m_pix_op_mix_w_4
7871                            <1>
7872                            <1>       ; 32 bit true colors
7873                            <1> m_pix_op_mix_w_3:
7874 0000FBCE BD[85FB0000]     <1>       mov    ebp, m_pix_op_mix_32
7875                            <1> m_pix_op_mix_w_4:
7876 0000FBD3 E957FDFFFF       <1>       jmp    m_pix_op_mix_w_x
7877                            <1>
7878                            <1> m_pix_op_and:
7879                            <1>       ; 06/02/2021
7880                            <1>       ; AND COLOR (MASKED, full screen)
7881                            <1>       ;
7882                            <1>       ; jump from pix_op_and
7883                            <1>       ;
7884                            <1>       ; INPUT:
7885                            <1>       ;   eax = color (AL, AX, EAX)
7886                            <1>       ;   ecx = [v_siz] ; display page pixel count
7887                            <1>       ;   esi = edi = [v_mem] ; LFB start address
7888                            <1>       ;
7889                            <1>       ;   [maskcolor] = mask color (to be excluded)
7890                            <1>       ;
7891                            <1>       ; OUTPUT:
7892                            <1>       ;      [u.r0] will be > 0 if succesful
7893                            <1>
7894                            <1>       ; Full screen
7895                            <1> m_pix_op_and_0:
7896 0000FBD8 803D[89120300]08 <1>       cmp    byte [v_bpp], 8 ; 8bpp
7897 0000FBDF 7717             <1>       ja     short m_pix_op_and_1
7898                            <1>       ; 256 colors (8bpp)
7899                            <1>       ;jmp   short m_pix_op_and_8
7900                            <1> m_pix_op_and_8:
7901                            <1>       ; 8 bit colors (256 colors)
7902 0000FBE1 88C2              <1>       mov    dl, al ; new color
7903                            <1> m_pix_op_and_8_0:
7904 0000FBE3 AC                <1>       lodsb
7905 0000FBE4 3A05[9A120300]   <1>       cmp    al, [maskcolor]
7906 0000FBEA 7408             <1>       je     short m_pix_op_and_8_1 ; exclude
7907 0000FBEC 2017             <1>       and    [edi], dl
7908 0000FBEE FF05[64030300]   <1>       inc    dword [u.r0] ; +1
7909                            <1> m_pix_op_and_8_1:
7910 0000FBF4 47                <1>       inc    edi
7911 0000FBF5 E2EC              <1>       loop   m_pix_op_and_8_0
7912 0000FBF7 C3                <1>       retn
7913                            <1> m_pix_op_and_1:
7914 0000FBF8 803D[89120300]18 <1>       cmp    byte [v_bpp], 24 ; 24bpp
7915 0000FBFF 774A             <1>       ja     short m_pix_op_and_3 ; 32bpp
7916 0000FC01 722B             <1>       jb     short m_pix_op_and_2 ; 16bpp
7917                            <1>       ; 24 bit true colors
7918                            <1>       ;jmp   short m_pix_op_and_24
7919                            <1> m_pix_op_and_24:
7920                            <1>       ; 24 bit true colors
7921 0000FC03 89C2              <1>       mov    edx, eax ; new color
7922                            <1>       ;and   edx, 0FFFFFFh
7923                            <1> m_pix_op_and_24_0:
7924 0000FC05 66AD              <1>       lodsw
7925 0000FC07 C1E010            <1>       shl    eax, 16
7926 0000FC0A AC                <1>       lodsb
7927 0000FC0B C1C010            <1>       rol    eax, 16
7928 0000FC0E 3B05[9A120300]   <1>       cmp    eax, [maskcolor]
7929 0000FC14 7412             <1>       je     short m_pix_op_and_24_1 ; exclude
7930 0000FC16 21D0             <1>       and    eax, edx
7931 0000FC18 668907           <1>       mov    [edi], ax
```

```
7932 0000FC1B C1E810              <1>       shr    eax, 16
7933 0000FC1E 884702              <1>       mov    [edi+2], al
7934 0000FC21 8305[64030300]03    <1>       add    dword [u.r0], 3 ; +3
7935                              <1> m_pix_op_and_24_1:
7936 0000FC28 83C703              <1>       add    edi, 3 ; +3
7937 0000FC2B E2D8                <1>       loop   m_pix_op_and_24_0
7938 0000FC2D C3                  <1>       retn
7939                              <1>       ; 65536 colors (16bpp)
7940                              <1> m_pix_op_and_2:
7941                              <1>       ;jmp   short m_pix_op_and_16
7942                              <1> m_pix_op_and_16:
7943                              <1>       ; 16 bit colors (65536 colors)
7944 0000FC2E 89C2                <1>       mov    edx, eax ; new color
7945                              <1>       ;and   edx, 0FFFFh
7946                              <1> m_pix_op_and_16_0:
7947 0000FC30 66AD                <1>       lodsw
7948 0000FC32 663B05[9A120300]    <1>       cmp    ax, [maskcolor]
7949 0000FC39 740A                <1>       je     short m_pix_op_and_16_1 ; exclude
7950 0000FC3B 662117              <1>       and    [edi], dx
7951 0000FC3E 8305[64030300]02    <1>       add    dword [u.r0], 2 ; +2
7952                              <1> m_pix_op_and_16_1:
7953 0000FC45 83C702              <1>       add    edi, 2 ; +2
7954 0000FC48 E2E6                <1>       loop   m_pix_op_and_16_0
7955 0000FC4A C3                  <1>       retn
7956                              <1> m_pix_op_and_3:
7957                              <1>       ; 32 bit true colors
7958                              <1>       ;jmp   short m_pix_op_and_32
7959                              <1> m_pix_op_and_32:
7960                              <1>       ; 32 bit true colors
7961 0000FC4B 89C2                <1>       mov    edx, eax ; new color
7962                              <1> m_pix_op_and_32_0:
7963 0000FC4D AD                  <1>       lodsd
7964 0000FC4E 3B05[9A120300]      <1>       cmp    eax, [maskcolor]
7965 0000FC54 7409                <1>       je     short m_pix_op_and_32_1 ; exclude
7966 0000FC56 2107                <1>       and    [edi], eax
7967 0000FC58 8305[64030300]04    <1>       add    dword [u.r0], 4 ; +4
7968                              <1> m_pix_op_and_32_1:
7969 0000FC5F 83C704              <1>       add    edi, 4 ; +4
7970 0000FC62 E2E9                <1>       loop   m_pix_op_and_32_0
7971 0000FC64 C3                  <1>       retn
7972                              <1>
7973                              <1> m_pix_op_and_w:
7974                              <1>       ; 06/02/2021
7975                              <1>       ; AND COLOR (MASKED, window)
7976                              <1>       ;
7977                              <1>       ; jump from pix_op_and_w
7978                              <1>       ;
7979                              <1>       ; INPUT:
7980                              <1>       ;   ecx = bytes per row (to be applied)
7981                              <1>       ;   edx = screen width in bytes
7982                              <1>       ;   ebx = row count
7983                              <1>       ;   eax = color
7984                              <1>       ;
7985                              <1>       ;   [maskcolor] = mask color (to be excluded)
7986                              <1>       ;
7987                              <1>       ; OUTPUT:
7988                              <1>       ;      [u.r0] will be > 0 if succesful
7989                              <1>
7990                              <1>       ; window
7991                              <1>       ;mov   edi, [v_str] ; LFB start address
7992                              <1>       ;mov   esi, edi
7993                              <1>
7994 0000FC65 803D[89120300]08    <1>       cmp    byte [v_bpp], 8 ; 8bpp
7995 0000FC6C 7707                <1>       ja     short m_pix_op_and_w_1
7996                              <1>
7997                              <1>       ; 256 colors (8bpp)
7998 0000FC6E BD[E1FB0000]        <1>       mov    ebp, m_pix_op_and_8
7999 0000FC73 EB1E                <1>       jmp    short m_pix_op_and_w_4
8000                              <1>
8001                              <1> m_pix_op_and_w_1:
8002 0000FC75 803D[89120300]18    <1>       cmp    byte [v_bpp], 24 ; 24bpp
8003 0000FC7C 7710                <1>       ja     short m_pix_op_and_w_3 ; 32bpp
8004 0000FC7E 7207                <1>       jb     short m_pix_op_and_w_2 ; 16bpp
8005                              <1>
8006                              <1>       ; 24 bit true colors
8007 0000FC80 BD[03FC0000]        <1>       mov    ebp, m_pix_op_and_24
8008 0000FC85 EB0C                <1>       jmp    short m_pix_op_and_w_4
8009                              <1>
8010                              <1>       ; 65536 colors (16bpp)
8011                              <1> m_pix_op_and_w_2:
8012 0000FC87 BD[2EFC0000]        <1>       mov    ebp, m_pix_op_and_16
8013 0000FC8C EB05                <1>       jmp    short m_pix_op_and_w_4
8014                              <1>
8015                              <1>       ; 32 bit true colors
8016                              <1> m_pix_op_and_w_3:
8017 0000FC8E BD[4BFC0000]        <1>       mov    ebp, m_pix_op_and_32
8018                              <1> m_pix_op_and_w_4:
8019 0000FC93 E997FCFFFF          <1>       jmp    m_pix_op_and_w_x
8020                              <1>
8021                              <1> m_pix_op_or:
8022                              <1>       ; 06/02/2021
8023                              <1>       ; OR COLOR (MASKED, full screen)
8024                              <1>       ;
8025                              <1>       ; jump from pix_op_orc
8026                              <1>       ;
8027                              <1>       ; INPUT:
8028                              <1>       ;   eax = color (AL, AX, EAX)
8029                              <1>       ;   ecx = [v_siz] ; display page pixel count
8030                              <1>       ;   esi = edi = [v_mem] ; LFB start address
8031                              <1>       ;
8032                              <1>       ;   [maskcolor] = mask color (to be excluded)
8033                              <1>       ;
8034                              <1>       ; OUTPUT:
8035                              <1>       ;      [u.r0] will be > 0 if succesful
8036                              <1>
```

```
                                <1>       ; Full screen
                                <1> m_pix_op_or_0:
0000FC98 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000FC9F 7717                   <1>       ja     short m_pix_op_or_1
                                <1>       ; 256 colors (8bpp)
                                <1>       ;jmp   short m_pix_op_or_8
                                <1> m_pix_op_or_8:
                                <1>       ; 8 bit colors (256 colors)
0000FCA1 88C2                   <1>       mov    dl, al ; new color
                                <1> m_pix_op_or_8_0:
0000FCA3 AC                     <1>       lodsb
0000FCA4 3A05[9A120300]         <1>       cmp    al, [maskcolor]
0000FCAA 7408                   <1>       je     short m_pix_op_or_8_1 ; exclude
0000FCAC 0817                   <1>       or     [edi], dl
0000FCAE FF05[64030300]         <1>       inc    dword [u.r0] ; +1
                                <1> m_pix_op_or_8_1:
0000FCB4 47                     <1>       inc    edi
0000FCB5 E2EC                   <1>       loop   m_pix_op_or_8_0
0000FCB7 C3                     <1>       retn
                                <1> m_pix_op_or_1:
0000FCB8 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000FCBF 774A                   <1>       ja     short m_pix_op_or_3 ; 32bpp
0000FCC1 722B                   <1>       jb     short m_pix_op_or_2 ; 16bpp
                                <1>       ; 24 bit true colors
                                <1>       ;jmp   short m_pix_op_or_24
                                <1> m_pix_op_or_24:
                                <1>       ; 24 bit true colors
0000FCC3 89C2                   <1>       mov    edx, eax ; new color
                                <1>       ;and   edx, 0FFFFFFh
                                <1> m_pix_op_or_24_0:
0000FCC5 66AD                   <1>       lodsw
0000FCC7 C1E010                 <1>       shl    eax, 16
0000FCCA AC                     <1>       lodsb
0000FCCB C1C010                 <1>       rol    eax, 16
0000FCCE 3B05[9A120300]         <1>       cmp    eax, [maskcolor]
0000FCD4 7412                   <1>       je     short m_pix_op_or_24_1 ; exclude
0000FCD6 09D0                   <1>       or     eax, edx
0000FCD8 668907                 <1>       mov    [edi], ax
0000FCDB C1E810                 <1>       shr    eax, 16
0000FCDE 884702                 <1>       mov    [edi+2], al
0000FCE1 8305[64030300]03       <1>       add    dword [u.r0], 3 ; +3
                                <1> m_pix_op_or_24_1:
0000FCE8 83C703                 <1>       add    edi, 3 ; +3
0000FCEB E2D8                   <1>       loop   m_pix_op_or_24_0
0000FCED C3                     <1>       retn
                                <1>       ; 65536 colors (16bpp)
                                <1> m_pix_op_or_2:
                                <1>       ;jmp   short m_pix_op_or_16
                                <1> m_pix_op_or_16:
                                <1>       ; 16 bit colors (65536 colors)
0000FCEE 89C2                   <1>       mov    edx, eax ; new color
                                <1>       ;and   edx, 0FFFFh
                                <1> m_pix_op_or_16_0:
0000FCF0 66AD                   <1>       lodsw
0000FCF2 663B05[9A120300]       <1>       cmp    ax, [maskcolor]
0000FCF9 740A                   <1>       je     short m_pix_op_or_16_1 ; exclude
0000FCFB 660917                 <1>       or     [edi], dx
0000FCFE 8305[64030300]02       <1>       add    dword [u.r0], 2 ; +2
                                <1> m_pix_op_or_16_1:
0000FD05 83C702                 <1>       add    edi, 2 ; +2
0000FD08 E2E6                   <1>       loop   m_pix_op_or_16_0
0000FD0A C3                     <1>       retn
                                <1> m_pix_op_or_3:
                                <1>       ; 32 bit true colors
                                <1>       ;jmp   short m_pix_op_or_32
                                <1> m_pix_op_or_32:
                                <1>       ; 32 bit true colors
0000FD0B 89C2                   <1>       mov    edx, eax ; new color
                                <1> m_pix_op_or_32_0:
0000FD0D AD                     <1>       lodsd
0000FD0E 3B05[9A120300]         <1>       cmp    eax, [maskcolor]
0000FD14 7409                   <1>       je     short m_pix_op_or_32_1 ; exclude
0000FD16 0907                   <1>       or     [edi], eax
0000FD18 8305[64030300]04       <1>       add    dword [u.r0], 4 ; +4
                                <1> m_pix_op_or_32_1:
0000FD1F 83C704                 <1>       add    edi, 4 ; +4
0000FD22 E2E9                   <1>       loop   m_pix_op_or_32_0
0000FD24 C3                     <1>       retn
                                <1>
                                <1> m_pix_op_or_w:
                                <1>       ; 06/02/2021
                                <1>       ; MIX COLOR (MASKED, window)
                                <1>       ;
                                <1>       ; jump from pix_op_or_w
                                <1>       ;
                                <1>       ; INPUT:
                                <1>       ;   ecx = bytes per row (to be applied)
                                <1>       ;   edx = screen width in bytes
                                <1>       ;   ebx = row count
                                <1>       ;   eax = color
                                <1>       ;
                                <1>       ;   [maskcolor] = mask color (to be excluded)
                                <1>       ;
                                <1>       ; OUTPUT:
                                <1>       ;      [u.r0] will be > 0 if succesful
                                <1>
                                <1>       ; window
                                <1>       ;mov   edi, [v_str] ; LFB start address
                                <1>       ;mov   esi, edi
                                <1>
0000FD25 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000FD2C 7707                   <1>       ja     short m_pix_op_or_w_1
                                <1>
                                <1>       ; 256 colors (8bpp)
0000FD2E BD[A1FC0000]           <1>       mov    ebp, m_pix_op_or_8
```

```
8142 0000FD33 EB1E                <1>       jmp    short m_pix_op_or_w_4
8143                              <1>
8144                              <1> m_pix_op_or_w_1:
8145 0000FD35 803D[89120300]18    <1>       cmp    byte [v_bpp], 24 ; 24bpp
8146 0000FD3C 7710                <1>       ja     short m_pix_op_or_w_3 ; 32bpp
8147 0000FD3E 7207                <1>       jb     short m_pix_op_or_w_2 ; 16bpp
8148                              <1>
8149                              <1>       ; 24 bit true colors
8150 0000FD40 BD[C3FC0000]        <1>       mov    ebp, m_pix_op_or_24
8151 0000FD45 EB0C                <1>       jmp    short m_pix_op_or_w_4
8152                              <1>
8153                              <1>       ; 65536 colors (16bpp)
8154                              <1> m_pix_op_or_w_2:
8155 0000FD47 BD[EEFC0000]        <1>       mov    ebp, m_pix_op_or_16
8156 0000FD4C EB05                <1>       jmp    short m_pix_op_or_w_4
8157                              <1>
8158                              <1>       ; 32 bit true colors
8159                              <1> m_pix_op_or_w_3:
8160 0000FD4E BD[0BFD0000]        <1>       mov    ebp, m_pix_op_or_32
8161                              <1> m_pix_op_or_w_4:
8162 0000FD53 E9D7FBFFFF          <1>       jmp    m_pix_op_orc_w_x
8163                              <1>
8164                              <1> m_pix_op_xor:
8165                              <1>       ; 06/02/2021
8166                              <1>       ; XOR COLOR (MASKED, full screen)
8167                              <1>       ;
8168                              <1>       ; jump from pix_op_xor
8169                              <1>       ;
8170                              <1>       ; INPUT:
8171                              <1>       ;   eax = color (AL, AX, EAX)
8172                              <1>       ;   ecx = [v_siz] ; display page pixel count
8173                              <1>       ;   esi = edi = [v_mem] ; LFB start address
8174                              <1>       ;
8175                              <1>       ;   [maskcolor] = mask color (to be excluded)
8176                              <1>       ;
8177                              <1>       ; OUTPUT:
8178                              <1>       ;      [u.r0] will be > 0 if succesful
8179                              <1>
8180                              <1>       ; Full screen
8181                              <1> m_pix_op_xor_0:
8182 0000FD58 803D[89120300]08    <1>       cmp    byte [v_bpp], 8 ; 8bpp
8183 0000FD5F 7717                <1>       ja     short m_pix_op_xor_1
8184                              <1>       ; 256 colors (8bpp)
8185                              <1>       ;jmp   short m_pix_op_xor_8
8186                              <1> m_pix_op_xor_8:
8187                              <1>       ; 8 bit colors (256 colors)
8188 0000FD61 88C2                <1>       mov    dl, al ; new color
8189                              <1> m_pix_op_xor_8_0:
8190 0000FD63 AC                  <1>       lodsb
8191 0000FD64 3A05[9A120300]      <1>       cmp    al, [maskcolor]
8192 0000FD6A 7408                <1>       je     short m_pix_op_xor_8_1 ; exclude
8193 0000FD6C 3017                <1>       xor    [edi], dl
8194 0000FD6E FF05[64030300]      <1>       inc    dword [u.r0] ; +1
8195                              <1> m_pix_op_xor_8_1:
8196 0000FD74 47                  <1>       inc    edi
8197 0000FD75 E2EC                <1>       loop   m_pix_op_xor_8_0
8198 0000FD77 C3                  <1>       retn
8199                              <1> m_pix_op_xor_1:
8200 0000FD78 803D[89120300]18    <1>       cmp    byte [v_bpp], 24 ; 24bpp
8201 0000FD7F 774A                <1>       ja     short m_pix_op_xor_3 ; 32bpp
8202 0000FD81 722B                <1>       jb     short m_pix_op_xor_2 ; 16bpp
8203                              <1>       ; 24 bit true colors
8204                              <1>       ;jmp   short m_pix_op_xor_24
8205                              <1> m_pix_op_xor_24:
8206                              <1>       ; 24 bit true colors
8207 0000FD83 89C2                <1>       mov    edx, eax ; new color
8208                              <1>       ;and   edx, 0FFFFFFh
8209                              <1> m_pix_op_xor_24_0:
8210 0000FD85 66AD                <1>       lodsw
8211 0000FD87 C1E010              <1>       shl    eax, 16
8212 0000FD8A AC                  <1>       lodsb
8213 0000FD8B C1C010              <1>       rol    eax, 16
8214 0000FD8E 3B05[9A120300]      <1>       cmp    eax, [maskcolor]
8215 0000FD94 7412                <1>       je     short m_pix_op_xor_24_1 ; exclude
8216 0000FD96 31D0                <1>       xor    eax, edx
8217 0000FD98 668907              <1>       mov    [edi], ax
8218 0000FD9B C1E810              <1>       shr    eax, 16
8219 0000FD9E 884702              <1>       mov    [edi+2], al
8220 0000FDA1 8305[64030300]03    <1>       add    dword [u.r0], 3 ; +3
8221                              <1> m_pix_op_xor_24_1:
8222 0000FDA8 83C703              <1>       add    edi, 3 ; +3
8223 0000FDAB E2D8                <1>       loop   m_pix_op_xor_24_0
8224 0000FDAD C3                  <1>       retn
8225                              <1>       ; 65536 colors (16bpp)
8226                              <1> m_pix_op_xor_2:
8227                              <1>       ;jmp   short m_pix_op_xor_16
8228                              <1> m_pix_op_xor_16:
8229                              <1>       ; 16 bit colors (65536 colors)
8230 0000FDAE 89C2                <1>       mov    edx, eax ; new color
8231                              <1>       ;and   edx, 0FFFFh
8232                              <1> m_pix_op_xor_16_0:
8233 0000FDB0 66AD                <1>       lodsw
8234 0000FDB2 663B05[9A120300]    <1>       cmp    ax, [maskcolor]
8235 0000FDB9 740A                <1>       je     short m_pix_op_xor_16_1 ; exclude
8236 0000FDBB 663117              <1>       xor    [edi], dx
8237 0000FDBE 8305[64030300]02    <1>       add    dword [u.r0], 2 ; +2
8238                              <1> m_pix_op_xor_16_1:
8239 0000FDC5 83C702              <1>       add    edi, 2 ; +2
8240 0000FDC8 E2E6                <1>       loop   m_pix_op_xor_16_0
8241 0000FDCA C3                  <1>       retn
8242                              <1> m_pix_op_xor_3:
8243                              <1>       ; 32 bit true colors
8244                              <1>       ;jmp   short m_pix_op_xor_32
8245                              <1> m_pix_op_xor_32:
8246                              <1>       ; 32 bit true colors
```

```
8247 0000FDCB 89C2                <1>       mov    edx, eax ; new color
8248                              <1> m_pix_op_xor_32_0:
8249 0000FDCD AD                  <1>       lodsd
8250 0000FDCE 3B05[9A120300]      <1>       cmp    eax, [maskcolor]
8251 0000FDD4 7409                <1>       je     short m_pix_op_xor_32_1 ; exclude
8252 0000FDD6 3107                <1>       xor    [edi], eax
8253 0000FDD8 8305[64030300]04    <1>       add    dword [u.r0], 4 ; +4
8254                              <1> m_pix_op_xor_32_1:
8255 0000FDDF 83C704              <1>       add    edi, 4 ; +4
8256 0000FDE2 E2E9                <1>       loop   m_pix_op_xor_32_0
8257 0000FDE4 C3                  <1>       retn
8258                              <1>
8259                              <1> m_pix_op_xor_w:
8260                              <1>       ; 06/02/2021
8261                              <1>       ; XOR COLOR (MASKED, window)
8262                              <1>       ;
8263                              <1>       ; jump from pix_op_xor_w
8264                              <1>       ;
8265                              <1>       ; INPUT:
8266                              <1>       ;   ecx = bytes per row (to be applied)
8267                              <1>       ;   edx = screen width in bytes
8268                              <1>       ;   ebx = row count
8269                              <1>       ;   eax = color
8270                              <1>       ;
8271                              <1>       ;   [maskcolor] = mask color (to be excluded)
8272                              <1>       ;
8273                              <1>       ; OUTPUT:
8274                              <1>       ;      [u.r0] will be > 0 if succesful
8275                              <1>
8276                              <1>       ; window
8277                              <1>       ;mov   edi, [v_str] ; LFB start address
8278                              <1>       ;mov   esi, edi
8279                              <1>
8280 0000FDE5 803D[89120300]08    <1>       cmp    byte [v_bpp], 8 ; 8bpp
8281 0000FDEC 7707                <1>       ja     short m_pix_op_xor_w_1
8282                              <1>
8283                              <1>       ; 256 colors (8bpp)
8284 0000FDEE BD[61FD0000]        <1>       mov    ebp, m_pix_op_xor_8
8285 0000FDF3 EB1E                <1>       jmp    short m_pix_op_xor_w_4
8286                              <1>
8287                              <1> m_pix_op_xor_w_1:
8288 0000FDF5 803D[89120300]18    <1>       cmp    byte [v_bpp], 24 ; 24bpp
8289 0000FDFC 7710                <1>       ja     short m_pix_op_xor_w_3 ; 32bpp
8290 0000FDFE 7207                <1>       jb     short m_pix_op_xor_w_2 ; 16bpp
8291                              <1>
8292                              <1>       ; 24 bit true colors
8293 0000FE00 BD[83FD0000]        <1>       mov    ebp, m_pix_op_xor_24
8294 0000FE05 EB0C                <1>       jmp    short m_pix_op_xor_w_4
8295                              <1>
8296                              <1>       ; 65536 colors (16bpp)
8297                              <1> m_pix_op_xor_w_2:
8298 0000FE07 BD[AEFD0000]        <1>       mov    ebp, m_pix_op_xor_16
8299 0000FE0C EB05                <1>       jmp    short m_pix_op_xor_w_4
8300                              <1>
8301                              <1>       ; 32 bit true colors
8302                              <1> m_pix_op_xor_w_3:
8303 0000FE0E BD[CBFD0000]        <1>       mov    ebp, m_pix_op_xor_32
8304                              <1> m_pix_op_xor_w_4:
8305 0000FE13 E917FBFFFF          <1>       jmp    m_pix_op_xor_w_x
8306                              <1>
8307                              <1> m_pix_op_not:
8308                              <1>       ; 06/02/2021
8309                              <1>       ; NOT COLOR (MASKED, full screen)
8310                              <1>       ;
8311                              <1>       ; jump from pix_op_not
8312                              <1>       ;
8313                              <1>       ; INPUT:
8314                              <1>       ;   ecx = [v_siz] ; display page pixel count
8315                              <1>       ;   esi = edi = [v_mem] ; LFB start address
8316                              <1>       ;
8317                              <1>       ;   [maskcolor] = mask color (to be excluded)
8318                              <1>       ;
8319                              <1>       ; OUTPUT:
8320                              <1>       ;      [u.r0] will be > 0 if succesful
8321                              <1>
8322                              <1>       ; Full screen
8323                              <1> m_pix_op_not_0:
8324 0000FE18 803D[89120300]08    <1>       cmp    byte [v_bpp], 8 ; 8bpp
8325 0000FE1F 7715                <1>       ja     short m_pix_op_not_1
8326                              <1>       ; 256 colors (8bpp)
8327                              <1>       ;jmp   short m_pix_op_not_8
8328                              <1> m_pix_op_not_8:
8329                              <1>       ; 8 bit colors (256 colors)
8330 0000FE21 AC                  <1>       lodsb
8331 0000FE22 3A05[9A120300]      <1>       cmp    al, [maskcolor]
8332 0000FE28 7408                <1>       je     short m_pix_op_not_8_1 ; exclude
8333 0000FE2A F617                <1>       not    byte [edi]
8334 0000FE2C FF05[64030300]      <1>       inc    dword [u.r0] ; +1
8335                              <1> m_pix_op_not_8_1:
8336 0000FE32 47                  <1>       inc    edi
8337 0000FE33 E2EC                <1>       loop   m_pix_op_not_8
8338 0000FE35 C3                  <1>       retn
8339                              <1> m_pix_op_not_1:
8340 0000FE36 803D[89120300]18    <1>       cmp    byte [v_bpp], 24 ; 24bpp
8341 0000FE3D 7746                <1>       ja     short m_pix_op_not_3 ; 32bpp
8342 0000FE3F 7229                <1>       jb     short m_pix_op_not_2 ; 16bpp
8343                              <1>       ; 24 bit true colors
8344                              <1>       ;jmp   short m_pix_op_not_24
8345                              <1> m_pix_op_not_24:
8346                              <1>       ; 24 bit true colors
8347 0000FE41 66AD                <1>       lodsw
8348 0000FE43 C1E010              <1>       shl    eax, 16
8349 0000FE46 AC                  <1>       lodsb
8350 0000FE47 C1C010              <1>       rol    eax, 16
8351 0000FE4A 3B05[9A120300]      <1>       cmp    eax, [maskcolor]
```

```
8352 0000FE50 7412              <1>       je    short m_pix_op_not_24_1 ; exclude
8353 0000FE52 F7D0              <1>       not   eax
8354 0000FE54 668907            <1>       mov   [edi], ax
8355 0000FE57 C1E810            <1>       shr   eax, 16
8356 0000FE5A 884702            <1>       mov   [edi+2], al
8357 0000FE5D 8305[64030300]03  <1>       add   dword [u.r0], 3 ; +3
8358                            <1> m_pix_op_not_24_1:
8359 0000FE64 83C703            <1>       add   edi, 3 ; +3
8360 0000FE67 E2D8              <1>       loop  m_pix_op_not_24
8361 0000FE69 C3                <1>       retn
8362                            <1>       ; 65536 colors (16bpp)
8363                            <1> m_pix_op_not_2:
8364                            <1>       ;jmp  short m_pix_op_not_16
8365                            <1> m_pix_op_not_16:
8366                            <1>       ; 16 bit colors (65536 colors)
8367 0000FE6A 66AD              <1>       lodsw
8368 0000FE6C 663B05[9A120300]  <1>       cmp   ax, [maskcolor]
8369 0000FE73 740A              <1>       je    short m_pix_op_not_16_1 ; exclude
8370 0000FE75 66F717            <1>       not   word [edi]
8371 0000FE78 8305[64030300]02  <1>       add   dword [u.r0], 2 ; +2
8372                            <1> m_pix_op_not_16_1:
8373 0000FE7F 83C702            <1>       add   edi, 2 ; +2
8374 0000FE82 E2E6              <1>       loop  m_pix_op_not_16
8375 0000FE84 C3                <1>       retn
8376                            <1> m_pix_op_not_3:
8377                            <1>       ; 32 bit true colors
8378                            <1>       ;jmp  short m_pix_op_not_32
8379                            <1> m_pix_op_not_32:
8380                            <1>       ; 32 bit true colors
8381 0000FE85 AD                <1>       lodsd
8382 0000FE86 3B05[9A120300]    <1>       cmp   eax, [maskcolor]
8383 0000FE8C 7409              <1>       je    short m_pix_op_not_32_1 ; exclude
8384 0000FE8E F717              <1>       not   dword [edi]
8385 0000FE90 8305[64030300]04  <1>       add   dword [u.r0], 4 ; +4
8386                            <1> m_pix_op_not_32_1:
8387 0000FE97 83C704            <1>       add   edi, 4 ; +4
8388 0000FE9A E2E9              <1>       loop  m_pix_op_not_32
8389 0000FE9C C3                <1>       retn
8390                            <1>
8391                            <1> m_pix_op_not_w:
8392                            <1>       ; 06/02/2021
8393                            <1>       ; NOT COLOR (MASKED, window)
8394                            <1>       ;
8395                            <1>       ; jump from pix_op_not_w
8396                            <1>       ;
8397                            <1>       ; INPUT:
8398                            <1>       ;   ecx = bytes per row (to be applied)
8399                            <1>       ;   edx = screen width in bytes
8400                            <1>       ;   ebx = row count
8401                            <1>       ;
8402                            <1>       ;   [maskcolor] = mask color (to be excluded)
8403                            <1>       ;
8404                            <1>       ; OUTPUT:
8405                            <1>       ;     [u.r0] will be > 0 if succesful
8406                            <1>
8407                            <1>       ; window
8408                            <1>       ;mov  edi, [v_str] ; LFB start address
8409                            <1>       ;mov  esi, edi
8410                            <1>
8411 0000FE9D 803D[89120300]08  <1>       cmp   byte [v_bpp], 8 ; 8bpp
8412 0000FEA4 7707              <1>       ja    short m_pix_op_not_w_1
8413                            <1>
8414                            <1>       ; 256 colors (8bpp)
8415 0000FEA6 BD[21FE0000]      <1>       mov   ebp, m_pix_op_not_8
8416 0000FEAB EB1E              <1>       jmp   short m_pix_op_not_w_4
8417                            <1>
8418                            <1> m_pix_op_not_w_1:
8419 0000FEAD 803D[89120300]18  <1>       cmp   byte [v_bpp], 24 ; 24bpp
8420 0000FEB4 7710              <1>       ja    short m_pix_op_not_w_3 ; 32bpp
8421 0000FEB6 7207              <1>       jb    short m_pix_op_not_w_2 ; 16bpp
8422                            <1>
8423                            <1>       ; 24 bit true colors
8424 0000FEB8 BD[41FE0000]      <1>       mov   ebp, m_pix_op_not_24
8425 0000FEBD EB0C              <1>       jmp   short m_pix_op_not_w_4
8426                            <1>
8427                            <1>       ; 65536 colors (16bpp)
8428                            <1> m_pix_op_not_w_2:
8429 0000FEBF BD[6AFE0000]      <1>       mov   ebp, m_pix_op_not_16
8430 0000FEC4 EB05              <1>       jmp   short m_pix_op_not_w_4
8431                            <1>
8432                            <1>       ; 32 bit true colors
8433                            <1> m_pix_op_not_w_3:
8434 0000FEC6 BD[85FE0000]      <1>       mov   ebp, m_pix_op_not_32
8435                            <1> m_pix_op_not_w_4:
8436 0000FECB E95FFAFFFF        <1>       jmp   m_pix_op_not_w_x
8437                            <1>
8438                            <1> m_pix_op_neg:
8439                            <1>       ; 06/02/2021
8440                            <1>       ; NEGATIVE COLOR (MASKED, full screen)
8441                            <1>       ;
8442                            <1>       ; jump from pix_op_neg
8443                            <1>       ;
8444                            <1>       ; INPUT:
8445                            <1>       ;   ecx = [v_siz] ; display page pixel count
8446                            <1>       ;   esi = edi = [v_mem] ; LFB start address
8447                            <1>       ;
8448                            <1>       ;   [maskcolor] = mask color (to be excluded)
8449                            <1>       ;
8450                            <1>       ; OUTPUT:
8451                            <1>       ;     [u.r0] will be > 0 if succesful
8452                            <1>
8453                            <1>       ; Full screen
8454                            <1> m_pix_op_neg_0:
8455 0000FED0 803D[89120300]08  <1>       cmp   byte [v_bpp], 8 ; 8bpp
8456 0000FED7 7715              <1>       ja    short m_pix_op_neg_1
```

```
                                <1>       ; 256 colors (8bpp)
                                <1>       ;jmp   short m_pix_op_neg_8
                                <1> m_pix_op_neg_8:
                                <1>       ; 8 bit colors (256 colors)
0000FED9 AC                     <1>       lodsb
0000FEDA 3A05[9A120300]         <1>       cmp    al, [maskcolor]
0000FEE0 7408                   <1>       je     short m_pix_op_neg_8_1 ; exclude
0000FEE2 F61F                   <1>       neg    byte [edi]
0000FEE4 FF05[64030300]         <1>       inc    dword [u.r0] ; +1
                                <1> m_pix_op_neg_8_1:
0000FEEA 47                     <1>       inc    edi
0000FEEB E2EC                   <1>       loop   m_pix_op_neg_8
0000FEED C3                     <1>       retn
                                <1> m_pix_op_neg_1:
0000FEEE 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000FEF5 7746                   <1>       ja     short m_pix_op_neg_3 ; 32bpp
0000FEF7 7229                   <1>       jb     short m_pix_op_neg_2 ; 16bpp
                                <1>       ; 24 bit true colors
                                <1>       ;jmp   short m_pix_op_neg_24
                                <1> m_pix_op_neg_24:
                                <1>       ; 24 bit true colors
0000FEF9 66AD                   <1>       lodsw
0000FEFB C1E010                 <1>       shl    eax, 16
0000FEFE AC                     <1>       lodsb
0000FEFF C1C010                 <1>       rol    eax, 16
0000FF02 3B05[9A120300]         <1>       cmp    eax, [maskcolor]
0000FF08 7412                   <1>       je     short m_pix_op_neg_24_1 ; exclude
0000FF0A F7D8                   <1>       neg    eax
0000FF0C 668907                 <1>       mov    [edi], ax
0000FF0F C1E810                 <1>       shr    eax, 16
0000FF12 884702                 <1>       mov    [edi+2], al
0000FF15 8305[64030300]03       <1>       add    dword [u.r0], 3 ; +3
                                <1> m_pix_op_neg_24_1:
0000FF1C 83C703                 <1>       add    edi, 3 ; +3
0000FF1F E2D8                   <1>       loop   m_pix_op_neg_24
0000FF21 C3                     <1>       retn
                                <1>       ; 65536 colors (16bpp)
                                <1> m_pix_op_neg_2:
                                <1>       ;jmp   short m_pix_op_neg_16
                                <1> m_pix_op_neg_16:
                                <1>       ; 16 bit colors (65536 colors)
0000FF22 66AD                   <1>       lodsw
0000FF24 663B05[9A120300]       <1>       cmp    ax, [maskcolor]
0000FF2B 740A                   <1>       je     short m_pix_op_neg_16_1 ; exclude
0000FF2D 66F71F                 <1>       neg    word [edi]
0000FF30 8305[64030300]02       <1>       add    dword [u.r0], 2 ; +2
                                <1> m_pix_op_neg_16_1:
0000FF37 83C702                 <1>       add    edi, 2 ; +2
0000FF3A E2E6                   <1>       loop   m_pix_op_neg_16
0000FF3C C3                     <1>       retn
                                <1> m_pix_op_neg_3:
                                <1>       ; 32 bit true colors
                                <1>       ;jmp   short m_pix_op_neg_32
                                <1> m_pix_op_neg_32:
                                <1>       ; 32 bit true colors
0000FF3D AD                     <1>       lodsd
0000FF3E 3B05[9A120300]         <1>       cmp    eax, [maskcolor]
0000FF44 7409                   <1>       je     short m_pix_op_neg_32_1 ; exclude
0000FF46 F71F                   <1>       neg    dword [edi]
0000FF48 8305[64030300]04       <1>       add    dword [u.r0], 4 ; +4
                                <1> m_pix_op_neg_32_1:
0000FF4F 83C704                 <1>       add    edi, 4 ; +4
0000FF52 E2E9                   <1>       loop   m_pix_op_neg_32
0000FF54 C3                     <1>       retn
                                <1>
                                <1> m_pix_op_neg_w:
                                <1>       ; 06/02/2021
                                <1>       ; NEGATIVE COLOR (MASKED, window)
                                <1>       ;
                                <1>       ; jump from pix_op_neg_w
                                <1>       ;
                                <1>       ; INPUT:
                                <1>       ;   ecx = bytes per row (to be applied)
                                <1>       ;   edx = screen width in bytes
                                <1>       ;   ebx = row count
                                <1>       ;
                                <1>       ;   [maskcolor] = mask color (to be excluded)
                                <1>       ;
                                <1>       ; OUTPUT:
                                <1>       ;      [u.r0] will be > 0 if succesful
                                <1>
                                <1>       ; window
                                <1>       ;mov   edi, [v_str] ; LFB start address
                                <1>       ;mov   esi, edi
                                <1>
0000FF55 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000FF5C 7707                   <1>       ja     short m_pix_op_neg_w_1
                                <1>
                                <1>       ; 256 colors (8bpp)
0000FF5E BD[D9FE0000]           <1>       mov    ebp, m_pix_op_neg_8
0000FF63 EB1E                   <1>       jmp    short m_pix_op_neg_w_4
                                <1>
                                <1> m_pix_op_neg_w_1:
0000FF65 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000FF6C 7710                   <1>       ja     short m_pix_op_neg_w_3 ; 32bpp
0000FF6E 7207                   <1>       jb     short m_pix_op_neg_w_2 ; 16bpp
                                <1>
                                <1>       ; 24 bit true colors
0000FF70 BD[F9FE0000]           <1>       mov    ebp, m_pix_op_neg_24
0000FF75 EB0C                   <1>       jmp    short m_pix_op_neg_w_4
                                <1>
                                <1>       ; 65536 colors (16bpp)
                                <1> m_pix_op_neg_w_2:
0000FF77 BD[22FF0000]           <1>       mov    ebp, m_pix_op_neg_16
0000FF7C EB05                   <1>       jmp    short m_pix_op_neg_w_4
```

```
                                <1>
                                <1>       ; 32 bit true colors
                                <1> m_pix_op_neg_w_3:
0000FF7E BD[3DFF0000]           <1>       mov    ebp, m_pix_op_neg_32
                                <1> m_pix_op_neg_w_4:
0000FF83 E9A7F9FFFF             <1>       jmp    m_pix_op_neg_w_x
                                <1>
                                <1> m_pix_op_inc:
                                <1>       ; 06/02/2021
                                <1>       ; INCREASE COLOR (MASKED, full screen)
                                <1>       ;
                                <1>       ; jump from pix_op_inc
                                <1>       ;
                                <1>       ; INPUT:
                                <1>       ;   ecx = [v_siz] ; display page pixel count
                                <1>       ;   esi = edi = [v_mem] ; LFB start address
                                <1>       ;
                                <1>       ;   [maskcolor] = mask color (to be excluded)
                                <1>       ;
                                <1>       ; OUTPUT:
                                <1>       ;      [u.r0] will be > 0 if succesful
                                <1>
                                <1>       ; Full screen
                                <1> m_pix_op_inc_0:
0000FF88 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0000FF8F 7719                   <1>       ja     short m_pix_op_inc_1
                                <1>       ; 256 colors (8bpp)
                                <1>       ;jmp   short m_pix_op_inc_8
                                <1> m_pix_op_inc_8:
                                <1>       ; 8 bit colors (256 colors)
0000FF91 AC                     <1>       lodsb
0000FF92 3A05[9A120300]         <1>       cmp    al, [maskcolor]
0000FF98 740C                   <1>       je     short m_pix_op_inc_8_1 ; exclude
0000FF9A FE07                   <1>       inc    byte [edi]
0000FF9C 7502                   <1>       jnz    short m_pix_op_inc_8_0
0000FF9E FE0F                   <1>       dec    byte [edi]
                                <1> m_pix_op_inc_8_0:
0000FFA0 FF05[64030300]         <1>       inc    dword [u.r0] ; +1
                                <1> m_pix_op_inc_8_1:
0000FFA6 47                     <1>       inc    edi
0000FFA7 E2E8                   <1>       loop   m_pix_op_inc_8
0000FFA9 C3                     <1>       retn
                                <1> m_pix_op_inc_1:
0000FFAA 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0000FFB1 7752                   <1>       ja     short m_pix_op_inc_3 ; 32bpp
0000FFB3 7230                   <1>       jb     short m_pix_op_inc_2 ; 16bpp
                                <1>       ; 24 bit true colors
                                <1>       ;jmp   short m_pix_op_inc_24
                                <1> m_pix_op_inc_24:
                                <1>       ; 24 bit true colors
0000FFB5 66AD                   <1>       lodsw
0000FFB7 C1E010                 <1>       shl    eax, 16
0000FFBA AC                     <1>       lodsb
0000FFBB C1C010                 <1>       rol    eax, 16
0000FFBE 3B05[9A120300]         <1>       cmp    eax, [maskcolor]
0000FFC4 7419                   <1>       je     short m_pix_op_inc_24_1 ; exclude
0000FFC6 40                     <1>       inc    eax
0000FFC7 3DFFFFFF00             <1>       cmp    eax, 0FFFFFFh
0000FFCC 7601                   <1>       jna    short m_pix_op_inc_24_0
0000FFCE 48                     <1>       dec    eax
                                <1> m_pix_op_inc_24_0:
0000FFCF 668907                 <1>       mov    [edi], ax
0000FFD2 C1E810                 <1>       shr    eax, 16
0000FFD5 884702                 <1>       mov    [edi+2], al
0000FFD8 8305[64030300]03       <1>       add    dword [u.r0], 3 ; +3
                                <1> m_pix_op_inc_24_1:
0000FFDF 83C703                 <1>       add    edi, 3 ; +3
0000FFE2 E2D1                   <1>       loop   m_pix_op_inc_24
0000FFE4 C3                     <1>       retn
                                <1>       ; 65536 colors (16bpp)
                                <1> m_pix_op_inc_2:
                                <1>       ;jmp   short m_pix_op_inc_16
                                <1> m_pix_op_inc_16:
                                <1>       ; 16 bit colors (65536 colors)
0000FFE5 66AD                   <1>       lodsw
0000FFE7 663B05[9A120300]       <1>       cmp    ax, [maskcolor]
0000FFEE 740F                   <1>       je     short m_pix_op_inc_16_1 ; exclude
0000FFF0 66FF07                 <1>       inc    word [edi]
0000FFF3 7503                   <1>       jnz    short m_pix_op_inc_16_0
0000FFF5 66FF0F                 <1>       dec    word [edi]
                                <1> m_pix_op_inc_16_0:
0000FFF8 8305[64030300]02       <1>       add    dword [u.r0], 2 ; +2
                                <1> m_pix_op_inc_16_1:
0000FFFF 83C702                 <1>       add    edi, 2 ; +2
00010002 E2E1                   <1>       loop   m_pix_op_inc_16
00010004 C3                     <1>       retn
                                <1> m_pix_op_inc_3:
                                <1>       ; 32 bit true colors
                                <1>       ;jmp   short m_pix_op_inc_32
                                <1> m_pix_op_inc_32:
                                <1>       ; 32 bit true colors
00010005 AD                     <1>       lodsd
00010006 3B05[9A120300]         <1>       cmp    eax, [maskcolor]
0001000C 740D                   <1>       je     short m_pix_op_inc_32_1 ; exclude
0001000E FF07                   <1>       inc    dword [edi]
00010010 7502                   <1>       jnz    short m_pix_op_inc_32_0
00010012 FF0F                   <1>       dec    dword [edi]
                                <1> m_pix_op_inc_32_0:
00010014 8305[64030300]04       <1>       add    dword [u.r0], 4 ; +4
                                <1> m_pix_op_inc_32_1:
0001001B 83C704                 <1>       add    edi, 4 ; +4
0001001E E2E5                   <1>       loop   m_pix_op_inc_32
00010020 C3                     <1>       retn
                                <1>
                                <1> m_pix_op_inc_w:
```

```
                                <1>       ; 06/02/2021
                                <1>       ; INCREASE COLOR (MASKED, window)
                                <1>       ;
                                <1>       ; jump from pix_op_inc_w
                                <1>       ;
                                <1>       ; INPUT:
                                <1>       ;   ecx = bytes per row (to be applied)
                                <1>       ;   edx = screen width in bytes
                                <1>       ;   ebx = row count
                                <1>       ;
                                <1>       ;   [maskcolor] = mask color (to be excluded)
                                <1>       ;
                                <1>       ; OUTPUT:
                                <1>       ;      [u.r0] will be > 0 if succesful
                                <1>
                                <1>       ; window
                                <1>       ;mov   edi, [v_str] ; LFB start address
                                <1>       ;mov   esi, edi
                                <1>
00010021 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
00010028 7707                   <1>       ja     short m_pix_op_inc_w_1
                                <1>
                                <1>       ; 256 colors (8bpp)
0001002A BD[91FF0000]           <1>       mov    ebp, m_pix_op_inc_8
0001002F EB1E                   <1>       jmp    short m_pix_op_inc_w_4
                                <1>
                                <1> m_pix_op_inc_w_1:
00010031 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
00010038 7710                   <1>       ja     short m_pix_op_inc_w_3 ; 32bpp
0001003A 7207                   <1>       jb     short m_pix_op_inc_w_2 ; 16bpp
                                <1>
                                <1>       ; 24 bit true colors
0001003C BD[B5FF0000]           <1>       mov    ebp, m_pix_op_inc_24
00010041 EB0C                   <1>       jmp    short m_pix_op_inc_w_4
                                <1>
                                <1>       ; 65536 colors (16bpp)
                                <1> m_pix_op_inc_w_2:
00010043 BD[E5FF0000]           <1>       mov    ebp, m_pix_op_inc_16
00010048 EB05                   <1>       jmp    short m_pix_op_inc_w_4
                                <1>
                                <1>       ; 32 bit true colors
                                <1> m_pix_op_inc_w_3:
0001004A BD[05000100]           <1>       mov    ebp, m_pix_op_inc_32
                                <1> m_pix_op_inc_w_4:
0001004F E9DBF8FFFF             <1>       jmp    m_pix_op_inc_w_x
                                <1>
                                <1> m_pix_op_dec:
                                <1>       ; 06/02/2021
                                <1>       ; DECREASE COLOR (MASKED, full screen)
                                <1>       ;
                                <1>       ; jump from pix_op_dec
                                <1>       ;
                                <1>       ; INPUT:
                                <1>       ;   ecx = [v_siz] ; display page pixel count
                                <1>       ;   esi = edi = [v_mem] ; LFB start address
                                <1>       ;
                                <1>       ;   [maskcolor] = mask color (to be excluded)
                                <1>       ;
                                <1>       ; OUTPUT:
                                <1>       ;      [u.r0] will be > 0 if succesful
                                <1>
                                <1>       ; Full screen
                                <1> m_pix_op_dec_0:
00010054 803D[89120300]08       <1>       cmp    byte [v_bpp], 8 ; 8bpp
0001005B 7719                   <1>       ja     short m_pix_op_dec_1
                                <1>       ; 256 colors (8bpp)
                                <1>       ;jmp   short m_pix_op_dec_8
                                <1> m_pix_op_dec_8:
                                <1>       ; 8 bit colors (256 colors)
0001005D AC                     <1>       lodsb
0001005E 3A05[9A120300]         <1>       cmp    al, [maskcolor]
00010064 740C                   <1>       je     short m_pix_op_dec_8_1 ; exclude
00010066 FE0F                   <1>       dec    byte [edi]
00010068 7902                   <1>       jns    short m_pix_op_dec_8_0
0001006A FE07                   <1>       inc    byte [edi]
                                <1> m_pix_op_dec_8_0:
0001006C FF05[64030300]         <1>       inc    dword [u.r0] ; +1
                                <1> m_pix_op_dec_8_1:
00010072 47                     <1>       inc    edi
00010073 E2E8                   <1>       loop   m_pix_op_dec_8
00010075 C3                     <1>       retn
                                <1> m_pix_op_dec_1:
00010076 803D[89120300]18       <1>       cmp    byte [v_bpp], 24 ; 24bpp
0001007D 774D                   <1>       ja     short m_pix_op_dec_3 ; 32bpp
0001007F 722B                   <1>       jb     short m_pix_op_dec_2 ; 16bpp
                                <1>       ; 24 bit true colors
                                <1>       ;jmp   short m_pix_op_dec_24
                                <1> m_pix_op_dec_24:
                                <1>       ; 24 bit true colors
00010081 66AD                   <1>       lodsw
00010083 C1E010                 <1>       shl    eax, 16
00010086 AC                     <1>       lodsb
00010087 C1C010                 <1>       rol    eax, 16
0001008A 3B05[9A120300]         <1>       cmp    eax, [maskcolor]
00010090 7414                   <1>       je     short m_pix_op_dec_24_1 ; exclude
00010092 48                     <1>       dec    eax
00010093 7901                   <1>       jns    short m_pix_op_dec_24_0
00010095 40                     <1>       inc    eax
                                <1> m_pix_op_dec_24_0:
00010096 668907                 <1>       mov    [edi], ax
00010099 C1E810                 <1>       shr    eax, 16
0001009C 884702                 <1>       mov    [edi+2], al
0001009F 8305[64030300]03       <1>       add    dword [u.r0], 3 ; +3
                                <1> m_pix_op_dec_24_1:
000100A6 83C703                 <1>       add    edi, 3 ; +3
```

```
8772 000100A9 E2D6               <1>       loop   m_pix_op_dec_24
8773 000100AB C3                 <1>       retn
8774                             <1>       ; 65536 colors (16bpp)
8775                             <1> m_pix_op_dec_2:
8776                             <1>       ;jmp   short m_pix_op_dec_16
8777                             <1> m_pix_op_dec_16:
8778                             <1>       ; 16 bit colors (65536 colors)
8779 000100AC 66AD               <1>       lodsw
8780 000100AE 663B05[9A120300]   <1>       cmp    ax, [maskcolor]
8781 000100B5 740F               <1>       je     short m_pix_op_dec_16_1 ; exclude
8782 000100B7 66FF0F             <1>       dec    word [edi]
8783 000100BA 7903               <1>       jns    short m_pix_op_dec_16_0
8784 000100BC 66FF07             <1>       inc    word [edi]
8785                             <1> m_pix_op_dec_16_0:
8786 000100BF 8305[64030300]02   <1>       add    dword [u.r0], 2 ; +2
8787                             <1> m_pix_op_dec_16_1:
8788 000100C6 83C702             <1>       add    edi, 2 ; +2
8789 000100C9 E2E1               <1>       loop   m_pix_op_dec_16
8790 000100CB C3                 <1>       retn
8791                             <1> m_pix_op_dec_3:
8792                             <1>       ; 32 bit true colors
8793                             <1>       ;jmp   short m_pix_op_dec_32
8794                             <1> m_pix_op_dec_32:
8795                             <1>       ; 32 bit true colors
8796 000100CC AD                 <1>       lodsd
8797 000100CD 3B05[9A120300]     <1>       cmp    eax, [maskcolor]
8798 000100D3 740D               <1>       je     short m_pix_op_dec_32_1 ; exclude
8799 000100D5 FF0F               <1>       dec    dword [edi]
8800 000100D7 7902               <1>       jns    short m_pix_op_dec_32_0
8801 000100D9 FF07               <1>       inc    dword [edi]
8802                             <1> m_pix_op_dec_32_0:
8803 000100DB 8305[64030300]04   <1>       add    dword [u.r0], 4 ; +4
8804                             <1> m_pix_op_dec_32_1:
8805 000100E2 83C704             <1>       add    edi, 4 ; +4
8806 000100E5 E2E5               <1>       loop   m_pix_op_dec_32
8807 000100E7 C3                 <1>       retn
8808                             <1>
8809                             <1> m_pix_op_dec_w:
8810                             <1>       ; 06/02/2021
8811                             <1>       ; DECREASE COLOR (MASKED, window)
8812                             <1>       ;
8813                             <1>       ; jump from pix_op_dec_w
8814                             <1>       ;
8815                             <1>       ; INPUT:
8816                             <1>       ;   ecx = bytes per row (to be applied)
8817                             <1>       ;   edx = screen width in bytes
8818                             <1>       ;   ebx = row count
8819                             <1>       ;
8820                             <1>       ;   [maskcolor] = mask color (to be excluded)
8821                             <1>       ;
8822                             <1>       ; OUTPUT:
8823                             <1>       ;      [u.r0] will be > 0 if succesful
8824                             <1>
8825                             <1>       ; window
8826                             <1>       ;mov   edi, [v_str] ; LFB start address
8827                             <1>       ;mov   esi, edi
8828                             <1>
8829 000100E8 803D[89120300]08   <1>       cmp    byte [v_bpp], 8 ; 8bpp
8830 000100EF 7707               <1>       ja     short m_pix_op_dec_w_1
8831                             <1>
8832                             <1>       ; 256 colors (8bpp)
8833 000100F1 BD[5D000100]       <1>       mov    ebp, m_pix_op_dec_8
8834 000100F6 EB1E               <1>       jmp    short m_pix_op_dec_w_4
8835                             <1>
8836                             <1> m_pix_op_dec_w_1:
8837 000100F8 803D[89120300]18   <1>       cmp    byte [v_bpp], 24 ; 24bpp
8838 000100FF 7710               <1>       ja     short m_pix_op_dec_w_3 ; 32bpp
8839 00010101 7207               <1>       jb     short m_pix_op_dec_w_2 ; 16bpp
8840                             <1>
8841                             <1>       ; 24 bit true colors
8842 00010103 BD[81000100]       <1>       mov    ebp, m_pix_op_dec_24
8843 00010108 EB0C               <1>       jmp    short m_pix_op_dec_w_4
8844                             <1>
8845                             <1>       ; 65536 colors (16bpp)
8846                             <1> m_pix_op_dec_w_2:
8847 0001010A BD[AC000100]       <1>       mov    ebp, m_pix_op_dec_16
8848 0001010F EB05               <1>       jmp    short m_pix_op_dec_w_4
8849                             <1>
8850                             <1>       ; 32 bit true colors
8851                             <1> m_pix_op_dec_w_3:
8852 00010111 BD[CC000100]       <1>       mov    ebp, m_pix_op_dec_32
8853                             <1> m_pix_op_dec_w_4:
8854 00010116 E914F8FFFF         <1>       jmp    m_pix_op_dec_w_x
8855                             <1>
8856                             <1> sysvideo_39:
8857                             <1>       ; 15/02/2021
8858                             <1>       ; 07/02/2021, 08/02/2021
8859                             <1>       ; 03/01/2021, 04/01/2021
8860                             <1>       ; 23/11/2020
8861                             <1>       ; BH = 3
8862                             <1>       ; PIXEL READ/WRITE
8863                             <1>
8864                             <1>       ; 07/02/2021
8865                             <1>       ; 04/01/2021 (TRDOS 386 v2.0.3)
8866 0001011B 80FB03             <1>       cmp    bl, 3
8867 0001011E 761A               <1>       jna    short sysvideo_39_1
8868                             <1>       ; 07/02/2021
8869 00010120 80FB06             <1>       cmp    bl, 6
8870 00010123 7705               <1>       ja     short sysvideo_39_0
8871 00010125 E91A010000         <1>       jmp    sysvideo_39_31
8872                             <1> sysvideo_39_0:
8873                             <1>       ; error
8874 0001012A B3FF               <1>       mov    bl, 0FFh
8875 0001012C 8B2D[60030300]     <1>       mov    ebp, [u.usp]  ; ebp points to user's registers
8876 00010132 895D10             <1>       mov    [ebp+16], ebx ; EBX
```

```
8877 00010135 E9C2D7FFFF          <1>       jmp    sysret
8878                              <1> sysvideo_39_1:
8879 0001013A 803D[CA6F0000]FF    <1>       cmp    byte [CRT_MODE], 0FFh
8880 00010141 7312                <1>       jnb    short sysvideo_39_2 ; SVGA (VESA VBE) video mode
8881                              <1>
8882                              <1>       ; Std VGA or CGA mode
8883 00010143 81C200000A00        <1>       add    edx, 0A0000h
8884 00010149 72DF                <1>       jc     short sysvideo_39_0
8885 0001014B 81FAFFFF0A00        <1>       cmp    edx, 0AFFFFh
8886 00010151 77D7                <1>       ja     short sysvideo_39_0
8887 00010153 EB1E                <1>       jmp    short sysvideo_39_3 ; 8bpp
8888                              <1>
8889                              <1> sysvideo_39_2:
8890                              <1>       ; use current vbe (svga) video mode
8891                              <1>
8892                              <1>       ; get LFB address
8893 00010155 A1[2C120300]        <1>       mov    eax, [LFB_ADDR] ; [LFB_Info+LFBINFO.LFB_addr]
8894 0001015A 09C0                <1>       or     eax, eax
8895 0001015C 74CC                <1>       jz     short sysvideo_39_0
8896 0001015E 3B15[30120300]      <1>       cmp    edx, [LFB_SIZE]
8897 00010164 73C4                <1>       jnb    short sysvideo_39_0
8898                              <1>
8899 00010166 01C2                <1>       add    edx, eax
8900                              <1>       ;jc    short sysvideo_39_0
8901                              <1>
8902                              <1>       ; Pixel read/write in VESA VBE (2/3) video mode
8903                              <1>       ; Video memory at Linear Frame Buffer base address
8904                              <1>
8905 00010168 8A3D[38120300]      <1>       mov    bh, [LFB_Info+LFBINFO.bpp]
8906                              <1>
8907 0001016E 80FF08              <1>       cmp    bh, 8 ; 8bpp
8908 00010171 775D                <1>       ja     short sysvideo_39_17
8909                              <1>
8910                              <1>       ; 8 bits per pixel
8911                              <1> sysvideo_39_3:
8912 00010173 80FB01              <1>       cmp    bl, 1 ; 1 = write pixel
8913 00010176 7406                <1>       je     short sysvideo_39_5
8914 00010178 7712                <1>       ja     short sysvideo_39_8
8915                              <1> sysvideo_39_4:
8916                              <1>       ; read pixel (8bpp)
8917 0001017A 8A02                <1>       mov    al, [edx]
8918                              <1>       ;mov   [u.r0], al
8919                              <1>       ;jmp   sysret
8920 0001017C EB04                <1>       jmp    short sysvideo_39_7
8921                              <1> sysvideo_39_5:
8922                              <1>       ; write pixel (8bpp)
8923 0001017E 88C8                <1>       mov    al, cl
8924                              <1> sysvideo_39_6:
8925 00010180 8802                <1>       mov    [edx], al
8926                              <1> sysvideo_39_7:
8927 00010182 A2[64030300]        <1>       mov    [u.r0], al
8928 00010187 E970D7FFFF          <1>       jmp    sysret
8929                              <1> sysvideo_39_8:
8930 0001018C 80FB03              <1>       cmp    bl, 3 ; mix
8931 0001018F 7208                <1>       jb     short sysvideo_39_9
8932                              <1>       ; mix  pixel colors (8bpp)
8933 00010191 8A02                <1>       mov    al, [edx]
8934 00010193 00C8                <1>       add    al, cl
8935 00010195 D0D8                <1>       rcr    al, 1
8936 00010197 EBE7                <1>       jmp    short sysvideo_39_6
8937                              <1> sysvideo_39_9:
8938                              <1>        ; swap pixel colors (8bpp)
8939 00010199 88C8                <1>       mov    al, cl
8940 0001019B 8602                <1>       xchg   [edx], al
8941 0001019D EBE3                <1>       jmp    short sysvideo_39_7
8942                              <1>
8943                              <1>       ; 16 bits per pixel
8944                              <1> sysvideo_39_10:
8945 0001019F 80FB01              <1>       cmp    bl, 1 ; 1 = write pixel
8946 000101A2 7406                <1>       je     short sysvideo_39_12
8947 000101A4 7714                <1>       ja     short sysvideo_39_15
8948                              <1> sysvideo_39_11:
8949                              <1>       ; read pixel (16bpp)
8950 000101A6 8B02                <1>       mov    eax, [edx]
8951                              <1>       ;mov   [u.r0], ax
8952                              <1>       ;jmp   sysret
8953 000101A8 EB05                <1>       jmp    short sysvideo_39_14
8954                              <1> sysvideo_39_12:
8955                              <1>       ; write pixel (16bpp)
8956 000101AA 89C8                <1>       mov    eax, ecx
8957                              <1> sysvideo_39_13:
8958 000101AC 668902              <1>       mov    [edx], ax
8959                              <1> sysvideo_39_14:
8960 000101AF 66A3[64030300]      <1>       mov    [u.r0], ax
8961 000101B5 E942D7FFFF          <1>       jmp    sysret
8962                              <1> sysvideo_39_15:
8963 000101BA 80FB03              <1>       cmp    bl, 3 ; mix
8964 000101BD 720A                <1>       jb     short sysvideo_39_16
8965                              <1>       ; mix  pixel colors (16bpp)
8966 000101BF 8B02                <1>       mov    eax, [edx]
8967 000101C1 6601C8              <1>       add    ax, cx
8968 000101C4 66D1D8              <1>       rcr    ax, 1
8969 000101C7 EBE3                <1>       jmp    short sysvideo_39_13
8970                              <1> sysvideo_39_16:
8971                              <1>        ; swap pixel colors (16bpp)
8972 000101C9 89C8                <1>       mov    eax, ecx
8973 000101CB 668702              <1>       xchg   [edx], ax
8974 000101CE EBDF                <1>       jmp    short sysvideo_39_14
8975                              <1> sysvideo_39_17:
8976 000101D0 80FF18              <1>       cmp    bh, 24
8977 000101D3 7743                <1>       ja     short sysvideo_39_24
8978 000101D5 72C8                <1>       jb     short sysvideo_39_10
8979                              <1>
8980                              <1>       ; 24 bits per pixel
8981 000101D7 81E1FFFFFF00        <1>       and    ecx, 0FFFFFFh
```

```
8982 000101DD 80FB01              <1>       cmp    bl, 1 ; 1 = write pixel
8983 000101E0 7406                <1>       je     short sysvideo_39_19
8984 000101E2 7712                <1>       ja     short sysvideo_39_22
8985                              <1> sysvideo_39_18:
8986                              <1>       ; read pixel (24bpp)
8987 000101E4 8B02                <1>       mov    eax, [edx]
8988                              <1>       ;and   eax, 0FFFFFFh
8989                              <1>       ;mov   [u.r0], eax
8990                              <1>       ;jmp   sysret
8991 000101E6 EB04                <1>       jmp    short sysvideo_39_21
8992                              <1> sysvideo_39_19:
8993                              <1>       ; write pixel (24bpp)
8994 000101E8 89C8                <1>       mov    eax, ecx
8995                              <1> sysvideo_39_20:
8996                              <1>       ;and   eax, 0FFFFFFh
8997 000101EA 8902                <1>       mov    [edx], eax
8998                              <1> sysvideo_39_21:
8999 000101EC A3[64030300]        <1>       mov    [u.r0], eax
9000 000101F1 E906D7FFFF          <1>       jmp    sysret
9001                              <1> sysvideo_39_22:
9002 000101F6 80FB03              <1>       cmp    bl, 3 ; mix
9003 000101F9 720D                <1>       jb     short sysvideo_39_23
9004                              <1>       ; mix  pixel colors (24bpp)
9005 000101FB 8B02                <1>       mov    eax, [edx]
9006 000101FD 25FFFFFF00          <1>       and    eax, 0FFFFFFh
9007                              <1>       ;and   ecx, 0FFFFFFh
9008 00010202 01C8                <1>       add    eax, ecx
9009 00010204 D1D8                <1>       rcr    eax, 1
9010 00010206 EBE2                <1>       jmp    short sysvideo_39_20
9011                              <1> sysvideo_39_23:
9012                              <1>        ; swap pixel colors (24bpp)
9013 00010208 89C8                <1>       mov    eax, ecx
9014                              <1>       ;and   eax, 0FFFFFFh
9015 0001020A 668702              <1>       xchg   [edx], ax
9016 0001020D C1C810              <1>       ror    eax, 16
9017 00010210 884202              <1>       mov    [edx+2], al
9018 00010213 C1C010              <1>       rol    eax, 16
9019 00010216 EBD4                <1>       jmp    short sysvideo_39_21
9020                              <1>
9021                              <1>       ; 32 bits per pixel
9022                              <1> sysvideo_39_24:
9023 00010218 80FB01              <1>       cmp    bl, 1 ; 1 = write pixel
9024 0001021B 7406                <1>       je     short sysvideo_39_26
9025 0001021D 7712                <1>       ja     short sysvideo_39_29
9026                              <1> sysvideo_39_25:
9027                              <1>       ; read pixel (32bpp)
9028 0001021F 8B02                <1>       mov    eax, [edx]
9029                              <1>       ;mov   [u.r0], eax
9030                              <1>       ;jmp   sysret
9031 00010221 EB04                <1>       jmp    short sysvideo_39_28
9032                              <1> sysvideo_39_26:
9033                              <1>       ; write pixel (32bpp)
9034 00010223 89C8                <1>       mov    eax, ecx
9035                              <1> sysvideo_39_27:
9036 00010225 8902                <1>       mov    [edx], eax
9037                              <1> sysvideo_39_28:
9038 00010227 A3[64030300]        <1>       mov    [u.r0], eax
9039 0001022C E9CBD6FFFF          <1>       jmp    sysret
9040                              <1> sysvideo_39_29:
9041 00010231 80FB03              <1>       cmp    bl, 3 ; mix
9042 00010234 7208                <1>       jb     short sysvideo_39_30
9043                              <1>       ; mix  pixel colors (32bpp)
9044 00010236 8B02                <1>       mov    eax, [edx]
9045 00010238 01C8                <1>       add    eax, ecx
9046 0001023A D1D8                <1>       rcr    eax, 1
9047 0001023C EBE7                <1>       jmp    short sysvideo_39_27
9048                              <1> sysvideo_39_30:
9049                              <1>        ; swap pixel colors (32bpp)
9050 0001023E 89C8                <1>       mov    eax, ecx
9051 00010240 8702                <1>       xchg   [edx], eax
9052 00010242 EBE3                <1>       jmp    short sysvideo_39_28
9053                              <1>
9054                              <1> sysvideo_39_31:
9055                              <1>       ; 08/02/2021
9056                              <1>       ; 07/02/2021
9057                              <1>       ; BL = 4 -> read pixels from user defined positions
9058                              <1>       ; BL = 5 -> write single color pixels to user defined pos.
9059                              <1>       ; BL = 6 -> write multi color pixels to user defined pos.
9060                              <1>       ; ECX = color (CL, CX, ECX)
9061                              <1>       ; EDX = number of pixels
9062                              <1>       ; ESI = user buffer contains dword pixel positions
9063                              <1>       ;       (and dword colors for BL input = 6)
9064                              <1>       ; EDI = user's pixel color buff (destination) for BL = 4
9065                              <1>
9066 00010244 890D[9A120300]      <1>       mov    [maskcolor], ecx
9067 0001024A 89D5                <1>       mov    ebp, edx ; number of pixels
9068 0001024C 803D[CA6F0000]FF    <1>       cmp    byte [CRT_MODE], 0FFh ; SVGA flag
9069 00010253 7317                <1>       jnb    short sysvideo_39_33 ; SVGA (VESA VBE mode)
9070                              <1>       ; Standard VGA mode
9071 00010255 B900000100          <1>       mov    ecx, 65536 ; Video page size (maximum)
9072 0001025A 39CA                <1>       cmp    edx, ecx
9073 0001025C 7709                <1>       ja     short sysvideo_39_32 ; abnormal value !
9074 0001025E B800000A00          <1>       mov    eax, 0A0000h ; Video page start address
9075 00010263 B208                <1>       mov    dl, 8  ; 8 bits per pixel (256 colors)
9076 00010265 EB20                <1>       jmp    short sysvideo_39_34
9077                              <1> sysvideo_39_32:
9078                              <1>       ; nonsense! (edx has abnormal value)
9079 00010267 E990D6FFFF          <1>       jmp    sysret
9080                              <1> sysvideo_39_33:
9081                              <1>       ; get LFB address
9082 0001026C A1[2C120300]        <1>       mov    eax, [LFB_ADDR] ; [LFB_Info+LFBINFO.LFB_addr]
9083 00010271 09C0                <1>       or     eax, eax
9084 00010273 74F2                <1>       jz     short sysvideo_39_32 ; LFB is not ready !
9085 00010275 8B0D[30120300]      <1>       mov    ecx, [LFB_SIZE]
9086 0001027B 39CA                <1>       cmp    edx, ecx
```

```
9087 0001027D 77E8              <1>       ja     short sysvideo_39_32 ; abnormal value !
9088                            <1>
9089                            <1>       ; 08/02/2021
9090 0001027F 89D5              <1>       mov    ebp, edx ; pixel count
9091                            <1>       ;shl   ebp, 2 ; byte count (pixel pos: 4 bytes)
9092                            <1>
9093                            <1>       ; bits per pixel (pixel color size)
9094 00010281 8A15[38120300]    <1>       mov    dl, [LFB_Info+LFBINFO.bpp]
9095                            <1> sysvideo_39_34:
9096 00010287 C1E502            <1>       shl    ebp, 2 ; 15/02/2021 (byte count)
9097 0001028A A3[8A120300]      <1>       mov    [v_mem], eax ; Save video page start address
9098                            <1>       ;mov   [v_siz], ecx ; Video page size (limit)
9099                            <1>       ;mov   ebx, [v_siz]
9100 0001028F 88DE              <1>       mov    dh, bl ; sub function
9101 00010291 89CB              <1>       mov    ebx, ecx ; [v_siz]
9102 00010293 8815[89120300]    <1>       mov    [v_bpp], dl ; bits per pixel (color size)
9103                            <1>
9104 00010299 B900080000        <1>       mov    ecx, 2048
9105 0001029E 39CD              <1>       cmp    ebp, ecx
9106 000102A0 7302              <1>       jnb    short sysvideo_39_35
9107 000102A2 89E9              <1>       mov    ecx, ebp ; fix to requested byte count
9108                            <1> sysvideo_39_35:
9109 000102A4 80FE04            <1>       cmp    dh, 4 ; 08/02/2021
9110                            <1>       ;cmp   bl, 4 ; read pixels from user defined positions
9111 000102A7 7605              <1>       jna    short sysvideo_39_36
9112 000102A9 E9B2000000        <1>       jmp    sysvideo_39_52
9113                            <1>       ; 08/02/2021
9114                            <1>       ;mov   [buffer8], edi  ; user's destination buff addr
9115                            <1> sysvideo_39_36:
9116                            <1>       ; 08/02/2021
9117                            <1>       ; read pixel positions
9118                            <1>       ; as defined in user's source buffer
9119 000102AE 893D[9E120300]    <1>       mov    [buffer8], edi  ; user's destination buff addr
9120 000102B4 BF00760900        <1>       mov    edi, VBE3SAVERESTOREBLOCK ; kernel buffer for
9121                            <1>                                     ; 2028 byte data
9122                            <1>       ; esi = user's source buffer for pixel positions
9123                            <1>       ; ecx = byte count
9124 000102B9 E87E180000        <1>       call   transfer_from_user_buffer
9125 000102BE 72A7              <1>       jc     short sysvideo_39_32 ; error
9126                            <1>       ; ecx  = transfer count (bytes)
9127                            <1>
9128 000102C0 57                <1>       push   edi ; *
9129 000102C1 56                <1>       push   esi ; **
9130 000102C2 51                <1>       push   ecx ; ***
9131                            <1>
9132 000102C3 89FE              <1>       mov    esi, edi ; kernel buffer
9133 000102C5 8B15[8A120300]    <1>       mov    edx, [v_mem] ; video memory
9134 000102CB C1E902            <1>       shr    ecx, 2 ; pixel count (within buffer capacity)
9135                            <1>
9136 000102CE 803D[89120300]08  <1>       cmp    byte [v_bpp], 8 ; 8bpp
9137 000102D5 7753              <1>       ja     short sysvideo_39_49
9138                            <1> sysvideo_39_37:
9139                            <1>       ; 8bpp
9140 000102D7 AD                <1>       lodsd
9141 000102D8 39D8              <1>       cmp    eax, ebx ; < [v_siz]
9142 000102DA 7309              <1>       jnb    short sysvideo_39_39
9143 000102DC 0FB60402          <1>       movzx  eax, byte [edx+eax]
9144                            <1> sysvideo_39_38:
9145 000102E0 AB                <1>       stosd
9146 000102E1 E2F4              <1>       loop   sysvideo_39_37
9147 000102E3 EB49              <1>       jmp    short sysvideo_39_50
9148                            <1> sysvideo_39_39:
9149                            <1>       ; write black color for improper positions
9150 000102E5 31C0              <1>       xor    eax, eax
9151 000102E7 EBF7              <1>       jmp    short sysvideo_39_38
9152                            <1> sysvideo_39_40:
9153 000102E9 803D[89120300]18  <1>       cmp    byte [v_bpp], 24 ; 24bpp
9154 000102F0 772A              <1>       ja     short sysvideo_39_47 ; 32bpp
9155 000102F2 7216              <1>       jb     short sysvideo_39_44 ; 16bpp
9156                            <1> sysvideo_39_41:
9157                            <1>       ; 24bpp
9158 000102F4 AD                <1>       lodsd
9159 000102F5 39D8              <1>       cmp    eax, ebx ; < [v_siz]
9160 000102F7 730D              <1>       jnb    short sysvideo_39_43
9161 000102F9 8B0402            <1>       mov    eax, [edx+eax]
9162 000102FC 25FFFFFF00        <1>       and    eax, 0FFFFFFh
9163                            <1> sysvideo_39_42:
9164 00010301 AB                <1>       stosd
9165 00010302 E2F0              <1>       loop   sysvideo_39_41
9166 00010304 EB28              <1>       jmp    short sysvideo_39_50
9167                            <1> sysvideo_39_43:
9168                            <1>       ; write black color for improper positions
9169 00010306 31C0              <1>       xor    eax, eax
9170 00010308 EBF7              <1>       jmp    short sysvideo_39_42
9171                            <1> sysvideo_39_44:
9172                            <1>       ; 16bpp
9173 0001030A AD                <1>       lodsd
9174 0001030B 39D8              <1>       cmp    eax, ebx ; < [v_siz]
9175 0001030D 7309              <1>       jnb    short sysvideo_39_46
9176 0001030F 0FB70402          <1>       movzx  eax, word [edx+eax]
9177                            <1> sysvideo_39_45:
9178 00010313 AB                <1>       stosd
9179 00010314 E2F4              <1>       loop   sysvideo_39_44
9180 00010316 EB16              <1>       jmp    short sysvideo_39_50
9181                            <1> sysvideo_39_46:
9182                            <1>       ; write black color for improper positions
9183 00010318 31C0              <1>       xor    eax, eax
9184 0001031A EBF7              <1>       jmp    short sysvideo_39_45
9185                            <1> sysvideo_39_47:
9186                            <1>       ; 32bpp
9187 0001031C AD                <1>       lodsd
9188 0001031D 39D8              <1>       cmp    eax, ebx ; < [v_siz]
9189 0001031F 7309              <1>       jnb    short sysvideo_39_49
9190 00010321 0FB70402          <1>       movzx  eax, word [edx+eax]
9191                            <1> sysvideo_39_48:
```

```
9192 00010325 AB                <1>       stosd
9193 00010326 E2F4              <1>       loop   sysvideo_39_47
9194 00010328 EB04              <1>       jmp    short sysvideo_39_50
9195                            <1> sysvideo_39_49:
9196                            <1>       ; write black color for improper positions
9197 0001032A 31C0              <1>       xor    eax, eax
9198 0001032C EBF7              <1>       jmp    short sysvideo_39_48
9199                            <1> sysvideo_39_50:
9200 0001032E 59                <1>       pop    ecx ; transfer count in bytes
9201 0001032F 5E                <1>       pop    esi ; ** ; kernel buffer
9202                            <1>       ;mov   esi, VBE3SAVERESTOREBLOCK ; kernel buffer for
9203                            <1>                                        ; 2028 byte data
9204 00010330 8B3D[9E120300]    <1>       mov    edi, [buffer8]
9205                            <1>       ; edi = user's destination buffer for pixel colors
9206                            <1>       ; ecx = byte count
9207 00010336 E8B7170000        <1>       call   transfer_to_user_buffer
9208 0001033B 5E                <1>       pop    esi ; *
9209 0001033C 7260              <1>       jc     short sysvideo_39_56 ; error
9210                            <1>       ; ecx  = transfer count (bytes)
9211 0001033E 89C8              <1>       mov    eax, ecx
9212 00010340 C1E802            <1>       shr    eax, 2
9213 00010343 0105[64030300]    <1>       add    [u.r0], eax ; transfer count (in pixels)
9214                            <1>
9215 00010349 29CD              <1>       sub    ebp, ecx
9216 0001034B 7651              <1>       jna    short sysvideo_39_56 ; completed/finished
9217 0001034D 01CE              <1>       add    esi, ecx ; next position in source buffer
9218                            <1>       ;add   [buffer8], ecx ; next pos in destination buff
9219 0001034F 01CF              <1>       add    edi, ecx
9220 00010351 66B90008          <1>       mov    cx, 2048 ; new count, limit: kernel buff size
9221 00010355 39CD              <1>       cmp    ebp, ecx ; remain >= limit ?
9222 00010357 7302              <1>       jnb    short sysvideo_39_51 ; yes
9223 00010359 89E9              <1>       mov    ecx, ebp ; fix byte count to remain bytes
9224                            <1> sysvideo_39_51:
9225 0001035B E94EFFFFFF        <1>       jmp    sysvideo_39_36
9226                            <1>
9227                            <1> sysvideo_39_52:
9228 00010360 80FE05            <1>       cmp    dh, 5 ; 08/02/2021
9229                            <1>       ;cmp   bl, 5 ; write pixels to user defined positions
9230 00010363 7605              <1>       jna    short sysvideo_39_53
9231 00010365 E99B000000        <1>       jmp    sysvideo_39_66
9232                            <1> sysvideo_39_53:
9233                            <1>       ; single color pixel writing
9234 0001036A BF00760900        <1>       mov    edi, VBE3SAVERESTOREBLOCK ; kernel buffer for
9235                            <1>                                        ; 2028 byte data
9236                            <1>       ; esi = user's source buffer for pixel positions
9237                            <1>       ; ecx = byte count
9238 0001036F E8C8170000        <1>       call   transfer_from_user_buffer
9239 00010374 7228              <1>       jc     short sysvideo_39_56 ; error
9240                            <1>       ; ecx = transfer count (bytes)
9241                            <1>
9242                            <1>       ; write pixels by using (user) defined positions
9243                            <1>       ; ecx = byte count (1,2,3,4 times pixel count)
9244                            <1>       ; edi = system buffer address
9245                            <1>
9246 00010376 56                <1>       push   esi ; **
9247 00010377 51                <1>       push   ecx ; *
9248                            <1>
9249 00010378 89FE              <1>       mov    esi, edi
9250 0001037A 8B3D[8A120300]    <1>       mov    edi, [v_mem]
9251                            <1>
9252                            <1>       ; 08/02/2021
9253 00010380 C1E902            <1>       shr    ecx, 2 ; pixel count
9254 00010383 8B15[9A120300]    <1>       mov    edx, [maskcolor]
9255                            <1>       ;mov   ebx, [v_siz]
9256                            <1>
9257 00010389 803D[89120300]08  <1>       cmp    byte [v_bpp], 8 ; 8bpp
9258 00010390 7711              <1>       ja     short sysvideo_39_57
9259                            <1> sysvideo_39_54:
9260                            <1>       ; 8bpp
9261 00010392 AD                <1>       lodsd
9262 00010393 39D8              <1>       cmp    eax, ebx ; < [v_siz]
9263 00010395 7303              <1>       jnb    short sysvideo_39_55
9264 00010397 881407            <1>       mov    [edi+eax], dl
9265                            <1> sysvideo_39_55:
9266 0001039A E2F6              <1>       loop   sysvideo_39_54
9267 0001039C EB50              <1>       jmp    short sysvideo_39_64
9268                            <1> sysvideo_39_56:
9269 0001039E E959D5FFFF        <1>       jmp    sysret
9270                            <1> sysvideo_39_57:
9271 000103A3 803D[89120300]18  <1>       cmp    byte [v_bpp], 24 ; 24bpp
9272 000103AA 7732              <1>       ja     short sysvideo_39_62 ; 32bpp
9273 000103AC 721D              <1>       jb     short sysvideo_39_60 ; 16bpp
9274                            <1> sysvideo_39_58:
9275                            <1>       ; 24bpp
9276 000103AE AD                <1>       lodsd
9277 000103AF 39D8              <1>       cmp    eax, ebx ; < [v_siz]
9278 000103B1 7314              <1>       jnb    short sysvideo_39_59
9279 000103B3 881407            <1>       mov    [edi+eax], dl
9280 000103B6 40                <1>       inc    eax
9281 000103B7 C1CA08            <1>       ror    edx, 8
9282 000103BA 66891407          <1>       mov    [edi+eax], dx
9283 000103BE C1C208            <1>       rol    edx, 8
9284 000103C1 FF05[64030300]    <1>       inc    dword [u.r0]
9285                            <1> sysvideo_39_59:
9286 000103C7 E2E5              <1>       loop   sysvideo_39_58
9287 000103C9 EB23              <1>       jmp    short sysvideo_39_64
9288                            <1> sysvideo_39_60:
9289                            <1>       ; 16bpp
9290 000103CB AD                <1>       lodsd
9291 000103CC 39D8              <1>       cmp    eax, ebx ; < [v_siz]
9292 000103CE 730A              <1>       jnb    short sysvideo_39_61
9293 000103D0 66891407          <1>       mov    [edi+eax], dx
9294 000103D4 FF05[64030300]    <1>       inc    dword [u.r0]
9295                            <1> sysvideo_39_61:
9296 000103DA E2EF              <1>       loop   sysvideo_39_60
```

```
9297 000103DC EB10              <1>       jmp    short sysvideo_39_64
9298                            <1> sysvideo_39_62:
9299                            <1>       ; 32bpp
9300 000103DE AD                <1>       lodsd
9301 000103DF 39D8              <1>       cmp    eax, ebx ; < [v_siz]
9302 000103E1 7309              <1>       jnb    short sysvideo_39_63
9303 000103E3 891407            <1>       mov    [edi+eax], edx
9304 000103E6 FF05[64030300]    <1>       inc    dword [u.r0]
9305                            <1> sysvideo_39_63:
9306 000103EC E2F0              <1>       loop   sysvideo_39_62
9307                            <1> sysvideo_39_64:
9308 000103EE 59                <1>       pop    ecx ; **
9309 000103EF 5E                <1>       pop    esi ; *
9310 000103F0 29CD              <1>       sub    ebp, ecx
9311 000103F2 76AA              <1>       jna    short sysvideo_39_56
9312 000103F4 01CE              <1>       add    esi, ecx
9313 000103F6 66B90008          <1>       mov    cx, 2048
9314 000103FA 39CD              <1>       cmp    ebp, ecx
9315 000103FC 7302              <1>       jnb    short sysvideo_39_65
9316 000103FE 89E9              <1>       mov    ecx, ebp
9317                            <1> sysvideo_39_65:
9318 00010400 E965FFFFFF        <1>       jmp    sysvideo_39_53
9319                            <1>
9320                            <1> sysvideo_39_66:
9321                            <1>       ; 15/02/2021
9322 00010405 D1E5              <1>       shl    ebp, 1 ; 8 bytes per pixel (position&color)
9323                            <1> sysvideo_39_67:
9324 00010407 66B90008          <1>       mov    cx, 2048
9325 0001040B 39CD              <1>       cmp    ebp, ecx
9326 0001040D 7302              <1>       jnb    short sysvideo_39_68
9327 0001040F 89E9              <1>       mov    ecx, ebp
9328                            <1> sysvideo_39_68:
9329                            <1>       ; multi colors pixel writing
9330 00010411 BF00760900        <1>       mov    edi, VBE3SAVERESTOREBLOCK ; kernel buffer for
9331                            <1>                                      ; 2048 byte data
9332                            <1>       ; esi = user's source buffer for pixel positions
9333                            <1>       ; ecx = byte count
9334 00010416 E821170000        <1>       call   transfer_from_user_buffer
9335 0001041B 7281              <1>       jc     short sysvideo_39_56 ; error
9336                            <1>       ; ecx  = transfer count
9337                            <1>
9338                            <1>       ; write pixels & colors as defined in user buffer
9339                            <1>       ; ecx = byte count (2,4,6,8 times pixel count)
9340                            <1>       ; edi = system buffer address
9341                            <1>
9342 0001041D 56                <1>       push   esi ; **
9343 0001041E 51                <1>       push   ecx ; *
9344                            <1>
9345 0001041F 89FE              <1>       mov    esi, edi
9346 00010421 8B3D[8A120300]    <1>       mov    edi, [v_mem]
9347                            <1>
9348                            <1>       ; 08/02/2021
9349 00010427 C1E903            <1>       shr    ecx, 3 ; pixel count
9350                            <1>
9351                            <1>       ;mov   ebx, [v_siz]
9352                            <1>
9353 0001042A 803D[89120300]08  <1>       cmp    byte [v_bpp], 8 ; 8bpp
9354 00010431 770F              <1>       ja     short sysvideo_39_71
9355                            <1> sysvideo_39_69:
9356                            <1>       ; 8bpp
9357 00010433 AD                <1>       lodsd  ; position
9358 00010434 89C2              <1>       mov    edx, eax
9359 00010436 AD                <1>       lodsd  ; color
9360 00010437 39DA              <1>       cmp    edx, ebx ; < [v_siz]
9361 00010439 7303              <1>       jnb    short sysvideo_39_70
9362 0001043B 880417            <1>       mov    [edi+edx], al
9363                            <1> sysvideo_39_70:
9364 0001043E E2F3              <1>       loop   sysvideo_39_69
9365 00010440 EB51              <1>       jmp    short sysvideo_39_78
9366                            <1> sysvideo_39_71:
9367 00010442 803D[89120300]18  <1>       cmp    byte [v_bpp], 24 ; 24bpp
9368 00010449 7735              <1>       ja     short sysvideo_39_76 ; 32bpp
9369 0001044B 721D              <1>       jb     short sysvideo_39_74 ; 16bpp
9370                            <1> sysvideo_39_72:
9371                            <1>       ; 24bpp
9372 0001044D AD                <1>       lodsd  ; position
9373 0001044E 89C2              <1>       mov    edx, eax
9374 00010450 AD                <1>       lodsd  ; color
9375 00010451 39DA              <1>       cmp    edx, ebx ; < [v_siz]
9376 00010453 7311              <1>       jnb    short sysvideo_39_73
9377 00010455 880417            <1>       mov    [edi+edx], al
9378 00010458 42                <1>       inc    edx
9379 00010459 C1E808            <1>       shr    eax, 8
9380 0001045C 66890417          <1>       mov    [edi+edx], ax
9381 00010460 FF05[64030300]    <1>       inc    dword [u.r0]
9382                            <1> sysvideo_39_73:
9383 00010466 E2E5              <1>       loop   sysvideo_39_72
9384 00010468 EB29              <1>       jmp    short sysvideo_39_78
9385                            <1> sysvideo_39_74:
9386                            <1>       ; 16bpp
9387 0001046A AD                <1>       lodsd  ; position
9388 0001046B 89C2              <1>       mov    edx, eax
9389 0001046D AD                <1>       lodsd  ; color
9390 0001046E 39DA              <1>       cmp    edx, ebx ; < [v_siz]
9391 00010470 730A              <1>       jnb    short sysvideo_39_75
9392 00010472 66890417          <1>       mov    [edi+edx], ax
9393 00010476 FF05[64030300]    <1>       inc    dword [u.r0]
9394                            <1> sysvideo_39_75:
9395 0001047C E2EC              <1>       loop   sysvideo_39_74
9396 0001047E EB13              <1>       jmp    short sysvideo_39_78
9397                            <1> sysvideo_39_76:
9398                            <1>       ; 32bpp
9399 00010480 AD                <1>       lodsd  ; position
9400 00010481 89C2              <1>       mov    edx, eax
9401 00010483 AD                <1>       lodsd  ; color
```

```
9402 00010484 39DA              <1>       cmp    edx, ebx ; < [v_siz]
9403 00010486 7309              <1>       jnb    short sysvideo_39_77
9404 00010488 890417            <1>       mov    [edi+edx], eax
9405 0001048B FF05[64030300]    <1>       inc    dword [u.r0]
9406                            <1> sysvideo_39_77:
9407 00010491 E2ED              <1>       loop   sysvideo_39_76
9408                            <1> sysvideo_39_78:
9409 00010493 59                <1>       pop    ecx ; *
9410 00010494 5E                <1>       pop    esi ; **
9411                            <1>
9412 00010495 29CD              <1>       sub    ebp, ecx
9413 00010497 762E              <1>       jna    short sysvideo_39_79
9414 00010499 01CE              <1>       add    esi, ecx
9415 0001049B E967FFFFFF        <1>       jmp    sysvideo_39_67
9416                            <1> ;sysvideo_39_79:
9417                            <1> ;     jmp    sysret
9418                            <1>
9419                            <1> sysvideo_16:
9420                            <1>       ; 23/11/2020
9421 000104A0 80FF04            <1>       cmp    bh, 4
9422 000104A3 0F8272FCFFFF      <1>       jb     sysvideo_39 ; bh = 3, pixel r/w
9423 000104A9 7721              <1>       ja     short sysvideo_17
9424                            <1>
9425                            <1>       ; BH = 4
9426                            <1>       ; Direct User Access for CGA video memory.
9427                            <1>       ; Setup user's page tables for direct access to 0B8000h.
9428                            <1>       ;
9429                            <1>       ; Permission checks are not implemented yet !
9430                            <1>       ; (11/07/2016)
9431                            <1>
9432 000104AB B800800B00        <1>       mov    eax, 0B8000h
9433 000104B0 B908000000        <1>       mov    ecx, 8 ; 8 pages (8*4K=32K)
9434 000104B5 89C3              <1>       mov    ebx, eax ; 12/05/2017 ; virtual = physical
9435 000104B7 E83E62FFFF        <1>       call   direct_memory_access
9436 000104BC 0F823AD4FFFF      <1>       jc     sysret
9437                            <1>       ; eax = 0B8000h if there is not an error
9438 000104C2 A3[64030300]      <1>       mov    [u.r0], eax
9439                            <1> sysvideo_39_79: ; 08/01/2021
9440 000104C7 E930D4FFFF        <1>       jmp    sysret
9441                            <1>
9442                            <1> sysvideo_17:
9443                            <1>       ; 23/12/2020
9444                            <1>       ; 11/12/2020
9445                            <1>       ; 10/12/2020
9446                            <1>       ; 23/11/2020
9447 000104CC 80FF06            <1>       cmp    bh, 6
9448 000104CF 740B              <1>       je     short sysvideo_17_0
9449 000104D1 0F82B9000000      <1>       jb     sysvideo_18
9450 000104D7 E913010000        <1>       jmp    sysvideo_20 ; ja
9451                            <1>
9452                            <1> sysvideo_17_0:
9453                            <1>       ; BH = 6
9454                            <1>       ; Direct User Access to Linear Frame Buffer.
9455                            <1>       ; Setup user's page tables for direct access to LFB.
9456                            <1>       ;
9457                            <1>       ; Permission checks are not implemented yet !
9458                            <1>       ; (10/12/2020)
9459                            <1>
9460 000104DC 80FBFF            <1>       cmp    bl, 0FFh ; current video mode
9461 000104DF 722C              <1>       jb     short sysvideo_17_2 ; for desired video mode
9462                            <1>
9463 000104E1 381D[CA6F0000]    <1>       cmp    [CRT_MODE], bl ; VESA VBE video mode ?
9464 000104E7 750E              <1>       jne    short sysvideo_17_1
9465 000104E9 668B0D[1E120300]  <1>       mov    cx, [video_mode]
9466 000104F0 6681E1FF01        <1>       and    cx, 1FFh
9467 000104F5 EB29              <1>       jmp    short sysvideo_17_3
9468                            <1> sysvideo_17_1:
9469                            <1>       ; 11/12/2020
9470 000104F7 88DF              <1>       mov    bh, bl ; 0FFh
9471 000104F9 8A1D[CA6F0000]    <1>       mov    bl, [CRT_MODE] ; VGA/CGA video mode
9472 000104FF 8B2D[60030300]    <1>       mov    ebp, [u.usp]  ; ebp points to user's registers
9473                            <1>       ; 23/12/2020
9474 00010505 895D10            <1>       mov    [ebp+16], ebx ; return to user with EBX value
9475 00010508 E9EFD3FFFF        <1>       jmp    sysret ; return to user with EAX = 0
9476                            <1> sysvideo_17_2:
9477                            <1>       ; bl = VESA video mode - 100h
9478 0001050D B701              <1>       mov    bh, 1 ; bx = 1XXh
9479 0001050F 53                <1>       push   ebx ; requested vesa video mode
9480 00010510 E89636FFFF        <1>       call   vbe_biosfn_return_current_mode
9481 00010515 59                <1>       pop    ecx ; requested vesa video mode
9482 00010516 6681E3FF01        <1>       and    bx, 1FFh
9483 0001051B 6639D9            <1>       cmp    cx, bx
9484 0001051E 7564              <1>       jne    short sysvideo_17_8
9485                            <1> sysvideo_17_3:
9486 00010520 663B0D[2A120300]  <1>       cmp    cx, [LFB_Info+LFBINFO.mode]
9487 00010527 755B              <1>       jne    short sysvideo_17_8
9488                            <1> sysvideo_17_4:
9489                            <1>       ; 11/12/2020
9490 00010529 A1[2C120300]      <1>       mov    eax, [LFB_Info+LFBINFO.LFB_addr]
9491                            <1>       ; 21/12/2020
9492 0001052E 09C0              <1>       or     eax, eax
9493 00010530 744D              <1>       jz     short sysvideo_17_7
9494                            <1>       ;
9495 00010532 8B0D[30120300]    <1>       mov    ecx, [LFB_Info+LFBINFO.LFB_size] ; buff size
9496 00010538 89C3              <1>       mov    ebx, eax ; user's address = physical address
9497                            <1>       ;push  ebx
9498 0001053A 51                <1>       push   ecx
9499                            <1>       ; 21/12/2020
9500 0001053B 81C1FF0F0000      <1>       add    ecx, 4095  ; PAGESIZE - 1
9501                            <1>       ; 14/12/2020
9502 00010541 C1E90C            <1>       shr    ecx, 12  ; convert bytes to pages
9503 00010544 E8B161FFFF        <1>       call   direct_memory_access
9504 00010549 5A                <1>       pop    edx  ; linear frame buffer size in bytes
9505                            <1>       ;pop   eax  ; linear frame buffer address (physical)
9506 0001054A 7233              <1>       jc     short sysvideo_17_7 ; [u.r0] = eax = 0
```

```
9507                                    <1> sysvideo_17_5:
9508 0001054C 668B0D[36120300]          <1>         mov   cx, [LFB_Info+LFBINFO.Y_res] ; screen height
9509 00010553 C1E110                    <1>         shl   ecx, 16
9510 00010556 668B0D[34120300]          <1>         mov   cx,  [LFB_Info+LFBINFO.X_res] ; screen width
9511 0001055D 31DB                      <1>         xor   ebx, ebx
9512 0001055F 8A1D[38120300]            <1>         mov   bl, [LFB_Info+LFBINFO.bpp] ; bits per pixel
9513 00010565 8A3D[2A120300]            <1>         mov   bh, [LFB_Info+LFBINFO.mode] ; XX part of 1XXh
9514                                    <1> sysvideo_26_4: ; 23/12/2020
9515 0001056B 8B2D[60030300]            <1>         mov   ebp, [u.usp]  ; ebp points to user's registers
9516 00010571 895514                    <1>         mov   [ebp+20], edx ; return to user with EDX value
9517 00010574 895D10                    <1>         mov   [ebp+16], ebx ; EBX
9518 00010577 894D18                    <1>         mov   [ebp+24], ecx ; ECX
9519                                    <1> sysvideo_17_6:
9520 0001057A A3[64030300]              <1>         mov   [u.r0], eax ; LFB address
9521                                    <1> sysvideo_17_7:
9522 0001057F E978D3FFFF                <1>         jmp   sysret
9523                                    <1> sysvideo_17_8:
9524                                    <1>         ; cx = mode
9525                                    <1>         ; 21/12/2020
9526 00010584 80CD40                    <1>         or    ch, 40h  ; Linear frame buffer flag
9527 00010587 E84134FFFF                <1>         call  _vbe_biosfn_return_mode_info
9528 0001058C 72F1                      <1>         jc    short sysvideo_17_7
9529 0001058E EB99                      <1>         jmp   short sysvideo_17_4
9530                                    <1>
9531                                    <1> sysvideo_18:
9532                                    <1>         ; BH = 5
9533                                    <1>         ; Direct User Access for VGA video memory.
9534                                    <1>         ; Setup user's page tables for direct access to 0A0000h.
9535                                    <1>         ;
9536                                    <1>         ; Permission checks are not implemented yet !
9537                                    <1>         ; (11/07/2016)
9538                                    <1>
9539 00010590 B800000A00               <1>         mov   eax, 0A0000h
9540 00010595 B910000000                <1>         mov   ecx, 16 ; 16 pages (16*4K=64K)
9541 0001059A 89C3                      <1>         mov   ebx, eax ; 12/05/2017 ; virtual = physical
9542 0001059C E85961FFFF                <1>         call  direct_memory_access
9543 000105A1 0F8255D3FFFF              <1>         jc    sysret
9544                                    <1>         ; eax = 0A0000h if there is not an error
9545 000105A7 A3[64030300]              <1>         mov   [u.r0], eax
9546 000105AC E94BD3FFFF                <1>         jmp   sysret
9547                                    <1>
9548                                    <1> sysvideo_19:
9549                                    <1>         ; 22/01/2021
9550                                    <1>         ; 12/12/2020
9551                                    <1>         ; 11/12/2020
9552                                    <1>         ; 23/11/2020
9553                                    <1>         ; BH = 7
9554                                    <1>         ; Get (Super/Extended VGA) mode
9555                                    <1>         ; and Linear Frame Buffer info.
9556                                    <1>
9557                                    <1>         ; 22/01/2021
9558 000105B1 B3FF                     <1>         mov   bl, 0FFh
9559                                    <1>         ; 11/12/2020
9560                                    <1>         ;cmp  byte [CRT_MODE], 0FFh ; (extended mode?)
9561                                    <1>         ; 22/01/2021
9562 000105B3 381D[CA6F0000]            <1>         cmp   [CRT_MODE], bl  ; 0FFh
9563                                    <1>         ;jb   sysvideo_17_1; not a VESA VBE mode
9564                                    <1>         ; 12/12/2020
9565 000105B9 7305                      <1>         jnb   short sysvideo_19_0
9566 000105BB E937FFFFFF                <1>         jmp   sysvideo_17_1
9567                                    <1>
9568                                    <1> sysvideo_19_0:
9569 000105C0 E8E635FFFF                <1>         call  vbe_biosfn_return_current_mode
9570 000105C5 6681E3FF01                <1>         and   bx, 1FFh
9571 000105CA 663B1D[2A120300]          <1>         cmp   bx, [LFB_Info+LFBINFO.mode]
9572 000105D1 7510                      <1>         jne   short sysvideo_19_2
9573                                    <1> sysvideo_19_1:
9574 000105D3 A1[2C120300]              <1>         mov   eax, [LFB_Info+LFBINFO.LFB_addr]
9575 000105D8 8B15[30120300]            <1>         mov   edx, [LFB_Info+LFBINFO.LFB_size]
9576 000105DE E969FFFFFF                <1>         jmp   sysvideo_17_5
9577                                    <1> sysvideo_19_2:
9578 000105E3 E8E533FFFF                <1>         call  vbe_biosfn_return_mode_info
9579 000105E8 73E9                      <1>         jnc   short sysvideo_19_1
9580 000105EA E90DD3FFFF                <1>         jmp   sysret
9581                                    <1>
9582                                    <1> sysvideo_20:
9583                                    <1>         ; 11/12/2020
9584                                    <1>         ; 23/11/2020
9585 000105EF 80FF08                    <1>         cmp   bh, 8
9586 000105F2 72BD                      <1>         jb    short sysvideo_19 ; video mode & lfb info
9587 000105F4 0F8780000000              <1>         ja    sysvideo_21 ; 12/12/2020
9588                                    <1>
9589                                    <1>         ; BH = 8
9590                                    <1>         ; Set (Super/Extended VGA) mode & return LFB info
9591                                    <1>         ;
9592                                    <1>
9593                                    <1>         ; 11/12/2020
9594 000105FA 80FBFF                   <1>         cmp   bl, 0FFh  ; CGA/VGA mode ?
9595 000105FD 7318                      <1>         jnb   short sysvideo_20_1
9596                                    <1>
9597                                    <1>         ;xor  ah, ah
9598 000105FF 88D8                     <1>         mov   al, bl
9599                                    <1> sysvideo_20_0:
9600 00010601 E86311FFFF                <1>         call  _int10h  ; uses vbe3 pmi32 option
9601 00010606 83F8FF                    <1>         cmp   eax, 0FFFFFFFFh ; -1
9602 00010609 7459                      <1>         je    short sysvideo_20_3 ; error
9603                                    <1>
9604                                    <1>         ; 11/12/2020
9605                                    <1>         ; alternative (it does not use vbe3 pmi32)
9606                                    <1>         ;push  eax
9607                                    <1>         ;call  _set_mode
9608                                    <1>         ;pop   eax
9609                                    <1>         ;jc    short sysvideo_20_3
9610                                    <1>
9611                                    <1>         ;inc   eax
```

```
9612 0001060B FEC0              <1>       inc    al
9613                            <1>       ;mov   [u.r0], ax ; video mode + 1
9614 0001060D A2[64030300]      <1>       mov    [u.r0], al
9615 00010612 E9E5D2FFFF        <1>       jmp    sysret
9616                            <1>
9617                            <1> sysvideo_20_1:
9618                            <1>       ; cx = vesa video mode
9619 00010617 6689C8            <1>       mov    ax, cx
9620 0001061A 663D0001          <1>       cmp    ax, 100h
9621 0001061E 72E1              <1>       jb     short sysvideo_20_0 ; VGA/CGA mode
9622 00010620 663DFF01          <1>       cmp    ax, 1FFh
9623                            <1>       ;ja    short sysvideo_20_4 ; not valid
9624 00010624 773E              <1>       ja     short sysvideo_20_3
9625 00010626 50                <1>       push   eax
9626 00010627 6689C3            <1>       mov    bx, ax
9627 0001062A 66B8024F          <1>       mov    ax, 4F02h
9628                            <1>
9629                            <1>       ; simulate _int10h (int 31h) for func 4F02h
9630                            <1>       ;pushfd
9631                            <1>       ;push  cs
9632                            <1>       ;push  sysvideo_20_1_retn
9633                            <1>       ;push  es ; *
9634                            <1>       ;push  ds ; **      ; SAVE WORK AND PARAMETER REGISTERS
9635                            <1>       ;jmp   VBE_func
9636                            <1> ;sysvideo_20_1_retn:
9637                            <1>
9638 0001062E E83611FFFF        <1>       call   _int10h ; simulate int 10h (int 31h)
9639                            <1>
9640 00010633 6683F84F          <1>       cmp    ax, 004Fh
9641 00010637 58                <1>       pop    eax
9642 00010638 752A              <1>       jne    short sysvideo_20_3 ; error
9643                            <1>       ;pop   eax
9644 0001063A 40                <1>       inc    eax
9645 0001063B A3[64030300]      <1>       mov    [u.r0], eax ; video mode + 1
9646 00010640 09D2              <1>       or     edx, edx ; is LFBINFO requested by user ?
9647                            <1>       ;jz    short sysvideo_20_4
9648 00010642 7420              <1>       jz     short sysvideo_20_3 ; no
9649                            <1>
9650                            <1>       ; 11/12/2020
9651                            <1>       ; Check LFBINFO table/structure
9652                            <1>       ; (it is set by vbe2 'vbe_biosfn_set_mode'
9653                            <1>       ; but if vbe3 vbios pmi is in use,
9654                            <1>       ; it will not set LFBINFO table)
9655                            <1>
9656 00010644 52                <1>       push   edx
9657 00010645 48                <1>       dec    eax  ; video mode
9658 00010646 BE[2A120300]      <1>       mov    esi, LFB_Info
9659 0001064B 663B06            <1>       cmp    ax, [esi+LFBINFO.mode]
9660 0001064E 7407              <1>       je     short sysvideo_20_2
9661                            <1>
9662 00010650 E87833FFFF        <1>       call   _vbe_biosfn_return_mode_info
9663                            <1>       ;jnc   short sysvideo_20_2
9664 00010655 7212              <1>       jc     short sysvideo_20_4 ; edx = 0
9665                            <1>
9666                            <1>       ;; clear LFBINFO table for invalidating
9667                            <1>       ;mov   ecx, LFBINFO.size ; 16
9668                            <1>       ;mov   edi, esi ; LFB_Info table address
9669                            <1>       ;xor   eax, eax
9670                            <1>       ;rep   stosb
9671                            <1>
9672                            <1> sysvideo_20_2:
9673                            <1>       ;pop   ecx
9674                            <1>       ;mov   edi, ecx ; user buffer
9675 00010657 5F                <1>       pop    edi
9676 00010658 B910000000        <1>       mov    ecx, LFBINFO.size ; 16
9677 0001065D E890140000        <1>       call   transfer_to_user_buffer ; fast transfer
9678 00010662 7206              <1>       jc     short sysvideo_20_5
9679                            <1>
9680                            <1>       ;jmp   sysret
9681                            <1> sysvideo_20_3:
9682                            <1>       ;pop   eax ; [u.r0] = 0
9683                            <1> ;sysvideo_20_4:
9684 00010664 E993D2FFFF        <1>       jmp    sysret
9685                            <1>
9686                            <1> sysvideo_20_4:
9687 00010669 5A                <1>       pop    edx
9688                            <1> sysvideo_20_5:
9689 0001066A 31D2              <1>       xor    edx, edx ; 0
9690                            <1>       ; edx = 0 -> invalid LFBINFO data
9691 0001066C 8B2D[60030300]    <1>       mov    ebp, [u.usp]  ; ebp points to user's registers
9692 00010672 895514            <1>       mov    [ebp+20], edx ; return to user with EDX value
9693 00010675 E982D2FFFF        <1>       jmp    sysret
9694                            <1>
9695                            <1> sysvideo_21:
9696                            <1>       ; 04/01/2021
9697                            <1>       ; 03/12/2020
9698 0001067A 80FF0A            <1>       cmp    bh, 10
9699 0001067D 0F82A8010000      <1>       jb     sysvideo_22 ; VESA VBE3 pmi parms
9700                            <1>       ; 23/12/2020
9701 00010683 0F8432020000      <1>       je     sysvideo_26 ; Video memory mapping
9702                            <1>
9703                            <1>       ; 04/01/2020
9704 00010689 80FF0B            <1>       cmp    bh, 11
9705 0001068C 0F87B4020000      <1>       ja     sysvideo_27
9706                            <1>
9707                            <1>       ; BH = 11
9708                            <1>       ; set/read DAC color registers (for 8bpp)
9709                            <1>
9710 00010692 80FB04            <1>       cmp    bl, 4
9711 00010695 0F83AB000000      <1>       jnb    sysvideo_21_7; BMP file type palette
9712                            <1>                           ; handling
9713 0001069B F6C301            <1>       test   bl, 1
9714 0001069E 7555              <1>       jnz    short sysvideo_21_4 ; set/write DAC colors
9715                            <1>
9716                            <1>       ; Read DAC color register or all DAC color registers
```

```
9717 000106A0 F6C302              <1>       test   bl, 2  ; read single DAC color register
9718 000106A3 7424                <1>       jz     short sysvideo_21_2 ; read all DAC color regs
9719                              <1>
9720                              <1>       ; read single DAC color register
9721                              <1>       ; CL = DAC color register (index)
9722                              <1>
9723 000106A5 66BAC703            <1>       mov    dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
9724 000106A9 88C8                <1>       mov    al, cl ; DAC color register
9725 000106AB 31C9                <1>       xor    ecx, ecx ; (this may not be necessary)
9726 000106AD EE                  <1>       out    dx, al
9727                              <1>       ;mov   dx, 3C9h ; VGAREG_DAC_DATA
9728 000106AE B2C9                <1>       mov    dl, 0C9h
9729 000106B0 EC                  <1>       in     al, dx
9730 000106B1 88C4                <1>       mov    ah, al ; red
9731 000106B3 EC                  <1>       in     al, dx
9732 000106B4 88C1                <1>       mov    cl, al ; green
9733 000106B6 EC                  <1>       in     al, dx
9734 000106B7 88C5                <1>       mov    ch, al ; blue
9735 000106B9 C1E108              <1>       shl    ecx, 8
9736 000106BC 88E1                <1>       mov    cl, ah ; red
9737                              <1>       ; CL = Red, CH = Green, byte 3 = Blue, byte 4 = 0
9738                              <1> sysvideo_21_0:
9739 000106BE 890D[64030300]      <1>       mov    [u.r0], ecx
9740                              <1> sysvideo_21_1:
9741 000106C4 E933D2FFFF          <1>       jmp    sysret
9742                              <1> sysvideo_21_2:
9743                              <1>       ; read all DAC color registers
9744 000106C9 89CB                <1>       mov    ebx, ecx ; user's buffer address
9745 000106CB BF00600900          <1>       mov    edi, VBE3STACKADDR
9746 000106D0 89FE                <1>       mov    esi, edi
9747 000106D2 B900030000          <1>       mov    ecx, 768 ; 256*3
9748 000106D7 51                  <1>       push   ecx
9749 000106D8 66BAC703            <1>       mov    dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
9750 000106DC 28C0                <1>       sub    al, al ; 0
9751 000106DE EE                  <1>       out    dx, al
9752                              <1>       ;mov   dx, 3C9h ; VGAREG_DAC_DATA
9753 000106DF B2C9                <1>       mov    dl, 0C9h
9754                              <1> sysvideo_21_3:
9755 000106E1 EC                  <1>       in     al, dx
9756 000106E2 AA                  <1>       stosb
9757 000106E3 EC                  <1>       in     al, dx
9758 000106E4 AA                  <1>       stosb
9759 000106E5 EC                  <1>       in     al, dx
9760 000106E6 AA                  <1>       stosb
9761 000106E7 E2F8                <1>       loop   sysvideo_21_3
9762 000106E9 59                  <1>       pop    ecx
9763                              <1>
9764 000106EA 89DF                <1>       mov    edi, ebx ; user's buffer address
9765                              <1>       ;mov   esi, VBE3STACKADDR
9766                              <1>       ;mov   ecx, 256*3 = 768
9767 000106EC E801140000          <1>       call   transfer_to_user_buffer
9768 000106F1 72D1                <1>       jc     short sysvideo_21_1
9769                              <1>       ;mov   [u.r0], ecx ; actual transfer count
9770 000106F3 EBC9                <1>       jmp    short sysvideo_21_0
9771                              <1>
9772                              <1> sysvideo_21_4:
9773                              <1>       ; Set/Write DAC color register or all registers
9774 000106F5 F6C302              <1>       test   bl, 2 ; write/set single DAC color register
9775 000106F8 741C                <1>       jz     short sysvideo_21_5 ; set all DAC color regs
9776                              <1>
9777                              <1>       ; set single DAC color register
9778                              <1>       ; CL = DAC color register (index)
9779                              <1>       ; (byte 1 = Red, byte 2 = Green, byte 3 = Blue)
9780                              <1>
9781 000106FA 66BAC803            <1>       mov    dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
9782 000106FE 89C8                <1>       mov    eax, ecx ; DAC color register (index)
9783 00010700 C1E910              <1>       shr    ecx, 16  ; cl = green, AH = Red
9784 00010703 EE                  <1>       out    dx, al
9785                              <1>       ;mov   dx, 3C9h ; VGAREG_DAC_DATA
9786 00010704 FEC2                <1>       inc    dl
9787 00010706 88E0                <1>       mov    al, ah ; Red
9788 00010708 EE                  <1>       out    dx, al
9789 00010709 88C8                <1>       mov    al, cl ; Green
9790 0001070B EE                  <1>       out    dx, al
9791 0001070C 88E8                <1>       mov    al, ch ; Blue
9792 0001070E EE                  <1>       out    dx, al
9793 0001070F C1C108              <1>       rol    ecx, 8
9794 00010712 88E1                <1>       mov    cl, ah ; Red
9795                              <1>              ; ecx = 00BBGGRRh
9796 00010714 EBA8                <1>       jmp    short sysvideo_21_0
9797                              <1>
9798                              <1> sysvideo_21_5:
9799                              <1>       ; write/set all DAC color registers
9800 00010716 89CE                <1>       mov    esi, ecx ; user's buffer address
9801 00010718 BF00600900          <1>       mov    edi, VBE3STACKADDR
9802 0001071D 89FB                <1>       mov    ebx, edi
9803 0001071F B900030000          <1>       mov    ecx, 768 ; 256*3
9804 00010724 E813140000          <1>       call   transfer_from_user_buffer
9805 00010729 7299                <1>       jc     short sysvideo_21_1
9806 0001072B 890D[64030300]      <1>       mov    [u.r0], ecx ; actual transfer count
9807                              <1>
9808 00010731 89DE                <1>       mov    esi, ebx ; VBE3STACKADDR
9809 00010733 66BAC803            <1>       mov    dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
9810 00010737 28C0                <1>       sub    al, al ; 0
9811 00010739 EE                  <1>       out    dx, al
9812                              <1>       ;mov   dx, 3C9h ; VGAREG_DAC_DATA
9813 0001073A FEC2                <1>       inc    dl
9814                              <1> sysvideo_21_6:
9815 0001073C AC                  <1>       lodsb
9816 0001073D EE                  <1>       out    dx, al
9817 0001073E AC                  <1>       lodsb
9818 0001073F EE                  <1>       out    dx, al
9819 00010740 AC                  <1>       lodsb
9820 00010741 EE                  <1>       out    dx, al
9821 00010742 E2F8                <1>       loop   sysvideo_21_6
```

```
9822 00010744 EB33              <1>       jmp    short sysvideo_21_9
9823                            <1>
9824                            <1> sysvideo_21_7:
9825                            <1>       ; BMP file type palette handling
9826                            <1>
9827 00010746 F6C301            <1>       test   bl, 1
9828 00010749 7571              <1>       jnz    short sysvideo_21_12 ; set/write DAC colors
9829                            <1>
9830                            <1>       ; Read DAC color register or all DAC color registers
9831 0001074B F6C302            <1>       test   bl, 2  ; read single DAC color register
9832 0001074E 742E              <1>       jz     short sysvideo_21_10 ; read all DAC color regs
9833                            <1>
9834                            <1>       ; read single DAC color register
9835                            <1>       ; CL = DAC color register (index)
9836                            <1>
9837 00010750 66BAC703          <1>       mov    dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
9838 00010754 88C8              <1>       mov    al, cl ; DAC color register
9839 00010756 31C9              <1>       xor    ecx, ecx
9840 00010758 EE                <1>       out    dx, al
9841                            <1>       ;mov   dx, 3C9h ; VGAREG_DAC_DATA
9842 00010759 B2C9              <1>       mov    dl, 0C9h
9843 0001075B EC                <1>       in     al, dx
9844 0001075C C0E002            <1>       shl    al, 2
9845 0001075F 88C5              <1>       mov    ch, al ; red
9846 00010761 EC                <1>       in     al, dx
9847 00010762 C0E002            <1>       shl    al, 2
9848 00010765 88C1              <1>       mov    cl, al ; green
9849 00010767 EC                <1>       in     al, dx
9850 00010768 C1E108            <1>       shl    ecx, 8
9851 0001076B C0E002            <1>       shl    al, 2
9852 0001076E 88C1              <1>       mov    cl, al ; blue
9853 00010770 C1C910            <1>       ror    ecx, 16
9854                            <1>       ; CL = Blue, CH = Green, byte 3 = Red, byte 4 = 0
9855                            <1> sysvideo_21_8:
9856 00010773 890D[64030300]    <1>       mov    [u.r0], ecx
9857                            <1> sysvideo_21_9:
9858 00010779 E97ED1FFFF        <1>       jmp    sysret
9859                            <1> sysvideo_21_10:
9860                            <1>       ; read all DAC color registers
9861 0001077E 89CD              <1>       mov    ebp, ecx ; user's buffer address
9862 00010780 BF00600900        <1>       mov    edi, VBE3STACKADDR
9863 00010785 89FE              <1>       mov    esi, edi
9864 00010787 B900040000        <1>       mov    ecx, 1024 ; 256*4
9865 0001078C 51                <1>       push   ecx
9866 0001078D 66BAC703          <1>       mov    dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
9867 00010791 28C0              <1>       sub    al, al ; 0
9868 00010793 EE                <1>       out    dx, al
9869                            <1>       ;mov   dx, 3C9h ; VGAREG_DAC_DATA
9870 00010794 B2C9              <1>       mov    dl, 0C9h
9871                            <1> sysvideo_21_11:
9872 00010796 31DB              <1>       xor    ebx, ebx
9873 00010798 EC                <1>       in     al, dx ; Red
9874 00010799 C0E002            <1>       shl    al, 2
9875 0001079C 88C7              <1>       mov    bh, al
9876 0001079E EC                <1>       in     al, dx ; Green
9877 0001079F C0E002            <1>       shl    al, 2
9878 000107A2 88C3              <1>       mov    bl, al
9879 000107A4 EC                <1>       in     al, dx ; Blue
9880 000107A5 C0E002            <1>       shl    al, 2
9881 000107A8 C1E308            <1>       shl    ebx, 8
9882 000107AB 89D8              <1>       mov    eax, ebx ; 00RRGGBBh
9883 000107AD AB                <1>       stosd
9884 000107AE E2E6              <1>       loop   sysvideo_21_11
9885 000107B0 59                <1>       pop    ecx
9886                            <1>
9887 000107B1 89EF              <1>       mov    edi, ebp ; user's buffer address
9888                            <1>       ;mov   esi, VBE3STACKADDR
9889                            <1>       ;mov   ecx, 1024 = 4*256
9890 000107B3 E83A130000        <1>       call   transfer_to_user_buffer
9891 000107B8 72BF              <1>       jc     short sysvideo_21_9
9892                            <1>       ;mov   [u.r0], ecx ; actual transfer count
9893 000107BA EBB7              <1>       jmp    short sysvideo_21_8
9894                            <1>
9895                            <1> sysvideo_21_12:
9896                            <1>       ; Set/Write DAC color register or all registers
9897 000107BC F6C302            <1>       test   bl, 2 ; write/set single DAC color register
9898 000107BF 7427              <1>       jz     short sysvideo_21_13 ; set all DAC color regs
9899                            <1>
9900                            <1>       ; set single DAC color register
9901                            <1>       ; CL = DAC color register (index)
9902                            <1>       ; (byte 1 = Blue, byte 2 = Green, byte 3 = Red)
9903                            <1>
9904 000107C1 66BAC803         <1>       mov    dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
9905 000107C5 88C8              <1>       mov    al, cl ; DAC color register (index)
9906 000107C7 88EC              <1>       mov    ah, ch ; Blue
9907 000107C9 C1E910            <1>       shr    ecx, 16
9908 000107CC EE                <1>       out    dx, al
9909                            <1>       ;mov   dx, 3C9h ; VGAREG_DAC_DATA
9910 000107CD FEC2              <1>       inc    dl
9911 000107CF 88E8              <1>       mov    al, ch ; Red
9912 000107D1 C0E802            <1>       shr    al, 2
9913 000107D4 EE                <1>       out    dx, al
9914 000107D5 88C8              <1>       mov    al, cl ; Green
9915 000107D7 C0E802            <1>       shr    al, 2
9916 000107DA EE                <1>       out    dx, al
9917 000107DB 88E0              <1>       mov    al, ah ; Blue
9918 000107DD C0E802            <1>       shr    al, 2
9919 000107E0 EE                <1>       out    dx, al
9920 000107E1 C1C108            <1>       rol    ecx, 8
9921 000107E4 88E1              <1>       mov    cl, ah
9922 000107E6 EB8B              <1>       jmp    short sysvideo_21_8
9923                            <1>
9924                            <1> sysvideo_21_13:
9925                            <1>       ; write/set all DAC color registers
9926 000107E8 89CE              <1>       mov    esi, ecx ; user's buffer address
```

```
9927 000107EA BF00600900          <1>       mov   edi, VBE3STACKADDR
9928 000107EF 89FB                <1>       mov   ebx, edi
9929 000107F1 B900040000          <1>       mov   ecx, 1024 ; 256*4
9930 000107F6 E841130000          <1>       call  transfer_from_user_buffer
9931 000107FB 7229                <1>       jc    short sysvideo_21_15
9932 000107FD 890D[64030300]      <1>       mov   [u.r0], ecx ; actual transfer count
9933                              <1>
9934 00010803 89DE                <1>       mov   esi, ebx ; VBE3STACKADDR
9935 00010805 66BAC803            <1>       mov   dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
9936 00010809 28C0                <1>       sub   al, al ; 0
9937 0001080B EE                  <1>       out   dx, al
9938                              <1>       ;mov  dx, 3C9h ; VGAREG_DAC_DATA
9939 0001080C FEC2                <1>       inc   dl
9940                              <1> sysvideo_21_14:
9941 0001080E AD                  <1>       lodsd
9942                              <1>       ; byte 0 = Blue, byte 1 = Green, byte 2 = Red
9943 0001080F 89C1                <1>       mov   ecx, eax
9944 00010811 C1E010              <1>       shl   eax, 16 ; byte 0 = Red
9945 00010814 C0E802              <1>       shr   al, 2
9946 00010817 EE                  <1>       out   dx, al ; Red
9947 00010818 88E8                <1>       mov   al, ch
9948 0001081A C0E802              <1>       shr   al, 2
9949 0001081D EE                  <1>       out   dx, al ; Green
9950 0001081E 88C8                <1>       mov   al, cl
9951 00010820 C0E802              <1>       shr   al, 2
9952 00010823 EE                  <1>       out   dx, al ; Blue
9953 00010824 E2E8                <1>       loop  sysvideo_21_14
9954                              <1> sysvideo_21_15:
9955 00010826 E9D1D0FFFF          <1>       jmp   sysret
9956                              <1>
9957                              <1> sysvideo_22:
9958                              <1>       ; 22/01/2021
9959                              <1>       ; 17/01/2021
9960                              <1>       ; 04/12/2020
9961                              <1>       ; 03/12/2020
9962                              <1>       ; BH = 9
9963                              <1>       ; Set/Get VESA VBE3 protected mode interface params
9964                              <1>
9965                              <1>       ; 22/01/2021
9966                              <1>       ;cmp  byte [vbe3], 3
9967                              <1>       ;jne  short sysvideo_25 ; not applicable if
9968                              <1>                               ; vbe3 compatible video bios
9969                              <1>                               ; is not detected by kernel
9970 0001082B 80FB02              <1>       cmp   bl, 2
9971                              <1>       ;ja   short sysvideo_25 ; bl > 2 not implemented
9972                              <1>       ; 17/01/2021
9973 0001082E 7716                <1>       ja    short sysvideo_22_0 ; srvs flag sub function
9974                              <1>       ;jb   short sysvideo_23
9975                              <1>
9976                              <1>       ; 21/01/2021
9977 00010830 803D[54090000]03    <1>       cmp   byte [vbe3], 3
9978 00010837 757D                <1>       jne   short sysvideo_25 ; not applicable if
9979                              <1>                               ; vbe3 compatible video bios
9980                              <1>                               ; is not detected by kernel
9981 00010839 80FB01              <1>       cmp   bl, 1
9982 0001083C 7663                <1>       jna   short sysvideo_23
9983                              <1>
9984 0001083E 8A1D[1C120300]      <1>       mov   bl, [pmi32] ; Video bios 32 bit PMI functions
9985 00010844 EB68                <1>       jmp   short sysvideo_24
9986                              <1>
9987                              <1> sysvideo_22_0:
9988                              <1>       ; 17/01/2021
9989                              <1>       ; save/restore video state user permission
9990 00010846 80FB05              <1>       cmp   bl, 5
9991 00010849 771E                <1>       ja    short sysvideo_22_2
9992 0001084B 7208                <1>       jb    short sysvideo_22_1
9993                              <1>       ; get srvs flag value/status
9994 0001084D 8A1D[80120300]      <1>       mov   bl, [srvsf] ; 0 = disabled, 1 = enabled
9995 00010853 EB2C                <1>       jmp   short sysvideo_22_3
9996                              <1>
9997                              <1> sysvideo_22_1:
9998                              <1>       ; permission (root and multi tasking) check
 9999 00010855 E836000000         <1>       call  sysvideo_22_4
10000 0001085A 735A               <1>       jnc   short sysvideo_25 ; not permitted !
10001                             <1>       ; cf = 1
10002 0001085C 80EB03             <1>       sub   bl, 3 ; disable = 0, enable = 1
10003                             <1>       ; 22/01/2021
10004 0001085F 881D[80120300]     <1>       mov   [srvsf], bl
10005 00010865 FEC3               <1>       inc   bl ; 1 = disabled, 2 = enabled
10006 00010867 EB18               <1>       jmp   short sysvideo_22_3
10007                             <1>
10008                             <1> sysvideo_22_2:
10009 00010869 80FB06             <1>       cmp   bl, 6
10010 0001086C 7748               <1>       ja    short sysvideo_25 ; invalid/unimplemented
10011                             <1>       ; get VESA VBE number/status
10012 0001086E 8A25[54090000]     <1>       mov   ah, [vbe3] ; vbe3 = 3, vbe2 = 2, others = 0
10013 00010874 A0[55090000]       <1>       mov   al, [vbe2bios] ; bochs/qemu/vbox emulator status
10014 00010879 66A3[64030300]     <1>       mov   [u.r0], ax
10015 0001087F EB35               <1>       jmp   short sysvideo_25
10016                             <1>
10017                             <1> sysvideo_22_3:
10018                             <1>       ; 22/01/2021
10019 00010881 8A3D[81120300]     <1>       mov   bh, [srvso] ; state options (> 80h -> svga)
10020 00010887 66891D[64030300]   <1>       mov   [u.r0], bx ; function result is return value
10021 0001088E EB26               <1>       jmp   short sysvideo_25
10022                             <1>
10023                             <1> sysvideo_22_4:
10024                             <1>       ; 17/01/2021 - permission will be given by root only
10025 00010890 803D[52960100]00   <1>       cmp   byte [multi_tasking], 0 ; in single user mode
10026 00010897 7707               <1>       ja    short sysvideo_22_5
10027                             <1>       ; 19/01/2021
10028 00010899 803D[B0030300]01   <1>       cmp   byte [u.uid], 1 ; ([u.uid] = 0 -> root)
10029                             <1> sysvideo_22_5:
10030                             <1>       ; [multi_tasking] = 0 & [u.uid] = 0  -> CF = 1
10031                             <1>       ; otherwise -> CF = 0
```

```
10032 000108A0 C3                  <1>       retn
10033                              <1>
10034                              <1> sysvideo_23:
10035                              <1>       ; 17/01/2021
10036                              <1>       ; permission (root and multi tasking) check
10037 000108A1 E8EAFFFFFF          <1>       call   sysvideo_22_4
10038 000108A6 730E                <1>       jnc    short sysvideo_25 ; not permitted !
10039                              <1>
10040 000108A8 881D[1C120300]      <1>       mov    [pmi32], bl ; 1 = enabled, 0 = disabled
10041                              <1> sysvideo_24:
10042 000108AE FEC3                <1>       inc    bl
10043 000108B0 881D[64030300]      <1>       mov    [u.r0], bl ; function result is return value
10044                              <1> sysvideo_25:
10045 000108B6 E941D0FFFF          <1>       jmp    sysret
10046                              <1>
10047                              <1> sysvideo_26:
10048                              <1>       ; 23/12/2020
10049                              <1>       ; BH = 10
10050                              <1>       ; Map video memory to user's buffer
10051                              <1>       ; (multiuser/owner r/w permisions are ignored
10052                              <1>       ;  for current TRDOS 386 version !)
10053                              <1>
10054 000108BB 6681E100F0         <1>       and    cx, ~4095  ; clear low 12 bits
10055 000108C0 09C9                <1>       or     ecx, ecx ; start address of user's buffer
10056 000108C2 74F2                <1>       jz     short sysvideo_25 ; error !
10057                              <1>
10058 000108C4 80FB01              <1>       cmp    bl, 1  ; VGA memory mapping ?
10059 000108C7 740E                <1>       je     short sysvideo_26_1
10060 000108C9 7718                <1>       ja     short sysvideo_26_2
10061                              <1> sysvideo_26_0:
10062                              <1>       ; BL = 0 : CGA memory (0B8000h) map (32K)
10063 000108CB B800800B00          <1>       mov    eax, 0B8000h
10064 000108D0 BB00800000          <1>       mov    ebx, 32768
10065 000108D5 EB37                <1>       jmp    short sysvideo_26_3
10066                              <1> sysvideo_26_1:
10067                              <1>       ; BL = 1 : VGA memory (0A0000h) map (64K)
10068 000108D7 B800000A00          <1>       mov    eax, 0A0000h
10069 000108DC BB00000100          <1>       mov    ebx, 65536
10070 000108E1 EB2B                <1>       jmp    short sysvideo_26_3
10071                              <1> sysvideo_26_2:
10072                              <1>       ; BL = 2 : SVGA memory (LFB) map to user's buffer
10073 000108E3 803D[54090000]02    <1>       cmp    byte [vbe3], 2  ; VESA VBE 2/3 vbios ready ?
10074 000108EA 72CA                <1>       jb     short sysvideo_25  ; no, error !
10075 000108EC 6681E200F0          <1>       and    dx, ~4095 ; clear low 12 bits
10076 000108F1 09D2                <1>       or     edx, edx  ; buffer size in bytes
10077 000108F3 74C1                <1>       jz     short sysvideo_25  ; error
10078 000108F5 89D3                <1>       mov    ebx, edx
10079 000108F7 A1[2C120300]        <1>       mov    eax, [LFB_ADDR] ; [LFB_Info+LFBINFO.LFB_addr]
10080 000108FC 21C0                <1>       and    eax, eax
10081 000108FE 7425                <1>       jz     short sysvideo_26_5
10082                              <1>                           ; (LFB parms are not set yet)
10083 00010900 3B1D[30120300]      <1>       cmp    ebx, [LFB_SIZE] ; [LFB_Info+LFBINFO.LFB_size]
10084 00010906 7606                <1>       jna    short sysvideo_26_3
10085 00010908 8B1D[30120300]      <1>       mov    ebx, [LFB_SIZE]
10086                              <1> sysvideo_26_3:
10087 0001090E 52                  <1>       push   edx
10088 0001090F 53                  <1>       push   ebx    ; buffer size in bytes
10089 00010910 51                  <1>       push   ecx    ; user's buffer address
10090 00010911 87D9                <1>       xchg   ebx, ecx
10091 00010913 C1E90C              <1>       shr    ecx, 12 ; convert buffer size to page count
10092 00010916 E8DF5DFFFF          <1>       call   direct_memory_access
10093 0001091B 59                  <1>       pop    ecx    ; user's buffer address
10094 0001091C 5B                  <1>       pop    ebx    ; buffer size
10095 0001091D 5A                  <1>       pop    edx
10096 0001091E 7296                <1>       jc     short sysvideo_25 ; error !
10097                              <1>                            ; [u.r0] = 0
10098                              <1> ;sysvideo_26_4:
10099                              <1>       ;mov   ebp, [u.usp] ; ebp points to user's registers
10100                              <1>       ;mov   [ebp+20], edx ; return to user with EDX value
10101                              <1>       ;mov   [ebp+16], ebx ; EBX
10102                              <1>       ;mov   [ebp+24], ecx ; ECX
10103                              <1>       ; eax = physical address of video memory (LFB)
10104                              <1>       ;mov   [u.r0], eax
10105                              <1>       ;jmp   sysret
10106 00010920 E946FCFFFF          <1>       jmp    sysvideo_26_4
10107                              <1>
10108                              <1> sysvideo_26_5:
10109 00010925 66A1[E30E0000]      <1>       mov    ax, [def_LFB_addr] ; default LFB for mode 118h
10110                              <1>       ; ah must be 0C0h or 0D0h or E0h
10111                              <1>       ; others are nonsence !?
10112 0001092B 08E4                <1>       or     ah, ah
10113 0001092D 7487                <1>       jz     short sysvideo_25 ; invalid lfb addr or
10114                              <1>                           ; it is not a vbe2 -bochs emu-
10115                              <1>                           ; or vbe3 -real- video bios
10116 0001092F 80FCF0              <1>       cmp    ah, 0F0h
10117 00010932 7382                <1>       jnb    short sysvideo_25 ; nonsence !?
10118 00010934 C1E010              <1>       shl    eax, 16
10119                              <1>       ;jz    short sysvideo_25 ; eax = 0
10120                              <1>
10121 00010937 81FB00907E00        <1>       cmp    ebx, 1920*1080*4 ; maximum value of possible
10122                              <1>                             ; buffer sizes
10123 0001093D 76CF                <1>       jna    short sysvideo_26_3  ; buffer size is proper
10124                              <1>       ; resize buffer to fit 4GB limit
10125 0001093F BB00907E00          <1>       mov    ebx, 1920*1080*4
10126 00010944 EBC8                <1>       jmp    short sysvideo_26_3
10127                              <1>
10128                              <1> sysvideo_27:
10129                              <1>       ; 18/01/2021
10130 00010946 80FF0C              <1>       cmp    bh, 12
10131 00010949 0F877C010000        <1>       ja     sysvideo_28 ; 19/01/2021
10132                              <1>
10133                              <1>       ; BH = 12
10134                              <1>       ; Font sub functions.
10135                              <1>       ; 12/02/2021
10136                              <1>       ; 11/01/2021
```

```
10137                                  <1>       ; 10/01/2021
10138                                  <1>       ;   BL = 0 : Disable system font overwrite
10139                                  <1>       ;   BL = 1 : Enable system font overwrite
10140                                  <1>       ;   BL = 2 : Read system font 8x8
10141                                  <1>       ;   BL = 3 : Read system font 8x14
10142                                  <1>       ;   BL = 4 : Read system font 8x16
10143                                  <1>       ;   BL = 5 : Read user defined font 8x8
10144                                  <1>       ;   BL = 6 : Read user defined font 8x16
10145                                  <1>       ;   BL = 7 : Write system font 8x8
10146                                  <1>       ;   BL = 8 : Write system font 8x14
10147                                  <1>       ;   BL = 9 : Write system font 8x16
10148                                  <1>       ;   BL = 10 : Write user defined font 8x8
10149                                  <1>       ;   BL = 11 : Write user defined font 8x16
10150                                  <1>       ;
10151                                  <1>       ;  BL > 11 : invalid (not implemented)
10152                                  <1>       ;
10153                                  <1>       ; For BL = 1 to 11
10154                                  <1>       ;   ECX = number of characters (<= 256)
10155                                  <1>       ;   EDX = first character (ascii code in DL)
10156                                  <1>       ;   ESI = user's buffer address
10157                                  <1>       ;
10158                                  <1>       ; Return: EAX = character count
10159                                  <1>
10160 0001094F 80FB0B                  <1>       cmp    bl, 11
10161 00010952 7605                    <1>       jna    short sysvideo_27_1
10162                                  <1> sysvideo_27_0:
10163 00010954 E9A3CFFFFF              <1>       jmp    sysret ; not implemented yet !
10164                                  <1> sysvideo_27_1:
10165 00010959 66B80001                <1>       mov    ax, 256
10166 0001095D 08DB                    <1>       or     bl, bl
10167 0001095F 750E                    <1>       jnz    short sysvideo_27_3
10168                                  <1>
10169                                  <1>       ; bl = 0
10170                                  <1>       ; disable system font overwrite
10171                                  <1>
10172 00010961 8025[7E120300]7F        <1>       and    byte [ufont], 7Fh ; clear bit 7
10173                                  <1> sysvideo_27_2:
10174                                  <1>       ;mov   word [u.r0], 256  ; > 0 -> successful
10175 00010968 A3[64030300]            <1>       mov    [u.r0], eax ; 256
10176 0001096D EBE5                    <1>       jmp    short sysvideo_27_0
10177                                  <1> sysvideo_27_3:
10178 0001096F 80FB01                  <1>       cmp    bl, 1
10179 00010972 7710                    <1>       ja     short sysvideo_27_4
10180                                  <1>
10181                                  <1>       ; bl = 1
10182                                  <1>       ; enable system font overwrite
10183                                  <1>       ;      if [multi_tasking]= 0 and [u.uid] = 0
10184                                  <1>
10185                                  <1>       ;cmp   byte [multi_tasking], 0
10186                                  <1>       ;                   ; multi tasking enabled ?
10187                                  <1>       ;ja    short sysvideo_27_0 ; yes
10188                                  <1>       ;; 19/01/2021
10189                                  <1>       ;; system maintenance or single user mode
10190                                  <1>       ;cmp   byte [u.uid], 0  ; root ?
10191                                  <1>       ;ja    short sysvideo_27_0 ; no
10192                                  <1>
10193                                  <1>       ; 19/01/2021
10194                                  <1>       ; multi tasking & root check
10195 00010974 E817FFFFFF              <1>       call   sysvideo_22_4
10196 00010979 73D9                    <1>       jnc    short sysvideo_27_0 ; not permitted
10197                                  <1>
10198                                  <1>       ; [multi_tasking]= 0 and [u.uid] = 0
10199                                  <1>
10200 0001097B 800D[7E120300]80        <1>       or     byte [ufont], 80h ; set bit 7
10201                                  <1>
10202 00010982 EBE4                    <1>       jmp    short sysvideo_27_2
10203                                  <1>
10204                                  <1> sysvideo_27_4:
10205 00010984 09C9                    <1>       or     ecx, ecx
10206 00010986 74CC                    <1>       jz     short sysvideo_27_0
10207 00010988 21D2                    <1>       and    edx, edx
10208 0001098A 7410                    <1>       jz     short sysvideo_27_4_0
10209                                  <1>       ;mov   ax, 256
10210 0001098C 39C1                    <1>       cmp    ecx, eax ; 256
10211 0001098E 77C4                    <1>       ja     short sysvideo_27_0
10212 00010990 48                      <1>       dec    eax
10213 00010991 39C2                    <1>       cmp    edx, eax ; 255
10214 00010993 77BF                    <1>       ja     short sysvideo_27_0
10215 00010995 40                      <1>       inc    eax
10216 00010996 29D0                    <1>       sub    eax, edx ; 256 - DX
10217 00010998 39C8                    <1>       cmp    eax, ecx
10218 0001099A 72B8                    <1>       jb     short sysvideo_27_0
10219                                  <1>
10220                                  <1> sysvideo_27_4_0:
10221 0001099C 89F5                    <1>       mov    ebp, esi
10222                                  <1>
10223 0001099E 80FB06                  <1>       cmp    bl, 6
10224 000109A1 776E                    <1>       ja     short sysvideo_27_13
10225 000109A3 7210                    <1>       jb     short sysvideo_27_5
10226                                  <1>       ; bl = 6
10227 000109A5 F605[7E120300]10        <1>       test   byte [ufont], 16 ; 8x16 user font loaded ?
10228 000109AC 74A6                    <1>       jz     short sysvideo_27_0
10229                                  <1>       ; read 8x16 user defined font
10230 000109AE BE00400900              <1>       mov    esi, VGAFONT16USER
10231 000109B3 EB0C                    <1>       jmp    short sysvideo_27_6
10232                                  <1> sysvideo_27_5:
10233 000109B5 80FB04                  <1>       cmp    bl, 4
10234 000109B8 723F                    <1>       jb     short sysvideo_27_11
10235 000109BA 7725                    <1>       ja     short sysvideo_27_9
10236                                  <1>       ; bl = 4
10237                                  <1>       ; read 8x16 system font
10238 000109BC BE[54760100]            <1>       mov    esi, vgafont16
10239                                  <1> sysvideo_27_6:
10240                                  <1>       ; read 8x16 font
10241 000109C1 66C1E204                <1>       shl    dx, 4 ; * 16
```

```
10242 000109C5 66C1E104            <1>       shl    cx, 4 ; * 16 ; 16 bytes per char
10243                              <1> sysvideo_27_7:
10244 000109C9 89EF                <1>       mov    edi, ebp
10245 000109CB 01D7                <1>       add    edi, edx
10246 000109CD 01D6                <1>       add    esi, edx
10247                              <1>       ; ecx = byte count
10248                              <1>       ; esi = source (in system memory)
10249                              <1>       ; edi = destination (in user memory)
10250 000109CF E81E110000          <1>       call   transfer_to_user_buffer
10251 000109D4 7206                <1>       jc     short sysvideo_27_8
10252 000109D6 890D[64030300]      <1>       mov    [u.r0], ecx
10253                              <1> sysvideo_27_8:
10254 000109DC E91BCFFFFF          <1>       jmp    sysret
10255                              <1> sysvideo_27_9:
10256                              <1>       ; bl = 5
10257 000109E1 F605[7E120300]08    <1>       test   byte [ufont], 8 ; 8x8 user font loaded ?
10258 000109E8 74F2                <1>       jz     short sysvideo_27_8
10259                              <1>       ; read 8x8 user defined font
10260 000109EA BE00500900          <1>       mov    esi, VGAFONT8USER
10261                              <1> sysvideo_27_10:
10262                              <1>       ; read 8x8 font
10263 000109EF 66C1E203            <1>       shl    dx, 3 ; * 8
10264 000109F3 66C1E103            <1>       shl    cx, 3 ; * 8 ; 8 bytes per char
10265 000109F7 EBD0                <1>       jmp    short sysvideo_27_7
10266                              <1>
10267                              <1> sysvideo_27_11:
10268 000109F9 80FB03              <1>       cmp    bl, 3  ; 8x14 system font
10269 000109FC 720C                <1>       jb     short sysvideo_27_12 ; 8x8 system font
10270                              <1>       ; bl = 3
10271                              <1>       ; read 8x14 system font
10272                              <1>       ;mov   al, 14
10273                              <1>       ;mul   dl
10274                              <1>       ;mov   dx, ax
10275                              <1>       ;push  edx
10276                              <1>       ;mov   ax, 14
10277                              <1>       ;mul   cx
10278                              <1>       ;mov   cx, ax
10279                              <1>       ;pop   edx
10280 000109FE E8AB000000          <1>       call   sysvideo_27_14
10281 00010A03 BE[54680100]        <1>       mov    esi, vgafont14
10282 00010A08 EBBF                <1>       jmp    short sysvideo_27_7
10283                              <1>
10284                              <1> sysvideo_27_12:
10285                              <1>       ; bl = 2
10286                              <1>       ; read 8x8 system font
10287 00010A0A BE[54600100]        <1>       mov    esi, vgafont8
10288 00010A0F EBDE                <1>       jmp    short sysvideo_27_10
10289                              <1>
10290                              <1> sysvideo_27_13:
10291                              <1>       ; overwrite font
10292 00010A11 80FB0A              <1>       cmp    bl, 10
10293 00010A14 7774                <1>       ja     short sysvideo_27_22 ; 8x16 user font
10294 00010A16 7224                <1>       jb     short sysvideo_27_15
10295                              <1>       ; bl = 10
10296 00010A18 BF00500900          <1>       mov    edi, VGAFONT8USER
10297 00010A1D F605[7E120300]08    <1>       test   byte [ufont], 8 ; 8x8 user font loaded ?
10298 00010A24 755A                <1>       jnz    short sysvideo_27_21 ; yes
10299 00010A26 08ED                <1>       or     ch, ch ; cx = 256
10300                              <1>       ;jnz   short sysvideo_27_21 ; 256 chars
10301 00010A28 7406                <1>       jz     short sysvideo_27_13_0
10302 00010A2A 66B90008            <1>       mov    cx, 8*256
10303 00010A2E EB39                <1>       jmp    short sysvideo_27_18_0
10304                              <1> sysvideo_27_13_0:
10305                              <1>       ; copy system font to user font before overwrite
10306 00010A30 BE[54600100]        <1>       mov    esi, vgafont8
10307                              <1>       ;push  edi
10308                              <1>       ;push  ecx
10309                              <1>       ;mov   cl, 64
10310                              <1>       ;rep   movsd
10311                              <1>       ;pop   ecx
10312                              <1>       ;pop   edi
10313                              <1>       ;mov   esi, ebp ; user's font buffer
10314 00010A35 E886000000          <1>       call   sysvideo_27_23
10315 00010A3A EB44                <1>       jmp    short sysvideo_27_21
10316                              <1>
10317                              <1> sysvideo_27_15:
10318                              <1>       ; check system font overwrite permission
10319 00010A3C F605[7E120300]80    <1>       test   byte [ufont], 80h
10320 00010A43 7497                <1>       jz     short sysvideo_27_8
10321                              <1>
10322 00010A45 80FB08              <1>       cmp    bl, 8
10323 00010A48 7740                <1>       ja     short sysvideo_27_22 ; 8x16 system font
10324 00010A4A 722F                <1>       jb     short sysvideo_27_20 ; 8x8 system font
10325                              <1>       ; bl = 8
10326                              <1>       ; overwrite 8x14 system font
10327                              <1>       ;mov   al, 14
10328                              <1>       ;mul   dl
10329                              <1>       ;mov   dx, ax
10330                              <1>       ;push  edx
10331                              <1>       ;mov   ax, 14
10332                              <1>       ;mul   cx
10333                              <1>       ;mov   cx, ax
10334                              <1>       ;pop   edx
10335 00010A4C E85D000000          <1>       call   sysvideo_27_14
10336 00010A51 BF[54680100]        <1>       mov    edi, vgafont14
10337 00010A56 EB0D                <1>       jmp    short sysvideo_27_18
10338                              <1> sysvideo_27_16:
10339                              <1>       ; bl = 9
10340                              <1>       ; overwrite 8x16 system font
10341 00010A58 BF[54760100]        <1>       mov    edi, vgafont16
10342                              <1> sysvideo_27_17:
10343                              <1>       ; overwrite 8x16 font
10344 00010A5D 66C1E204            <1>       shl    dx, 4 ; * 16
10345 00010A61 66C1E104            <1>       shl    cx, 4 ; * 16 ; 16 bytes per char
10346                              <1> sysvideo_27_18:
```

```
10347 00010A65 01D7              <1>       add    edi, edx
10348 00010A67 01D6              <1>       add    esi, edx
10349                            <1> sysvideo_27_18_0:
10350                            <1>       ; ecx = byte count
10351                            <1>       ; esi = source (in user memory)
10352                            <1>       ; edi = destination (in system memory)
10353 00010A69 E8CE100000        <1>       call   transfer_from_user_buffer
10354 00010A6E 7206              <1>       jc     short sysvideo_27_19
10355 00010A70 890D[64030300]    <1>       mov    [u.r0], ecx
10356                            <1> sysvideo_27_19:
10357 00010A76 E981CEFFFF        <1>       jmp    sysret
10358                            <1> sysvideo_27_20:
10359                            <1>       ; bl = 7
10360                            <1>       ; overwrite 8x8 system font
10361 00010A7B BF[54600100]      <1>       mov    edi, vgafont8
10362                            <1> sysvideo_27_21:
10363                            <1>       ; write 8x8 font
10364 00010A80 66C1E203          <1>       shl    dx, 3 ; * 8
10365 00010A84 66C1E103          <1>       shl    cx, 3 ; * 8 ; 8 bytes per char
10366 00010A88 EBDB              <1>       jmp    short sysvideo_27_18
10367                            <1> sysvideo_27_22:
10368                            <1>       ; bl = 11
10369                            <1>       ; overwrite 8x16 user defined font
10370 00010A8A BF00400900        <1>       mov    edi, VGAFONT16USER
10371 00010A8F F605[7E120300]10  <1>       test   byte [ufont], 16 ; 8x16 user font loaded ?
10372 00010A96 75C5              <1>       jnz    short sysvideo_27_17 ; yes
10373 00010A98 08ED              <1>       or     ch, ch ; cx = 256
10374                            <1>       ;jnz   short sysvideo_27_17 ; 256 chars
10375 00010A9A 7406              <1>       jz     short sysvideo_27_22_0
10376 00010A9C 66B90010          <1>       mov    cx, 16*256
10377 00010AA0 EBC7              <1>       jmp    short sysvideo_27_18_0
10378                            <1> sysvideo_27_22_0:
10379                            <1>       ; copy system font to user font before overwrite
10380 00010AA2 BE[54760100]      <1>       mov    esi, vgafont16
10381                            <1>       ;push  edi
10382                            <1>       ;push  ecx
10383                            <1>       ;mov   cl, 64
10384                            <1>       ;rep   movsd
10385                            <1>       ;pop   ecx
10386                            <1>       ;pop   edi
10387                            <1>       ;mov   esi, ebp ; user's font buffer
10388 00010AA7 E814000000        <1>       call   sysvideo_27_23
10389 00010AAC EBAF              <1>       jmp    short sysvideo_27_17
10390                            <1>
10391                            <1> sysvideo_27_14:
10392 00010AAE B00E              <1>       mov    al, 14
10393 00010AB0 F6E2              <1>       mul    dl
10394 00010AB2 6689C2            <1>       mov    dx, ax
10395 00010AB5 52                <1>       push   edx
10396                            <1>       ; 12/02/2021
10397 00010AB6 66B80E00          <1>       mov    ax, 14
10398                            <1>       ;mov   eax, 14
10399                            <1>       ;mul   cx
10400 00010ABA F7E1              <1>       mul    ecx
10401                            <1>       ;mov   cx, ax
10402 00010ABC 89C1              <1>       mov    ecx, eax
10403 00010ABE 5A                <1>       pop    edx
10404 00010ABF C3                <1>       retn
10405                            <1>
10406                            <1> sysvideo_27_23:
10407 00010AC0 57                <1>       push   edi
10408 00010AC1 51                <1>       push   ecx
10409 00010AC2 B140              <1>       mov    cl, 64
10410 00010AC4 F3A5              <1>       rep    movsd
10411 00010AC6 59                <1>       pop    ecx
10412 00010AC7 5F                <1>       pop    edi
10413 00010AC8 89EE              <1>       mov    esi, ebp ; user's font buffer
10414 00010ACA C3                <1>       retn
10415                            <1>
10416                            <1> sysvideo_28:
10417                            <1>       ; 24/01/2021
10418                            <1>       ; 23/01/2021
10419                            <1>       ; 18/01/2021
10420 00010ACB 80FF0E            <1>       cmp    bh, 14
10421 00010ACE 0F8275010000      <1>       jb     sysvideo_29
10422 00010AD4 0F8754020000      <1>       ja     sysvideo_30
10423                            <1>
10424                            <1>       ; BH = 14
10425                            <1>       ; Save/Restore Super VGA video state
10426                            <1>
10427                            <1>       ; BL = options
10428                            <1>       ;      bit 0 - Save (0) or Restore (1)
10429                            <1>       ;      bit 1 - controller hardware state
10430                            <1>       ;      bit 2 - BIOS data state
10431                            <1>       ;      bit 3 - DAC state
10432                            <1>       ;      bit 4 - (extended) Register state
10433                            <1>       ;      bit 5 - system (0) or user (1) memory
10434                            <1>       ;      bit 6 - verify without transfer
10435                            <1>       ;      bit 7 - not used (must be 0)
10436                            <1>
10437                            <1>       ; ECX = Buffer address or VideoStateID
10438                            <1>
10439 00010ADA 803D[54090000]02  <1>       cmp    byte [vbe3], 2 ;  VESA VBE2 or VBE3 ?
10440 00010AE1 7717              <1>       ja     short sysvideo_28_0 ; yes
10441 00010AE3 7210              <1>       jb     short sysvideo_28_16 ; not a SVGA sys !
10442                            <1>
10443                            <1>       ; == VBE2 ==
10444                            <1>       ; Check Bochs/Qemu/VirtualBox PC emulator
10445                            <1>       ; (vbe2 is usable only for emulator's vbios)
10446 00010AE5 8A25[55090000]    <1>       mov    ah, [vbe2bios]
10447 00010AEB 80FCC0            <1>       cmp    ah, 0C0h
10448 00010AEE 7205              <1>       jb     short sysvideo_28_16 ; unknown vbios !
10449 00010AF0 80FCC5            <1>       cmp    ah, 0C5h
10450 00010AF3 7605              <1>       jna    short sysvideo_28_0
10451                            <1>              ; Use kernel's vbios functions (video.s)
```

```
10452                                  <1> sysvideo_28_16:
10453                                  <1>       ; unknown vbios !
10454 00010AF5 E902CEFFFF               <1>       jmp    sysret
10455                                  <1>
10456                                  <1> sysvideo_28_0:
10457 00010AFA 80FB7F                  <1>       cmp    bl, 7Fh
10458 00010AFD 77F6                    <1>       ja     short sysvideo_28_16 ; unknown options
10459                                  <1>
10460 00010AFF 88DA                    <1>       mov    dl, bl
10461 00010B01 80E21F                  <1>       and    dl, 1Fh
10462 00010B04 D0EA                    <1>       shr    dl, 1
10463 00010B06 74ED                    <1>       jz     short sysvideo_28_16 ; invalid !
10464                                  <1>       ; DL = VBE Function 4F04h Save/Restore options
10465                                  <1>       ;  bit 0 : controller hardware state
10466                                  <1>       ;  bit 1 : BIOS data state
10467                                  <1>       ;  bit 2 : DAC state
10468                                  <1>       ;  bit 3 : (extended) Register state
10469                                  <1>
10470 00010B08 F6C320                  <1>       test   bl, 32 ; bit 5
10471 00010B0B 0F85B1000000            <1>       jnz    sysvideo_28_7 ; user buffer
10472                                  <1>
10473                                  <1>       ; source or destination is kernel/system buffer
10474                                  <1>
10475 00010B11 803D[80120300]00        <1>       cmp    byte [srvsf], 0 ; srs permission flag
10476 00010B18 76DB                    <1>       jna    short sysvideo_28_16 ; not permitted
10477                                  <1>
10478 00010B1A F6C301                  <1>       test   bl, 1
10479 00010B1D 743A                    <1>       jz     short sysvideo_28_4 ; Save
10480                                  <1>
10481                                  <1>       ; Restore
10482 00010B1F 3B0D[82120300]          <1>       cmp    ecx, [VideoStateID]
10483 00010B25 75CE                    <1>       jne    short sysvideo_28_16 ; not correct ID !
10484                                  <1>
10485 00010B27 0FB6CA                  <1>       movzx  ecx, dl
10486 00010B2A 80CA80                  <1>       or     dl, 80h
10487 00010B2D 3A15[81120300]          <1>       cmp    dl, [srvso]
10488 00010B33 75C0                    <1>       jne    short sysvideo_28_16 ; not correct !
10489                                  <1>
10490 00010B35 88DA                    <1>       mov    dl, bl
10491                                  <1>
10492                                  <1>       ; ecx = cl = options
10493 00010B37 E83331FFFF              <1>       call   vbe_srs_gbs
10494                                  <1>       ; ebx = state buffer size (data size)
10495                                  <1>
10496 00010B3C 891D[64030300]          <1>       mov    [u.r0], ebx
10497                                  <1>
10498 00010B42 F6C240                  <1>       test   dl, 64 ; verify without transfer
10499 00010B45 75AE                    <1>       jnz    short sysvideo_28_16 ; yes
10500                                  <1>
10501 00010B47 BE00580900              <1>       mov    esi, VBE3VIDEOSTATE
10502 00010B4C BF00760900              <1>       mov    edi, VBE3SAVERESTOREBLOCK
10503 00010B51 87CB                    <1>       xchg   ecx, ebx
10504 00010B53 F3A4                    <1>       rep    movsb
10505                                  <1>
10506 00010B55 88D9                    <1>       mov    cl, bl
10507                                  <1>
10508                                  <1>       ; 23/01/2021
10509 00010B57 EB44                    <1>       jmp    short sysvideo_28_10
10510                                  <1>
10511                                  <1> sysvideo_28_4:
10512 00010B59 53                      <1>       push   ebx
10513                                  <1>       ; 24/01/2021
10514 00010B5A 31DB                    <1>       xor    ebx, ebx ; 0 ; use kernel's buffer
10515 00010B5C 881D[81120300]          <1>       mov    [srvso], bl ; 0 ; invalidate
10516 00010B62 891D[82120300]          <1>       mov    [VideoStateID], ebx ; 0 ; invalidate
10517 00010B68 0FB6CA                  <1>       movzx  ecx, dl ; options
10518 00010B6B B201                    <1>       mov    dl, 1 ; save state
10519 00010B6D E890000000              <1>       call   sysvideo_28_11 ; 23/01/2021
10520                                  <1>       ; Note: VBE3 BIOS data save option will be
10521                                  <1>       ;      disabled.. ; 24/01/2021
10522 00010B72 89CA                    <1>       mov    edx, ecx ; state (save) options
10523 00010B74 5B                      <1>       pop    ebx
10524                                  <1>
10525 00010B75 6683F84F                <1>       cmp    ax, 4Fh ; successful ?
10526 00010B79 7536                    <1>       jne    short sysvideo_28_3 ; no !
10527                                  <1>
10528 00010B7B F6C340                  <1>       test   bl, 64 ; verify without transfer
10529 00010B7E 7536                    <1>       jnz    short sysvideo_28_6 ; yes
10530                                  <1>
10531                                  <1>       ; ecx = cl = options
10532 00010B80 E8EA30FFFF              <1>       call   vbe_srs_gbs
10533                                  <1>       ; ebx = state buffer size (data size)
10534                                  <1>
10535 00010B85 BE00760900              <1>       mov    esi, VBE3SAVERESTOREBLOCK
10536 00010B8A BF00580900              <1>       mov    edi, VBE3VIDEOSTATE
10537 00010B8F 89D9                    <1>       mov    ecx, ebx
10538 00010B91 F3A4                    <1>       rep    movsb
10539                                  <1>
10540 00010B93 88D1                    <1>       mov    cl, dl
10541 00010B95 80C980                  <1>       or     cl, 80h ; SVGA (VESA VBE) flag
10542                                  <1>       ;mov   [srvso], dl
10543                                  <1>
10544 00010B98 E908010000              <1>       jmp    sysvideo_28_15
10545                                  <1>
10546                                  <1>       ; 23/01/2021
10547                                  <1> sysvideo_28_10:
10548                                  <1>       ; CL = VESA VBE3 Save/Restore options
10549                                  <1>
10550 00010B9D B202                    <1>       mov    dl, 2 ; restore state
10551                                  <1>
10552 00010B9F E85C000000              <1>       call   sysvideo_28_1
10553                                  <1>
10554 00010BA4 6683F84F                <1>       cmp    ax, 4Fh ; successful ?
10555 00010BA8 7407                    <1>       je     short sysvideo_28_3
10556                                  <1>       ;jmp   short sysvideo_28_9
```

```
10557                                  <1>
10558                                  <1> sysvideo_28_9:
10559                                  <1>       ; return zero size (error) to user
10560 00010BAA 29C0                     <1>       sub    eax, eax
10561                                  <1> sysvideo_28_5:
10562 00010BAC A3[64030300]              <1>       mov    [u.r0], eax
10563                                  <1> sysvideo_28_3:
10564 00010BB1 E946CDFFFF                <1>       jmp    sysret
10565                                  <1>
10566                                  <1> sysvideo_28_6:
10567                                  <1>       ; use timer ticks as VideoStateID
10568 00010BB6 A1[408A0100]              <1>       mov    eax, [TIMER_LH]
10569 00010BBB 09C0                      <1>       or     eax, eax
10570 00010BBD 75ED                      <1>       jnz    short sysvideo_28_5
10571 00010BBF 40                        <1>       inc    eax
10572 00010BC0 EBEA                      <1>       jmp    short sysvideo_28_5
10573                                  <1>
10574                                  <1> sysvideo_28_7:
10575                                  <1>       ; save/restore to/from user buffer
10576                                  <1>
10577                                  <1>       ; 23/01/2021
10578 00010BC2 89CE                      <1>       mov    esi, ecx ; user's vstate buffer
10579 00010BC4 BF00760900                <1>       mov    edi, VBE3SAVERESTOREBLOCK
10580                                  <1>
10581 00010BC9 0FB6CA                    <1>       movzx  ecx, dl ; VESA VBE func 4F04h options
10582                                  <1>
10583                                  <1>       ; source or destination is user buffer
10584 00010BCC F6C301                    <1>       test   bl, 1
10585 00010BCF 7444                      <1>       jz     short sysvideo_28_12  ; Save
10586                                  <1>
10587                                  <1>       ; Restore
10588 00010BD1 803D[80120300]00          <1>       cmp    byte [srvsf], 0 ; srs permission flag
10589 00010BD8 766A                      <1>       jna    short sysvideo_28_14 ; not permitted
10590                                  <1>
10591 00010BDA 88DA                      <1>       mov    dl, bl ; 'sysvideo' options
10592                                  <1>
10593                                  <1>       ; ecx = cl = options
10594 00010BDC E88E30FFFF                <1>       call   vbe_srs_gbs
10595                                  <1>       ; ebx = state buffer size (data size)
10596                                  <1>
10597 00010BE1 891D[64030300]            <1>       mov    [u.r0], ebx ; transfer count
10598                                  <1>
10599 00010BE7 F6C240                    <1>       test   dl, 64 ; verify without transfer
10600 00010BEA 7558                      <1>       jnz    short sysvideo_28_14 ; yes
10601                                  <1>
10602 00010BEC 6681FB0008                <1>       cmp    bx, 2048
10603 00010BF1 73B7                      <1>       jnb    short sysvideo_28_9 ; invalid
10604                                  <1>
10605 00010BF3 87CB                      <1>       xchg   ecx, ebx
10606                                  <1>       ; esi = user buffer
10607                                  <1>       ; edi = VBE3SAVERESTOREBLOCK
10608                                  <1>
10609 00010BF5 E8420F0000                <1>       call   transfer_from_user_buffer
10610 00010BFA 72AE                      <1>       jc     short sysvideo_28_9 ; error
10611                                  <1>
10612 00010BFC 89D9                      <1>       mov    ecx, ebx ; Function 4F04h options
10613 00010BFE EB9D                      <1>       jmp    short sysvideo_28_10 ; 23/01/2021
10614                                  <1>
10615                                  <1> sysvideo_28_1:
10616 00010C00 31DB                      <1>       xor    ebx, ebx ; 0 ; use kernel's buffer
10617                                  <1> sysvideo_28_11:
10618                                  <1>       ; 24/01/2021
10619 00010C02 803D[54090000]03          <1>       cmp    byte [vbe3], 3
10620 00010C09 7405                      <1>       je     short sysvideo_28_2
10621                                  <1>
10622                                  <1>       ; VESA VBE2 (BOCHS/QEMU/VBOX) video bios
10623 00010C0B E9C82FFFFF                <1>       jmp    _vbe_biosfn_save_restore_state
10624                                  <1> sysvideo_28_2:
10625                                  <1>       ;24/01/2021
10626                                  <1>       ;mov   eax, 4F04h  ; Save/Restore vstate
10627                                  <1>       ; VESA VBE3 video bios
10628 00010C10 E9210EFFFF                <1>       jmp    _vbe3_pmfn_save_restore_state
10629                                  <1>
10630                                  <1> sysvideo_28_12:
10631                                  <1>       ; Save
10632                                  <1>       ;mov   edi, VBE3SAVERESTOREBLOCK
10633                                  <1>
10634                                  <1>       ;movzx ecx, dl ; options
10635 00010C15 56                        <1>       push   esi
10636 00010C16 53                        <1>       push   ebx
10637                                  <1>       ; 23/01/2021
10638 00010C17 B201                      <1>       mov    dl, 1 ; save state
10639 00010C19 E8E2FFFFFF                <1>       call   sysvideo_28_1
10640 00010C1E 5A                        <1>       pop    edx ; 'sysvideo' options
10641 00010C1F 5F                        <1>       pop    edi ; user's video state buffer
10642                                  <1>
10643 00010C20 6683F84F                  <1>       cmp    ax, 4Fh ; successful ?
10644 00010C24 751E                      <1>       jne    short sysvideo_28_14 ; no !
10645                                  <1>
10646                                  <1>       ; ecx = cl = options
10647 00010C26 E84430FFFF                <1>       call   vbe_srs_gbs
10648                                  <1>       ; ebx = state buffer size (data size)
10649                                  <1>
10650 00010C2B 89D9                      <1>       mov    ecx, ebx ; transfer count
10651                                  <1>
10652 00010C2D F6C240                    <1>       test   dl, 64 ; verify without transfer
10653 00010C30 750C                      <1>       jnz    short sysvideo_28_13 ; yes
10654                                  <1>
10655                                  <1>       ;mov   edi, esi
10656 00010C32 BE00760900                <1>       mov    esi, VBE3SAVERESTOREBLOCK
10657 00010C37 E8B60E0000                <1>       call   transfer_to_user_buffer
10658 00010C3C 7206                      <1>       jc     short sysvideo_28_14
10659                                  <1> sysvideo_28_13:
10660 00010C3E 890D[64030300]            <1>       mov    [u.r0], ecx
10661                                  <1> sysvideo_28_14:
```

```
10662 00010C44 E9B3CCFFFF          <1>       jmp    sysret
10663                              <1>
10664                              <1> sysvideo_29:
10665                              <1>       ; 18/01/2021
10666                              <1>       ; BH = 13
10667                              <1>       ; Save/Restore std VGA video state
10668                              <1>
10669                              <1>       ; bl = 0..3
10670                              <1>       ;      save to or restore from
10671                              <1>       ;      system buffer, VBE3VIDEOSTATE
10672                              <1>       ;      ECX = VideoStateID for restoring
10673                              <1>       ; bl = 4..7
10674                              <1>       ;      save to or restore from
10675                              <1>       ;      user buffer pointed by ECX
10676                              <1>
10677 00010C49 80FB03             <1>       cmp    bl, 3
10678 00010C4C 7776               <1>       ja     short sysvideo_29_6
10679                              <1>
10680                              <1>       ; source or destination is kernel/system buffer
10681                              <1>
10682 00010C4E 803D[80120300]00    <1>       cmp    byte [srvsf], 0 ; srs permission flag
10683 00010C55 7668               <1>       jna    short sysvideo_29_5 ; not permitted
10684                              <1>
10685 00010C57 F6C301             <1>       test   bl, 1
10686 00010C5A 7437               <1>       jz     short sysvideo_29_2  ; Save
10687                              <1>
10688                              <1>       ; Restore
10689 00010C5C 3B0D[82120300]      <1>       cmp    ecx, [VideoStateID]
10690 00010C62 755B               <1>       jne    short sysvideo_29_5 ; not correct !
10691 00010C64 80FB01             <1>       cmp    bl, 1
10692 00010C67 7709               <1>       ja     short sysvideo_29_0
10693                              <1>       ; bl = 1
10694 00010C69 BB6E000000          <1>       mov    ebx, 110
10695 00010C6E B103               <1>       mov    cl, 3 ; ctrl, vbios data
10696 00010C70 EB07               <1>       jmp    short sysvideo_29_1
10697                              <1> sysvideo_29_0:
10698                              <1>       ; bl  = 3
10699 00010C72 BB72030000          <1>       mov    ebx, 882
10700 00010C77 B107               <1>       mov    cl, 7 ; ctrl, vbios data, dac
10701                              <1> sysvideo_29_1:
10702 00010C79 3A0D[81120300]      <1>       cmp    cl, [srvso]
10703 00010C7F 753E               <1>       jne    short sysvideo_29_5 ; not correct !
10704                              <1>
10705 00010C81 BE00580900          <1>       mov    esi, VBE3VIDEOSTATE ; 22/01/2021
10706 00010C86 E87E31FFFF          <1>       call   biosfn_restore_video_state
10707 00010C8B 891D[64030300]      <1>       mov    [u.r0], ebx ; video state size (bytes)
10708                              <1>       ;jmp   sysret
10709 00010C91 EB2C               <1>       jmp    short sysvideo_29_5
10710                              <1> sysvideo_29_2:
10711                              <1>       ;mov   esi, ecx
10712 00010C93 BF00580900          <1>       mov    edi, VBE3VIDEOSTATE
10713                              <1>
10714 00010C98 B107               <1>       mov    cl, 7 ; ctrl, vbios data, dac
10715 00010C9A 08DB               <1>       or     bl, bl
10716 00010C9C 7502               <1>       jnz    short sysvideo_29_3 ; bl = 2
10717                              <1>       ; bl = 0
10718 00010C9E B103               <1>       mov    cl, 3 ; ctrl, vbios data
10719                              <1> sysvideo_29_3:
10720 00010CA0 E8F62FFFFF          <1>       call   biosfn_save_video_state
10721                              <1> sysvideo_28_15:
10722                              <1>       ; use timer ticks as VideoStateID
10723 00010CA5 A1[408A0100]        <1>       mov    eax, [TIMER_LH]
10724 00010CAA 21C0               <1>       and    eax, eax
10725 00010CAC 7501               <1>       jnz    short sysvideo_29_4
10726 00010CAE 40                 <1>       inc    eax
10727                              <1> sysvideo_29_4:
10728 00010CAF 880D[81120300]      <1>       mov    [srvso], cl
10729 00010CB5 A3[82120300]        <1>       mov    [VideoStateID], eax
10730 00010CBA A3[64030300]        <1>       mov    [u.r0], eax
10731                              <1> sysvideo_29_5:
10732 00010CBF E938CCFFFF          <1>       jmp    sysret
10733                              <1>
10734                              <1> sysvideo_29_6:
10735 00010CC4 80FB07             <1>       cmp    bl, 7
10736 00010CC7 77F6               <1>       ja     short sysvideo_29_5 ; invalid sub function
10737                              <1>
10738 00010CC9 89CE               <1>       mov    esi, ecx
10739 00010CCB BF00760900          <1>       mov    edi, VBE3SAVERESTOREBLOCK
10740                              <1>
10741                              <1>       ; source or destination is user buffer
10742 00010CD0 F6C301             <1>       test   bl, 1
10743 00010CD3 7434               <1>       jz     short sysvideo_29_9  ; Save
10744                              <1>
10745                              <1>       ; Restore
10746 00010CD5 803D[80120300]00    <1>       cmp    byte [srvsf], 0 ; srs permission flag
10747 00010CDC 76E1               <1>       jna    short sysvideo_29_5 ; not permitted
10748                              <1>
10749                              <1>       ;mov   esi, ecx
10750                              <1>       ;mov   edi, VBE3SAVERESTOREBLOCK
10751                              <1>
10752 00010CDE 80FB07             <1>       cmp    bl, 7
10753 00010CE1 7409               <1>       je     short sysvideo_29_7
10754                              <1>       ; bl = 5
10755 00010CE3 B303               <1>       mov    bl, 3
10756 00010CE5 B96E000000          <1>       mov    ecx, 110
10757 00010CEA EB05               <1>       jmp    short sysvideo_29_8
10758                              <1> sysvideo_29_7:
10759                              <1>       ; bl = 7
10760 00010CEC B972030000          <1>       mov    ecx, 882
10761                              <1> sysvideo_29_8:
10762 00010CF1 E8460E0000          <1>       call   transfer_from_user_buffer
10763 00010CF6 72C7               <1>       jc     short sysvideo_29_5
10764 00010CF8 890D[64030300]      <1>       mov    [u.r0], ecx
10765 00010CFE 88D9               <1>       mov    cl, bl ; mov cl,7 (mov cl,3)
10766 00010D00 89FE               <1>       mov    esi, edi ; VBE3SAVERESTOREBLOCK
```

```
10767                                  <1>       ; cl = 3 or 7
10768 00010D02 E80231FFFF              <1>       call  biosfn_restore_video_state
10769 00010D07 EBB6                    <1>       jmp   sysvideo_29_5
10770                                  <1>       ;jmp  sysret
10771                                  <1> sysvideo_29_9:
10772                                  <1>       ; Save
10773                                  <1>       ;mov  edi, VBE3SAVERESTOREBLOCK
10774                                  <1>
10775 00010D09 80FB06                  <1>       cmp   bl, 6
10776 00010D0C 7409                    <1>       je    short sysvideo_29_10
10777                                  <1>       ; bl = 4
10778 00010D0E BB6E000000              <1>       mov   ebx, 110
10779 00010D13 B103                    <1>       mov   cl, 3 ; ctrl, vbios data
10780 00010D15 EB07                    <1>       jmp   short sysvideo_29_11
10781                                  <1> sysvideo_29_10:
10782                                  <1>       ; bl = 6
10783 00010D17 BB72030000              <1>       mov   ebx, 882
10784 00010D1C B107                    <1>       mov   cl, 7 ; ctrl, vbios data, dac
10785                                  <1> sysvideo_29_11:
10786 00010D1E E8782FFFFF              <1>       call  biosfn_save_video_state
10787                                  <1>
10788 00010D23 89D9                    <1>       mov   ecx, ebx ; transfer count
10789 00010D25 89F7                    <1>       mov   edi, esi
10790 00010D27 BE00760900              <1>       mov   esi, VBE3SAVERESTOREBLOCK
10791                                  <1>
10792                                  <1>       ;call  transfer_to_user_buffer
10793                                  <1>       ;jc    short sysvideo_29_5
10794                                  <1>       ;mov   [u.r0], ecx ; transfer count
10795                                  <1>       ;;jmp  sysret
10796                                  <1>       ;jmp   short sysvideo_29_5
10797                                  <1>
10798 00010D2C EB1A                    <1>       jmp   short sysvideo_29_12
10799                                  <1>
10800                                  <1> sysvideo_30:
10801 00010D2E 80FF0F                  <1>       cmp   bh, 15
10802 00010D31 7722                    <1>       ja    short sysvideo_31 ; invalid function
10803                                  <1>
10804                                  <1>       ; BH = 15
10805                                  <1>       ; Copy VESA EDID to user's buffer
10806                                  <1>
10807 00010D33 803D[2F430000]4F        <1>       cmp   byte [edid], 4Fh
10808 00010D3A 7519                    <1>       jne   short sysvideo_31 ; not ready !
10809                                  <1>
10810                                  <1>       ;and  ecx, ecx
10811                                  <1>       ;jz   short sysvideo_31
10812                                  <1>
10813                                  <1>       ; ecx = user's buffer address
10814 00010D3C 89CF                    <1>       mov   edi, ecx
10815 00010D3E BE[9C110300]            <1>       mov   esi, edid_info
10816 00010D43 B980000000              <1>       mov   ecx, 128 ; 128 bytes
10817                                  <1> sysvideo_29_12:
10818 00010D48 E8A50D0000              <1>       call  transfer_to_user_buffer
10819 00010D4D 7206                    <1>       jc    short sysvideo_31
10820                                  <1>
10821 00010D4F 890D[64030300]          <1>       mov   [u.r0], ecx ; EDID size, 128 bytes
10822                                  <1> sysvideo_31:
10823 00010D55 E9A2CBFFFF              <1>       jmp   sysret
10824                                  <1>
10825                                  <1> mkdir:
10826                                  <1>       ; 04/12/2015 (14 byte directory names)
10827                                  <1>       ; 12/10/2015
10828                                  <1>       ; 17/06/2015 (Retro UNIX 386 v1 - Beginning)
10829                                  <1>       ; 29/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
10830                                  <1>       ;
10831                                  <1>       ; 'mkdir' makes a directory entry from the name pointed to
10832                                  <1>       ; by u.namep into the current directory.
10833                                  <1>       ;
10834                                  <1>       ; INPUTS ->
10835                                  <1>       ;    u.namep - points to a file name
10836                                  <1>       ;                 that is about to be a directory entry.
10837                                  <1>       ;    ii - current directory's i-number.
10838                                  <1>       ; OUTPUTS ->
10839                                  <1>       ;    u.dirbuf+2 - u.dirbuf+10 - contains file name.
10840                                  <1>       ;    u.off - points to entry to be filled
10841                                  <1>       ;          in the current directory
10842                                  <1>       ;    u.base - points to start of u.dirbuf.
10843                                  <1>       ;    r1 - contains i-number of current directory
10844                                  <1>       ;
10845                                  <1>       ; ((AX = R1)) output
10846                                  <1>       ;
10847                                  <1>       ;    (Retro UNIX Prototype : 11/11/2012, UNIXCOPY.ASM)
10848                                  <1>         ;    ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
10849                                  <1>       ;
10850                                  <1>
10851                                  <1>       ; 17/06/2015 - 32 bit modifications (Retro UNIX 386 v1)
10852 00010D5A 31C0                    <1>       xor   eax, eax
10853 00010D5C BF[9A030300]            <1>       mov     edi, u.dirbuf+2
10854 00010D61 89FE                    <1>       mov   esi, edi
10855 00010D63 AB                      <1>       stosd
10856 00010D64 AB                      <1>       stosd
10857                                  <1>       ; 04/12/2015 (14 byte directory names)
10858 00010D65 AB                      <1>       stosd
10859 00010D66 66AB                    <1>       stosw
10860                                  <1>             ; jsr r0,copyz; u.dirbuf+2; u.dirbuf+10. / clear this
10861 00010D68 89F7                    <1>       mov   edi, esi ; offset to u.dirbuf
10862                                  <1>       ; 12/10/2015 ([u.namep] -> ebp)
10863                                  <1>       ;mov   ebp, [u.namep]
10864 00010D6A E80D030000              <1>       call  trans_addr_nmbp ; convert virtual address to physical
10865                                  <1>             ; esi = physical address (page start + offset)
10866                                  <1>             ; ecx = byte count in the page (1 - 4096)
10867                                  <1>       ; edi = offset to u.dirbuf (edi is not modified in trans_addr_nm)
10868                                  <1>             ; mov u.namep,r2 / r2 points to name of directory entry
10869                                  <1>             ; mov $u.dirbuf+2,r3 / r3 points to u.dirbuf+2
10870                                  <1> mkdir_1: ; 1:
10871 00010D6F 45                      <1>       inc   ebp ; 12/10/2015
```

```
10872                                  <1>       ;
10873                                  <1>       ; / put characters in the directory name in u.dirbuf+2 - u.dirbuf+10
10874                                  <1>        ; 01/08/2013
10875 00010D70 AC                      <1>       lodsb
10876                                  <1>             ; movb (r2)+,r1 / move character in name to r1
10877 00010D71 20C0                    <1>       and    al, al
10878 00010D73 7427                    <1>       jz     short mkdir_3
10879                                  <1>             ; beq 1f / if null, done
10880 00010D75 3C2F                    <1>       cmp    al, '/'
10881                                  <1>             ; cmp r1,$'/ / is it a "/"?
10882 00010D77 7414                    <1>       je     short mkdir_err
10883                                  <1>       ;je    error
10884                                  <1>             ; beq error9 / yes, error
10885                                  <1>       ; 12/10/2015
10886 00010D79 6649                    <1>       dec    cx
10887 00010D7B 7505                    <1>       jnz    short mkdir_2
10888                                  <1>       ; 12/10/2015 ([u.namep] -> ebp)
10889 00010D7D E800030000              <1>       call   trans_addr_nm ; convert virtual address to physical
10890                                  <1>             ; esi = physical address (page start + offset)
10891                                  <1>             ; ecx = byte count in the page
10892                                  <1>       ; edi = offset to u.dirbuf (edi is not modified in trans_addr_nm)
10893                                  <1> mkdir_2:
10894 00010D82 81FF[A8030300]          <1>       cmp     edi, u.dirbuf+16 ; ; 04/12/2015 (10 -> 16)
10895                                  <1>             ; cmp r3,$u.dirbuf+10. / have we reached the last slot for
10896                                  <1>                                    ; / a char?
10897 00010D88 74E5                    <1>       je     short mkdir_1
10898                                  <1>             ; beq 1b / yes, go back
10899 00010D8A AA                      <1>       stosb
10900                                  <1>             ; movb r1,(r3)+ / no, put the char in the u.dirbuf
10901 00010D8B EBE2                    <1>       jmp    short mkdir_1
10902                                  <1>             ; br 1b / get next char
10903                                  <1> mkdir_err:
10904                                  <1>       ; 17/06/2015
10905 00010D8D C705[C8030300]1300-     <1>       mov    dword [u.error], ERR_NOT_DIR ; 'not a valid directory !'
10905 00010D95 0000                    <1>
10906 00010D97 E940CBFFFF              <1>       jmp    error
10907                                  <1>
10908                                  <1> mkdir_3: ; 1:
10909 00010D9C A1[78030300]            <1>       mov    eax, [u.dirp]
10910 00010DA1 A3[80030300]            <1>       mov    [u.off], eax
10911                                  <1>             ; mov u.dirp,u.off / pointer to empty current directory
10912                                  <1>                                ; / slot to u.off
10913                                  <1> wdir: ; 29/04/2013
10914 00010DA6 C705[84030300]-         <1>         mov     dword [u.base], u.dirbuf
10914 00010DAC [98030300]              <1>
10915                                  <1>             ; mov $u.dirbuf,u.base / u.base points to created file name
10916 00010DB0 C705[88030300]1000-     <1>         mov     dword [u.count], 16 ; 04/12/2015 (10 -> 16)
10916 00010DB8 0000                    <1>
10917                                  <1>             ; mov $10.,u.count / u.count = 10
10918 00010DBA 66A1[51040300]          <1>       mov    ax, [ii]
10919                                  <1>             ; mov ii,r1 / r1 has i-number of current directory
10920 00010DC0 B201                    <1>       mov    dl, 1 ; owner flag mask ; RETRO UNIX 8086 v1 modification !
10921 00010DC2 E8741D0000              <1>       call   access
10922                                  <1>             ; jsr r0,access; 1 / get i-node and set its file up
10923                                  <1>                               ; / for writing
10924                                  <1>       ; AX = i-number of current directory
10925                                  <1>       ; 01/08/2013
10926 00010DC7 FE05[C6030300]          <1>       inc     byte [u.kcall] ; the caller is 'mkdir' sign
10927 00010DCD E8820F0000              <1>       call   writei
10928                                  <1>             ; jsr r0,writei / write into directory
10929 00010DD2 C3                      <1>       retn
10930                                  <1>             ; rts r0
10931                                  <1>
10932                                  <1> sysexec:
10933                                  <1>       ; 18/11/2017
10934                                  <1>       ; 14/11/2017
10935                                  <1>       ; 13/11/2017
10936                                  <1>       ; 24/10/2016, 04/01/2017
10937                                  <1>       ; 24/04/2016 - TRDOS 386 (TRDOS v2.0)
10938                                  <1>       ; 23/06/2015 - 23/10/2015 (Retro UNIX 386 v1)
10939                                  <1>       ; 03/06/2013 - 06/12/2013 (Retro UNIX 8086 v1)
10940                                  <1>       ;
10941                                  <1>       ; 'sysexec' initiates execution of a file whose path name if
10942                                  <1>       ; pointed to by 'name' in the sysexec call.
10943                                  <1>       ; 'sysexec' performs the following operations:
10944                                  <1>       ;    1. obtains i-number of file to be executed via 'namei'.
10945                                  <1>       ;    2. obtains i-node of file to be exceuted via 'iget'.
10946                                  <1>       ;    3. sets trap vectors to system routines.
10947                                  <1>       ;    4. loads arguments to be passed to executing file into
10948                                  <1>       ;      highest locations of user's core
10949                                  <1>       ;    5. puts pointers to arguments in locations immediately
10950                                  <1>       ;      following arguments.
10951                                  <1>       ;    6.      saves number of arguments in next location.
10952                                  <1>       ;    7. intializes user's stack area so that all registers
10953                                  <1>       ;      will be zeroed and the PS is cleared and the PC set
10954                                  <1>       ;      to core when 'sysret' restores registers
10955                                  <1>       ;      and does an rti.
10956                                  <1>       ;    8. inializes u.r0 and u.sp
10957                                  <1>       ;    9. zeros user's core down to u.r0
10958                                  <1>       ;   10.      reads executable file from storage device into core
10959                                  <1>       ;      starting at location 'core'.
10960                                  <1>       ;   11.      sets u.break to point to end of user's code with
10961                                  <1>       ;      data area appended.
10962                                  <1>       ;   12.      calls 'sysret' which returns control at location
10963                                  <1>       ;      'core' via 'rti' instruction.
10964                                  <1>       ;
10965                                  <1>       ; Calling sequence:
10966                                  <1>       ;      sysexec; namep; argp
10967                                  <1>       ; Arguments:
10968                                  <1>       ;      namep - points to pathname of file to be executed
10969                                  <1>       ;      argp  - address of table of argument pointers
10970                                  <1>       ;      argp1... argpn - table of argument pointers
10971                                  <1>       ;      argp1:<...0> ... argpn:<...0> - argument strings
10972                                  <1>       ; Inputs: (arguments)
10973                                  <1>       ; Outputs: -
```

```
10974                                  <1>      ; ..............................................................
10975                                  <1>      ;
10976                                  <1>      ; Retro UNIX 386 v1 modification:
10977                                  <1>      ;     User application runs in it's own virtual space
10978                                  <1>      ;     which is izolated from kernel memory (and other
10979                                  <1>      ;     memory pages) via 80386   paging in ring 3
10980                                  <1>      ;     privilige mode. Virtual start address is always 0.
10981                                  <1>      ;     User's core memory starts at linear address 400000h
10982                                  <1>      ;     (the end of the 1st 4MB).
10983                                  <1>      ;
10984                                  <1>      ; Retro UNIX 8086 v1 modification:
10985                                  <1>      ;     user/application segment and system/kernel segment
10986                                  <1>      ;     are different and sysenter/sysret/sysrele routines
10987                                  <1>      ;     are different (user's registers are saved to
10988                                  <1>      ;     and then restored from system's stack.)
10989                                  <1>      ;
10990                                  <1>      ;     NOTE: Retro UNIX 8086 v1 'arg2' routine gets these
10991                                  <1>      ;           arguments which were in these registers;
10992                                  <1>      ;           but, it returns by putting the 1st argument
10993                                  <1>      ;           in 'u.namep' and the 2nd argument
10994                                  <1>      ;           on top of stack. (1st argument is offset of the
10995                                  <1>      ;           file/path name in the user's program segment.)
10996                                  <1>
10997                                  <1>      ;call  arg2
10998                                  <1>      ; * name - 'u.namep' points to address of file/path name
10999                                  <1>      ;          in the user's program segment ('u.segmnt')
11000                                  <1>      ;          with offset in BX register (as sysopen argument 1).
11001                                  <1>      ; * argp - sysexec argument 2 is in CX register
11002                                  <1>      ;          which is on top of stack.
11003                                  <1>      ;
11004                                  <1>           ; jsr r0,arg2 / arg0 in u.namep,arg1 on top of stack
11005                                  <1>
11006                                  <1>      ; 23/06/2015 (32 bit modifications)
11007                                  <1>
11008                                  <1>      ;; 13/11/2017
11009                                  <1>      ;;mov  [u.namep], ebx ; argument 1
11010                                  <1>        ; 18/10/2015
11011 00010DD3 890D[4C040300]          <1>      mov     [argv], ecx  ; * ; argument 2
11012                                  <1>
11013                                  <1>      ; 13/11/2017
11014 00010DD9 89DE                    <1>      mov    esi, ebx
11015 00010DDB E88E210000              <1>      call   set_working_path_x
11016 00010DE0 7319                    <1>      jnc    short sysexec_0
11017                                  <1>
11018                                  <1>      ;; 'bad command or file name'
11019                                  <1>      ;mov   eax, ERR_BAD_CMD_ARG ; 01h ; TRDOS 8086
11020                                  <1>
11021                                  <1>      ; 'file not found !' error
11022 00010DE2 B802000000              <1>      mov    eax, ERR_NOT_FOUND ; 02h ; TRDOS 8086
11023                                  <1> sysexec_not_found_err:
11024                                  <1> sysexec_access_error:
11025                                  <1> sysexec_ext_error:
11026 00010DE7 A3[64030300]            <1>      mov    [u.r0], eax
11027 00010DEC A3[C8030300]            <1>      mov    [u.error], eax
11028 00010DF1 E84D220000              <1>      call   reset_working_path
11029 00010DF6 E9E1CAFFFF              <1>      jmp    error
11030                                  <1>
11031                                  <1> sysexec_0:
11032                                  <1>      ; 13/11/2017
11033                                  <1>      ;mov   esi, FindFile_Name
11034 00010DFB 66B80018                <1>        mov  ax, 1800h ; Only files
11035 00010DFF E87D86FFFF              <1>      call   find_first_file
11036 00010E04 72E1                    <1>      jc     short sysexec_not_found_err ; eax = 2
11037                                  <1>
11038                                  <1>      ; check_ file attributes
11039                                  <1>      ; (attribute bits = 00ADVSHR) ; 18h = Directory+Volume
11040                                  <1>      ; BL = Attributes byte
11041                                  <1>
11042 00010E06 F6C306                  <1>        test bl, 6  ; system file or hidden file (S+H)
11043                                  <1>      ;jz    short sysexec_0ext
11044 00010E09 7417                    <1>      jz     short sysexec_1 ; yes
11045                                  <1>
11046                                  <1>      ; 13/11/2017
11047                                  <1>      ; /// TRDOS386 permission check for multiuser mode ///
11048                                  <1>      ; SYSTEM file or HIDDEN file !!
11049                                  <1>      ; (Only super user has permission to run this file.)
11050                                  <1>
11051                                  <1>      ; ([u.uid]=0 for super user or root in multiuser mode)
11052                                  <1>      ; ([u.uid]=0 for any users in singleuser mode)
11053 00010E0B 803D[B0030300]00        <1>      cmp    byte [u.uid], 0 ; Super User ([u.uid]=0) ?
11054                                  <1>      ;jna   short sysexec_0ext
11055 00010E12 760E                    <1>      jna    short sysexec_1 ; yes
11056                                  <1>
11057                                  <1>      ; 'permission denied !' error
11058 00010E14 B80B000000              <1>        mov  eax, ERR_FILE_ACCESS  ; 11 = ERR_PERM_DENIED
11059 00010E19 EBCC                    <1>        jmp  short sysexec_access_error
11060                                  <1>
11061                                  <1> sysexec_not_exf:
11062                                  <1>      ; 'not executable file !' error
11063 00010E1B B816000000              <1>      mov    eax, ERR_NOT_EXECUTABLE
11064 00010E20 EBC5                    <1>      jmp    sysexec_ext_error
11065                                  <1>
11066                                  <1> ;sysexec_0ext:
11067                                  <1> sysexec_1:
11068                                  <1>      ; 18/11/2017
11069 00010E22 BE[68930100]            <1>      mov    esi, FindFile_Name
11070                                  <1>      ; 13/11/2017
11071                                  <1>      ; check program file name extension
11072                                  <1>      ; ('.PRG' for current TRDOS version)
11073 00010E27 E8F8A0FFFF              <1>      call   check_prg_filename_ext
11074 00010E2C 72ED                    <1>      jc     short sysexec_not_exf
11075                                  <1>
11076                                  <1>      ; 18/11/2017
11077 00010E2E 3C50                    <1>      cmp    al, 'P'
11078 00010E30 75E9                    <1>      jne    short sysexec_not_exf
```

```
11079                                  <1>
11080                                  <1>       ; '.PRG' extension is OK.
11081                                  <1>       ; Only '.PRG' files are valid program files
11082                                  <1>       ; for current TRDOS 386 version.
11083                                  <1>
11084 00010E32 8B15[94930100]          <1>       mov    edx, [FindFile_DirEntry+DirEntry_FileSize]
11085 00010E38 66A1[8C930100]          <1>       mov    ax, [FindFile_DirEntry+DirEntry_FstClusHI]
11086 00010E3E C1E010                  <1>       shl    eax, 16
11087 00010E41 66A1[92930100]          <1>       mov    ax, [FindFile_DirEntry+DirEntry_FstClusLO]
11088                                  <1>       ; EAX = First Cluster number
11089                                  <1>       ; EDX = File Size
11090                                  <1>
11091 00010E47 A3[51040300]            <1>       mov    [ii], eax
11092 00010E4C 8915[55040300]          <1>       mov    [i.size], edx
11093                                  <1>
11094                                  <1> ;sysexec_1:
11095                                  <1>       ; 13/11/2017 - TRDOS 386 (TRDOS v2.0)
11096                                  <1>       ; 24/06/2015 - 23/10/2015 (Retro UNIX 386 v1)
11097                                  <1>         ; Moving arguments to the end of [u.upage]
11098                                  <1>       ; (by regarding page borders in user's memory space)
11099                                  <1>       ;
11100                                  <1>       ; 10/10/2015
11101                                  <1>       ; 21/07/2015
11102 00010E52 89E5                   <1>       mov    ebp, esp ; (**)
11103                                  <1>       ; 18/10/2015
11104 00010E54 89EF                   <1>       mov    edi, ebp
11105 00010E56 B900010000             <1>       mov    ecx, MAX_ARG_LEN ; 256
11106                                  <1>       ;sub   edi, MAX_ARG_LEN ; 256
11107 00010E5B 29CF                   <1>       sub    edi, ecx
11108 00010E5D 89FC                   <1>       mov    esp, edi ; *!*
11109 00010E5F 31C0                   <1>       xor    eax, eax
11110 00010E61 A3[8C030300]           <1>       mov    [u.nread], eax ; 0
11111 00010E66 66A3[4A040300]         <1>       mov    [argc], ax ; 0 ; 13/11/2017
11112 00010E6C 49                     <1>       dec    ecx ; 256 - 1
11113 00010E6D 890D[88030300]         <1>       mov    [u.count], ecx ; MAX_ARG_LEN - 1 ; 255
11114                                  <1>       ;mov   dword [u.count], MAX_ARG_LEN - 1 ; 255
11115                                  <1> sysexec_2:
11116 00010E73 8B35[4C040300]          <1>       mov    esi, [argv] ; 18/10/2015
11117 00010E79 E866000000              <1>       call   get_argp
11118 00010E7E B904000000              <1>       mov    ecx, 4 ; mov ecx, 4
11119                                  <1> sysexec_3:
11120 00010E83 21C0                   <1>       and    eax, eax
11121 00010E85 0F846F050000           <1>         jz     sysexec_6
11122                                  <1>       ; 18/10/2015
11123 00010E8B 010D[4C040300]         <1>       add    [argv], ecx ; 4
11124 00010E91 66FF05[4A040300]       <1>       inc    word [argc]
11125                                  <1>       ;
11126 00010E98 A3[84030300]           <1>       mov    [u.base], eax
11127                                  <1>       ; 23/10/2015
11128 00010E9D 66C705[C4030300]00-     <1>       mov    word [u.pcount], 0
11128 00010EA5 00                      <1>
11129                                  <1> sysexec_4:
11130 00010EA6 E8E70B0000              <1>       call   cpass ; get a character from user's core memory
11131 00010EAB 750E                    <1>         jnz  short sysexec_5
11132                                  <1>              ; (max. 255 chars + null)
11133                                  <1>       ; 18/10/2015
11134 00010EAD 28C0                   <1>       sub    al, al
11135 00010EAF AA                     <1>       stosb
11136 00010EB0 FF05[8C030300]         <1>       inc    dword [u.nread]
11137 00010EB6 E93F050000             <1>       jmp    sysexec_6 ; 24/04/2016
11138                                  <1> sysexec_5:
11139 00010EBB AA                     <1>       stosb
11140 00010EBC 20C0                   <1>       and    al, al
11141 00010EBE 75E6                   <1>       jnz    short sysexec_4
11142 00010EC0 B904000000             <1>       mov    ecx, 4
11143 00010EC5 390D[48040300]         <1>       cmp    [ncount], ecx ; 4
11144 00010ECB 72A6                   <1>       jb     short sysexec_2
11145 00010ECD 8B35[44040300]         <1>       mov    esi, [nbase]
11146 00010ED3 010D[44040300]         <1>       add    [nbase], ecx ; 4
11147 00010ED9 66290D[48040300]       <1>       sub    [ncount], cx
11148 00010EE0 8B06                   <1>       mov    eax, [esi]
11149 00010EE2 EB9F                   <1>       jmp    short sysexec_3
11150                                  <1>
11151                                  <1> get_argp:
11152                                  <1>       ; 14/11/2017 - TRDOS 386 (TRDOS v2.0)
11153                                  <1>       ; 18/10/2015 (nbase, ncount)
11154                                  <1>       ; 21/07/2015
11155                                  <1>       ; 24/06/2015 (Retro UNIX 386 v1)
11156                                  <1>       ; Get (virtual) address of argument from user's core memory
11157                                  <1>       ;
11158                                  <1>       ; INPUT:
11159                                  <1>       ;      esi = virtual address of argument pointer
11160                                  <1>       ; OUTPUT:
11161                                  <1>       ;      eax = virtual address of argument
11162                                  <1>       ;
11163                                  <1>       ; Modified registers: EAX, EBX, ECX, EDX, ESI
11164                                  <1>       ;
11165 00010EE4 833D[BC030300]00       <1>       cmp     dword [u.ppgdir], 0 ; /etc/init ?
11166                                  <1>                                 ; (the caller is kernel)
11167 00010EEB 7667                   <1>         jna     short get_argpk
11168                                  <1>       ;
11169 00010EED 89F3                   <1>       mov    ebx, esi
11170 00010EEF E8F453FFFF             <1>       call   get_physical_addr ; get physical address
11171 00010EF4 0F8289000000           <1>         jc     get_argp_err
11172 00010EFA A3[44040300]           <1>       mov    [nbase], eax ; physical address
11173 00010EFF 66890D[48040300]       <1>       mov    [ncount], cx ; remain byte count in page (1-4096)
11174 00010F06 B804000000             <1>       mov    eax, 4 ; 21/07/2015
11175 00010F0B 6639C1                 <1>       cmp    cx, ax ; 4
11176 00010F0E 735D                   <1>       jnb    short get_argp2
11177 00010F10 89F3                   <1>       mov    ebx, esi
11178 00010F12 01CB                   <1>       add    ebx, ecx
11179 00010F14 E8CF53FFFF             <1>       call   get_physical_addr ; get physical address
11180 00010F19 7268                   <1>       jc     short get_argp_err
11181                                  <1>       ;push  esi
11182 00010F1B 89C6                   <1>       mov    esi, eax
```

```
11183 00010F1D 66870D[48040300]    <1>       xchg   cx, [ncount]
11184 00010F24 8735[44040300]      <1>       xchg   esi, [nbase]
11185 00010F2A B504                <1>       mov    ch, 4
11186 00010F2C 28CD                <1>       sub    ch, cl
11187                              <1> get_argp0:
11188 00010F2E AC                  <1>       lodsb
11189 00010F2F 6650                <1>       push   ax
11190 00010F31 FEC9                <1>       dec    cl
11191 00010F33 75F9                <1>         jnz     short get_argp0
11192 00010F35 8B35[44040300]      <1>       mov    esi, [nbase]
11193                              <1>       ; 21/07/2015
11194 00010F3B 0FB6C5              <1>       movzx  eax, ch
11195 00010F3E 0105[44040300]      <1>       add    [nbase], eax
11196 00010F44 662905[48040300]    <1>       sub    [ncount], ax
11197                              <1> get_argp1:
11198 00010F4B AC                  <1>       lodsb
11199 00010F4C FECD                <1>       dec    ch
11200 00010F4E 7447                <1>         jz      short get_argp3
11201 00010F50 6650                <1>         push ax
11202 00010F52 EBF7                <1>       jmp     short get_argp1
11203                              <1> get_argpk:
11204                              <1>       ; Argument is in kernel's memory space
11205 00010F54 66C705[48040300]00- <1>       mov    word [ncount], PAGE_SIZE ; 4096
11205 00010F5C 10                  <1>
11206 00010F5D 8935[44040300]      <1>       mov    [nbase], esi
11207 00010F63 8305[44040300]04    <1>       add    dword [nbase], 4
11208 00010F6A 8B06                <1>       mov    eax, [esi] ; virtual addr. = physical addr.
11209 00010F6C C3                  <1>       retn
11210                              <1> get_argp2:
11211                              <1>       ; 21/07/2015
11212                              <1>       ;mov   eax, 4
11213 00010F6D 8B15[44040300]      <1>       mov    edx, [nbase] ; 18/10/2015
11214 00010F73 0105[44040300]      <1>       add    [nbase], eax
11215 00010F79 662905[48040300]    <1>       sub    [ncount], ax
11216                              <1>       ;
11217 00010F80 8B02                <1>       mov    eax, [edx]
11218 00010F82 C3                  <1>       retn
11219                              <1> get_argp_err:
11220 00010F83 A3[C8030300]        <1>       mov    [u.error], eax
11221                              <1>       ; 14/11/2017
11222 00010F88 B801000000          <1>       mov    eax, ERR_BAD_CMD_ARG ; 01h ; TRDOS 8086
11223 00010F8D A3[64030300]        <1>       mov    [u.r0], eax
11224 00010F92 E945C9FFFF          <1>       jmp    error
11225                              <1> get_argp3:
11226 00010F97 B103                <1>       mov    cl, 3
11227                              <1> get_argp4:
11228 00010F99 C1E008              <1>       shl    eax, 8
11229 00010F9C 665A                <1>       pop    dx
11230 00010F9E 88D0                <1>       mov    al, dl
11231 00010FA0 E2F7                <1>         loop    get_argp4
11232                              <1>       ;pop   esi
11233 00010FA2 C3                  <1>       retn
11234                              <1>
11235                              <1> sysstat:
11236                              <1>       ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
11237                              <1>       ; temporary !
11238 00010FA3 B801000000          <1>       mov    eax, ERR_INV_FNUMBER ; 'invalid function number !'
11239 00010FA8 A3[C8030300]        <1>         mov     [u.error], eax
11240 00010FAD A3[64030300]        <1>         mov     [u.r0], eax
11241 00010FB2 E925C9FFFF          <1>       jmp    error
11242                              <1>
11243                              <1> sysfstat:
11244                              <1>       ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
11245                              <1>       ; temporary !
11246 00010FB7 B801000000          <1>       mov    eax, ERR_INV_FNUMBER ; 'invalid function number !'
11247 00010FBC A3[C8030300]        <1>         mov     [u.error], eax
11248 00010FC1 A3[64030300]        <1>         mov     [u.r0], eax
11249 00010FC6 E911C9FFFF          <1>       jmp    error
11250                              <1>
11251                              <1> fclose:
11252                              <1>       ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
11253                              <1>       ;
11254                              <1>       ; 18/06/2015 (Retro UNIX 386 v1 - Beginning)
11255                              <1>       ;            (32 bit offset pointer modification)
11256                              <1>       ; 19/04/2013 - 12/01/2014 (Retro UNIX 8086 v1)
11257                              <1>       ;
11258                              <1>       ; Given the file descriptor (index to the u.fp list)
11259                              <1>       ; 'fclose' first gets the i-number of the file via 'getf'.
11260                              <1>       ; If i-node is active (i-number > 0) the entry in
11261                              <1>       ; u.fp list is cleared. If all the processes that opened
11262                              <1>       ; that file close it, then fsp etry is freed and the file
11263                              <1>       ; is closed. If not a return is taken.
11264                              <1>       ; If the file has been deleted while open, 'anyi' is called
11265                              <1>       ; to see anyone else has it open, i.e., see if it is appears
11266                              <1>       ; in another entry in the fsp table. Upon return from 'anyi'
11267                              <1>       ; a check is made to see if the file is special.
11268                              <1>       ;
11269                              <1>       ; INPUTS ->
11270                              <1>       ;    r1 - contains the file descriptor (value=0,1,2...)
11271                              <1>       ;    u.fp - list of entries in the fsp table
11272                              <1>       ;    fsp - table of entries (4 words/entry) of open files.
11273                              <1>       ; OUTPUTS ->
11274                              <1>       ;    r1 - contains the same file descriptor
11275                              <1>       ;    r2 - contains i-number
11276                              <1>       ;
11277                              <1>       ; ((AX = R1))
11278                              <1>       ; ((Modified registers: eDX, eBX, eCX, eSI, eDI, eBP))
11279                              <1>       ;
11280                              <1>       ; Retro UNIX 8086 v1 modification : CF = 1
11281                              <1>       ;             if i-number of the file is 0. (error)
11282                              <1>       ;
11283                              <1>       ; TRDOS 386 (06/10/2016)
11284                              <1>       ;
11285                              <1>       ; INPUT:
11286                              <1>       ;     EAX = File Handle (File Descriptor, File Index)
```

```
11287                                  <1>       ;
11288                                  <1>       ; OUTPUT:
11289                                  <1>       ;      CF = 1 -> File not open !
11290                                  <1>       ;      CF = 0 -> OK!
11291                                  <1>       ;           EBX = File Number (System)
11292                                  <1>       ;           [cdev] = Logical DOS Drive Number
11293                                  <1>       ;           EAX = File Handle/Number (user)
11294                                  <1>       ;
11295                                  <1>       ; Modified Registers: EBX
11296                                  <1>
11297 00010FCB 50                      <1>       push   eax ; File handle
11298                                  <1>
11299 00010FCC E846000000              <1>       call   getf
11300 00010FD1 0F8245240000            <1>       jc     device_close ; eax = device number
11301                                  <1>
11302 00010FD7 80BB[E6990100]01        <1>       cmp    byte [ebx+OF_MODE], 1 ; open mode ; 0 = empty entry
11303 00010FDE 722E                    <1>       jb     short fclose_1           ; 1 = read, 2 = write
11304                                  <1>
11305 00010FE0 83F801                  <1>       cmp    eax, 1 ; is the first cluster number > 0
11306 00010FE3 7229                    <1>       jb     short fclose_1 ; no, this is empty entry
11307                                  <1>
11308                                  <1> fclose_0:
11309 00010FE5 FE8B[FA990100]          <1>       dec    byte [ebx+OF_OPENCOUNT] ; decrement the number of processes
11310                                  <1>                                    ; that have opened the file
11311 00010FEB 7921                    <1>       jns    short fclose_1 ; jump if not negative (jump if bit 7 is 0)
11312                                  <1>                     ; if all processes haven't closed the file, return
11313                                  <1>       ;
11314                                  <1>       ; eax ; First cluster
11315 00010FED 31C0                    <1>       xor    eax, eax ; 0
11316 00010FEF 8883[E6990100]          <1>       mov    [ebx+OF_MODE], al ; 0 = empty entry
11317                                  <1>       ;mov   [ebx+OF_STATUS], al ; 0 = empty entry
11318 00010FF5 66C1E302                <1>       shl    bx, 2
11319 00010FF9 8983[B4990100]          <1>       mov    [ebx+OF_FCLUSTER], eax ; 0
11320 00010FFF 8983[CC9A0100]          <1>       mov    [ebx+OF_CCLUSTER], eax ; 0
11321                                  <1>       ;mov   [ebx+OF_CCINDEX], eax ; 0
11322 00011005 A3[74030300]            <1>       mov    [u.fofp], eax ; 0
11323 0001100A 66C1EB02                <1>       shr    bx, 2
11324                                  <1> fclose_1: ; 1:
11325 0001100E 58                      <1>       pop    eax ; File handle (File Descriptor, File Index)
11326 0001100F C680[6A030300]00        <1>       mov    byte [eax+u.fp], 0 ; clear that entry in the u.fp list
11327 00011016 C3                      <1>       retn
11328                                  <1>
11329                                  <1> getf:
11330                                  <1>       ; 12/10/2016
11331                                  <1>       ; 11/10/2016
11332                                  <1>       ; 08/10/2016
11333                                  <1>       ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
11334                                  <1>       ; / get the device number and the i-number of an open file
11335                                  <1>       ; 13/05/2015
11336                                  <1>       ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
11337                                  <1>       ; 19/04/2013 - 18/11/2013 (Retro UNIX 8086 v1)
11338                                  <1>       ;
11339 00011017 89C3                    <1>       mov    ebx, eax
11340                                  <1> getf1:
11341 00011019 83FB0A                  <1>       cmp    ebx, 10
11342 0001101C 730A                    <1>         jnb  short getf2
11343 0001101E 8A9B[6A030300]          <1>       mov    bl, [ebx+u.fp]
11344 00011024 08DB                    <1>       or     bl, bl
11345 00011026 7503                    <1>       jnz    short getf3
11346                                  <1> getf2:
11347                                  <1>       ; 'File not open !' error (ax=0)
11348 00011028 29C0                    <1>       sub    eax, eax
11349 0001102A C3                      <1>       retn
11350                                  <1> getf3:
11351 0001102B F6C380                  <1>       test   bl, 80h
11352 0001102E 7530                    <1>       jnz    short getf5 ; device
11353 00011030 FECB                    <1>       dec    bl ; 0 based
11354 00011032 8A83[DC990100]          <1>       mov    al, [ebx+OF_DRIVE]
11355 00011038 A2[46030300]            <1>       mov    [cdev], al
11356 0001103D C0E302                  <1>       shl    bl, 2 ; *4 (dword offset)
11357 00011040 8B83[2C9A0100]          <1>       mov    eax, [ebx+OF_SIZE]
11358 00011046 A3[55040300]            <1>       mov    [i.size], eax ; file size
11359 0001104B 8D83[049A0100]          <1>       lea    eax, [ebx+OF_POINTER] ;12/10/2016
11360 00011051 A3[74030300]            <1>       mov    [u.fofp], eax
11361 00011056 8B83[B4990100]          <1>       mov    eax, [ebx+OF_FCLUSTER]
11362 0001105C C0EB02                  <1>       shr    bl, 2 ; /4 (byte offset)
11363                                  <1> getf4:
11364 0001105F C3                      <1>       retn
11365                                  <1> getf5:
11366                                  <1>       ; get device number
11367 00011060 80E37F                  <1>       and    bl, 7Fh ; 1 to 7Fh
11368 00011063 FECB                    <1>       dec    bl ; 0 based (0 to 7Eh)
11369 00011065 8A83[0E980100]          <1>       mov    al, [ebx+DEV_DRIVER]
11370 0001106B 8AAB[78970100]          <1>       mov    ch, [ebx+DEV_ACCESS]
11371 00011071 8A8B[2C980100]          <1>       mov    cl, [ebx+DEV_OPENMODE]
11372 00011077 80E5FE                  <1>       and    ch, 0FEh ; reset bit 0 ; dev_close
11373 0001107A F9                      <1>       stc ; cf = 1
11374 0001107B C3                      <1>       retn
11375                                  <1>
11376                                  <1> trans_addr_nmbp:
11377                                  <1>       ; 18/10/2015
11378                                  <1>       ; 12/10/2015
11379 0001107C 8B2D[7C030300]          <1>       mov    ebp, [u.namep]
11380                                  <1> trans_addr_nm:
11381                                  <1>       ; Convert virtual (pathname) address to physical address
11382                                  <1>       ; (Retro UNIX 386 v1 feature only !)
11383                                  <1>       ; 18/10/2015
11384                                  <1>       ; 12/10/2015 (u.pnbase & u.pncount has been removed from code)
11385                                  <1>       ; 02/07/2015
11386                                  <1>       ; 17/06/2015
11387                                  <1>       ; 16/06/2015
11388                                  <1>       ;
11389                                  <1>       ; INPUTS:
11390                                  <1>       ;      ebp = pathname address (virtual) ; [u.namep]
11391                                  <1>       ;      [u.pgdir] = user's page directory
```

```
11392                                  <1>       ; OUTPUT:
11393                                  <1>       ;       esi = physical address of the pathname
11394                                  <1>       ;      ecx = remain byte count in the page
11395                                  <1>       ;
11396                                  <1>       ; (Modified registers: EAX, EBX, ECX, EDX, ESI)
11397                                  <1>       ;
11398 00011082 833D[BC030300]00        <1>         cmp     dword [u.ppgdir], 0  ; /etc/init ? (sysexec)
11399 00011089 7618                    <1>       jna    short trans_addr_nmk ; the caller is os kernel;
11400                                  <1>                                   ; it is already physical address
11401 0001108B 50                      <1>       push   eax
11402 0001108C 89EB                    <1>       mov    ebx, ebp ; [u.namep] ; pathname address (virtual)
11403 0001108E E85552FFFF              <1>              call   get_physical_addr ; get physical address
11404 00011093 7204                    <1>       jc     short tr_addr_nm_err
11405                                  <1>       ; 18/10/2015
11406                                  <1>       ; eax = physical address
11407                                  <1>       ; cx = remain byte count in page (1-4096)
11408                                  <1>              ; 12/10/2015 (cx = [u.pncount])
11409 00011095 89C6                    <1>       mov    esi, eax ; 12/10/2015 (esi=[u.pnbase])
11410 00011097 58                      <1>       pop    eax
11411 00011098 C3                      <1>       retn
11412                                  <1>
11413                                  <1> tr_addr_nm_err:
11414 00011099 A3[C8030300]            <1>       mov    [u.error], eax
11415                                  <1>       ;pop   eax
11416 0001109E E939C8FFFF              <1>       jmp    error
11417                                  <1>
11418                                  <1> trans_addr_nmk:
11419                                  <1>       ; 12/10/2015
11420                                  <1>       ; 02/07/2015
11421 000110A3 8B35[7C030300]          <1>       mov    esi, [u.namep]  ; [u.pnbase]
11422 000110A9 66B90010                <1>       mov    cx, PAGE_SIZE ; 4096 ; [u.pncount]
11423 000110AD C3                      <1>       retn
11424                                  <1>
11425                                  <1> sysbreak:
11426                                  <1>       ; 18/10/2015
11427                                  <1>       ; 07/10/2015
11428                                  <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
11429                                  <1>       ; 20/06/2013 - 24/03/2014 (Retro UNIX 8086 v1)
11430                                  <1>       ;
11431                                  <1>       ; 'sysbreak' sets the programs break points.
11432                                  <1>       ; It checks the current break point (u.break) to see if it is
11433                                  <1>       ; between "core" and the stack (sp). If it is, it is made an
11434                                  <1>       ; even address (if it was odd) and the area between u.break
11435                                  <1>       ; and the stack is cleared. The new breakpoint is then put
11436                                  <1>       ; in u.break and control is passed to 'sysret'.
11437                                  <1>       ;
11438                                  <1>       ; Calling sequence:
11439                                  <1>       ;     sysbreak; addr
11440                                  <1>       ; Arguments: -
11441                                  <1>       ;
11442                                  <1>       ; Inputs: u.break - current breakpoint
11443                                  <1>       ; Outputs: u.break - new breakpoint
11444                                  <1>       ;     area between old u.break and the stack (sp) is cleared.
11445                                  <1>       ; .........................................................
11446                                  <1>       ;
11447                                  <1>       ; Retro UNIX 8086 v1 modification:
11448                                  <1>       ;     The user/application program puts breakpoint address
11449                                  <1>       ;      in BX register as 'sysbreak' system call argument.
11450                                  <1>       ;     (argument transfer method 1)
11451                                  <1>       ;
11452                                  <1>       ;  NOTE: Beginning of core is 0 in Retro UNIX 8086 v1 !
11453                                  <1>       ;     ((!'sysbreak' is not needed in Retro UNIX 8086 v1!))
11454                                  <1>       ;  NOTE:
11455                                  <1>       ;     'sysbreak' clears extended part (beyond of previous
11456                                  <1>       ;     'u.break' address) of user's memory for original unix's
11457                                  <1>       ;     'bss' compatibility with Retro UNIX 8086 v1 (19/11/2013)
11458                                  <1>
11459                                  <1>              ; mov u.break,r1 / move users break point to r1
11460                                  <1>              ; cmp r1,$core / is it the same or lower than core?
11461                                  <1>              ; blos 1f / yes, 1f
11462                                  <1>       ; 23/06/2015
11463 000110AE 8B2D[90030300]          <1>       mov    ebp, [u.break] ; virtual address (offset)
11464                                  <1>       ;and   ebp, ebp
11465                                  <1>       ;jz    short sysbreak_3
11466                                  <1>       ; Retro UNIX 386 v1 NOTE: u.break points to virtual address !!!
11467                                  <1>       ; (Even break point address is not needed for Retro UNIX 386 v1)
11468 000110B4 8B15[5C030300]          <1>       mov    edx, [u.sp] ; kernel stack at the beginning of sys call
11469 000110BA 83C20C                  <1>       add    edx, 12 ; EIP -4-> CS -4-> EFLAGS -4-> ESP (user)
11470                                  <1>       ; 07/10/2015
11471 000110BD 891D[90030300]          <1>       mov    [u.break], ebx ; virtual address !!!
11472                                  <1>       ;
11473 000110C3 3B1A                    <1>       cmp    ebx, [edx] ; compare new break point with
11474                                  <1>                         ; with top of user's stack (virtual!)
11475 000110C5 7323                    <1>       jnb    short sysbreak_3
11476                                  <1>              ; cmp r1,sp / is it the same or higher
11477                                  <1>                       ; / than the stack?
11478                                  <1>              ; bhis 1f / yes, 1f
11479 000110C7 89DE                    <1>       mov    esi, ebx
11480 000110C9 29EE                    <1>       sub    esi, ebp ; new break point - old break point
11481 000110CB 761D                    <1>       jna    short sysbreak_3
11482                                  <1>       ;push  ebx
11483                                  <1> sysbreak_1:
11484 000110CD 89EB                    <1>       mov    ebx, ebp
11485 000110CF E81452FFFF              <1>       call   get_physical_addr ; get physical address
11486 000110D4 72C3                    <1>       jc     tr_addr_nm_err
11487                                  <1>       ; 18/10/2015
11488 000110D6 89C7                    <1>       mov    edi, eax
11489 000110D8 29C0                    <1>       sub    eax, eax ; 0
11490                                  <1>              ; ECX = remain byte count in page (1-4096)
11491 000110DA 39CE                    <1>       cmp    esi, ecx
11492 000110DC 7302                    <1>       jnb    short sysbreak_2
11493 000110DE 89F1                    <1>       mov    ecx, esi
11494                                  <1> sysbreak_2:
11495 000110E0 29CE                    <1>       sub    esi, ecx
11496 000110E2 01CD                    <1>       add    ebp, ecx
```

```
11497 000110E4 F3AA              <1>       rep    stosb
11498 000110E6 09F6              <1>       or     esi, esi
11499 000110E8 75E3              <1>       jnz    short sysbreak_1
11500                            <1>       ;
11501                            <1>              ; bit $1,r1 / is it an odd address
11502                            <1>              ; beq 2f / no, its even
11503                            <1>              ; clrb (r1)+ / yes, make it even
11504                            <1>       ; 2: / clear area between the break point and the stack
11505                            <1>              ; cmp r1,sp / is it higher or same than the stack
11506                            <1>              ; bhis 1f / yes, quit
11507                            <1>              ; clr (r1)+ / clear word
11508                            <1>              ; br 2b / go back
11509                            <1>       ;pop   ebx
11510                            <1> sysbreak_3: ; 1:
11511                            <1>       ;mov   [u.break], ebx ; virtual address !!!
11512                            <1>              ; jsr r0,arg; u.break / put the "address"
11513                            <1>                    ; / in u.break (set new break point)
11514                            <1>              ; br sysret4 / br sysret
11515 000110EA E90DC8FFFF        <1>       jmp    sysret
11516                            <1>
11517                            <1> sysseek: ; / moves read write pointer in an fsp entry
11518                            <1>       ; 06/11/2016 - TRDOS 386 (TRDOS v2.0)
11519                            <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
11520                            <1>       ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
11521                            <1>       ;
11522                            <1>       ; 'sysseek' changes the r/w pointer of (3rd word of in an
11523                            <1>       ; fsp entry) of an open file whose file descriptor is in u.r0.
11524                            <1>       ; The file descriptor refers to a file open for reading or
11525                            <1>       ; writing. The read (or write) pointer is set as follows:
11526                            <1>       ;     * if 'ptrname' is 0, the pointer is set to offset.
11527                            <1>       ;     * if 'ptrname' is 1, the pointer is set to its
11528                            <1>       ;       current location plus offset.
11529                            <1>       ;     * if 'ptrname' is 2, the pointer is set to the
11530                            <1>       ;       size of file plus offset.
11531                            <1>       ; The error bit (e-bit) is set for an undefined descriptor.
11532                            <1>       ;
11533                            <1>       ; Calling sequence:
11534                            <1>       ;     sysseek; offset; ptrname
11535                            <1>       ; Arguments:
11536                            <1>       ;     offset - number of bytes desired to move
11537                            <1>       ;              the r/w pointer
11538                            <1>       ;     ptrname - a switch indicated above
11539                            <1>       ;
11540                            <1>       ; Inputs: r0 - file descriptor
11541                            <1>       ; Outputs: -
11542                            <1>       ; ..........................................................
11543                            <1>       ;
11544                            <1>       ; Retro UNIX 8086 v1 modification:
11545                            <1>       ;       'sysseek' system call has three arguments; so,
11546                            <1>       ;     * 1st argument, file descriptor is in BX (BL) register
11547                            <1>       ;     * 2nd argument, offset is in CX register
11548                            <1>       ;     * 3rd argument, ptrname/switch is in DX (DL) register
11549                            <1>
11550 000110EF E821000000        <1>       call   seektell
11551                            <1>       ; EAX = Current R/W pointer of the file
11552                            <1>       ; EBX = [u.fofp]
11553                            <1>       ; [u.base] = offset (ECX input)
11554                            <1>
11555 000110F4 0305[84030300]    <1>       add    eax, [u.base]
11556 000110FA 8903              <1>       mov    [ebx], eax
11557 000110FC E9FBC7FFFF        <1>       jmp    sysret
11558                            <1>
11559                            <1> systell: ; / get the r/w pointer
11560                            <1>       ; 06/11/2016 - TRDOS 386 (TRDOS v2.0) - temporary !-
11561                            <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
11562                            <1>       ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
11563                            <1>       ;
11564                            <1>       ; Retro UNIX 8086 v1 modification:
11565                            <1>       ; ! 'systell' does not work in original UNIX v1,
11566                            <1>       ;          it returns with error !
11567                            <1>       ; Inputs: r0 - file descriptor
11568                            <1>       ; Outputs: r0 - file r/w pointer
11569                            <1>
11570                            <1>       ;xor   ecx, ecx ; 0
11571 00011101 BA01000000        <1>       mov    edx, 1 ; 05/08/2013
11572                            <1>       ;call  seektell
11573 00011106 E810000000        <1>       call   seektell0 ; 05/08/2013
11574                            <1>       ;; 06/11/2016
11575                            <1>       ;; mov eax, [ebx]
11576 0001110B A3[64030300]      <1>       mov    [u.r0], eax
11577 00011110 E9E7C7FFFF        <1>       jmp    sysret
11578                            <1>
11579                            <1> ; Original unix v1 'systell' system call:
11580                            <1>             ; jsr r0,seektell
11581                            <1>             ; br error4
11582                            <1>
11583                            <1> seektell:
11584                            <1>       ; 06/11/2016 - TRDOS 386 (TRDOS v2.0)
11585                            <1>       ; 03/01/2016
11586                            <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
11587                            <1>       ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
11588                            <1>       ;
11589                            <1>       ; 'seektell' puts the arguments from sysseek and systell
11590                            <1>       ; call in u.base and u.count. It then gets the i-number of
11591                            <1>       ; the file from the file descriptor in u.r0 and by calling
11592                            <1>       ; getf. The i-node is brought into core and then u.count
11593                            <1>       ; is checked to see it is a 0, 1, or 2.
11594                            <1>       ; If it is 0 - u.count stays the same
11595                            <1>       ;          1 - u.count = offset (u.fofp)
11596                            <1>       ;        2 - u.count = i.size (size of file)
11597                            <1>       ;
11598                            <1>       ; !! Retro UNIX 8086 v1 modification:
11599                            <1>       ;     Argument 1, file descriptor is in BX;
11600                            <1>       ;     Argument 2, offset is in CX;
11601                            <1>       ;     Argument 3, ptrname/switch is in DX register.
```

```
11602                                   <1>       ;
11603                                   <1>       ; ((Return -> eax = base for offset (position= base+offset))
11604                                   <1>       ;
11605 00011115 890D[84030300]            <1>       mov    [u.base], ecx ; offset
11606                                   <1> seektell0:
11607 0001111B 8915[88030300]            <1>       mov    [u.count], edx
11608                                   <1>       ; EBX = file descriptor (file number)
11609 00011121 E8F3FEFFFF                <1>       call   getf1
11610                                   <1>       ; EAX = First cluster of the file
11611                                   <1>       ; EBX = File number (Open file number)
11612                                   <1>       ; [u.fofp] = Pointer to File pointer
11613                                   <1>       ; [i.size] = File size
11614                                   <1>
11615 00011126 09C0                      <1>       or     eax, eax
11616 00011128 7514                      <1>       jnz    short seektell1
11617                                   <1>
11618 0001112A B80A000000                <1>       mov    eax, ERR_FILE_NOT_OPEN
11619 0001112F A3[64030300]              <1>       mov    [u.r0], eax
11620 00011134 A3[C8030300]              <1>       mov    dword [u.error], eax ; 'file not open !'
11621 00011139 E99EC7FFFF                <1>       jmp    error
11622                                   <1>
11623                                   <1> seektell1:
11624 0001113E 8B1D[74030300]            <1>         mov     ebx, [u.fofp]
11625 00011144 803D[88030300]01          <1>       cmp    byte [u.count], 1
11626 0001114B 7705                      <1>       ja     short seektell2
11627 0001114D 7409                      <1>       je     short seektell3
11628 0001114F 31C0                      <1>       xor    eax, eax
11629 00011151 C3                        <1>       retn
11630                                   <1>
11631                                   <1> seektell2:
11632 00011152 A1[55040300]              <1>         mov        eax, [i.size]
11633 00011157 C3                        <1>       retn
11634                                   <1>
11635                                   <1> seektell3:
11636 00011158 8B03                      <1>       mov    eax, [ebx]
11637 0001115A C3                        <1>       retn
11638                                   <1>
11639                                   <1> sysintr: ; / set interrupt handling
11640                                   <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
11641                                   <1>       ; 07/07/2013 (Retro UNIX 8086 v1)
11642                                   <1>       ;
11643                                   <1>       ; 'sysintr' sets the interrupt handling value. It puts
11644                                   <1>       ; argument of its call in u.intr then branches into 'sysquit'
11645                                   <1>       ; routine. u.tty is checked if to see if a control tty exists.
11646                                   <1>       ; If one does the interrupt character in the tty buffer is
11647                                   <1>       ; cleared and 'sysret'is called. If one does not exits
11648                                   <1>       ; 'sysret' is just called.
11649                                   <1>       ;
11650                                   <1>       ; Calling sequence:
11651                                   <1>       ;     sysintr; arg
11652                                   <1>       ; Argument:
11653                                   <1>       ;     arg - if 0, interrupts (ASCII DELETE) are ignored.
11654                                   <1>       ;         - if 1, intterupts cause their normal result
11655                                   <1>       ;            i.e force an exit.
11656                                   <1>       ;         - if arg is a location within the program,
11657                                   <1>       ;           control is passed to that location when
11658                                   <1>       ;           an interrupt occurs.
11659                                   <1>       ; Inputs: -
11660                                   <1>       ; Outputs: -
11661                                   <1>       ; ..........................................................
11662                                   <1>       ;
11663                                   <1>       ; Retro UNIX 8086 v1 modification:
11664                                   <1>       ;      'sysintr' system call sets u.intr to value of BX
11665                                   <1>       ;     then branches into sysquit.
11666                                   <1>       ;
11667 0001115B 66891D[AA030300]          <1>       mov    [u.intr], bx
11668                                   <1>             ; jsr r0,arg; u.intr / put the argument in u.intr
11669                                   <1>             ; br 1f / go into quit routine
11670 00011162 E995C7FFFF                <1>       jmp    sysret
11671                                   <1>
11672                                   <1> sysquit:
11673                                   <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
11674                                   <1>       ; 07/07/2013 (Retro UNIX 8086 v1)
11675                                   <1>       ;
11676                                   <1>       ; 'sysquit' turns off the quit signal. it puts the argument of
11677                                   <1>       ; the call in u.quit. u.tty is checked if to see if a control
11678                                   <1>       ; tty exists. If one does the interrupt character in the tty
11679                                   <1>       ; buffer is cleared and 'sysret'is called. If one does not exits
11680                                   <1>       ; 'sysret' is just called.
11681                                   <1>       ;
11682                                   <1>       ; Calling sequence:
11683                                   <1>       ;     sysquit; arg
11684                                   <1>       ; Argument:
11685                                   <1>       ;     arg - if 0, this call diables quit signals from the
11686                                   <1>       ;           typewriter (ASCII FS)
11687                                   <1>       ;         - if 1, quits are re-enabled and cause execution to
11688                                   <1>       ;           cease and a core image to be produced.
11689                                   <1>       ;            i.e force an exit.
11690                                   <1>       ;         - if arg is an addres in the program,
11691                                   <1>       ;           a quit causes control to sent to that
11692                                   <1>       ;           location.
11693                                   <1>       ; Inputs: -
11694                                   <1>       ; Outputs: -
11695                                   <1>       ; ..........................................................
11696                                   <1>       ;
11697                                   <1>       ; Retro UNIX 8086 v1 modification:
11698                                   <1>       ;      'sysquit' system call sets u.quit to value of BX
11699                                   <1>       ;     then branches into 'sysret'.
11700                                   <1>       ;
11701 00011167 66891D[AC030300]          <1>       mov    [u.quit], bx
11702 0001116E E989C7FFFF                <1>       jmp    sysret
11703                                   <1>             ; jsr r0,arg; u.quit / put argument in u.quit
11704                                   <1>       ;1:
11705                                   <1>             ; mov u.ttyp,r1 / move pointer to control tty buffer
11706                                   <1>                         ; / to r1
```

```
11707                                  <1>             ; beq sysret4 / return to user
11708                                  <1>             ; clrb 6(r1) / clear the interrupt character
11709                                  <1>                     ; / in the tty buffer
11710                                  <1>             ; br sysret4 / return to user
11711                                  <1>
11712                                  <1> anyi:
11713                                  <1>       ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
11714                                  <1>       ; Major Modification!
11715                                  <1>       ; TRDOS 386 does not permit to delete a file while it is open
11716                                  <1>       ; The role of 'anyi' procedure has beeen changed to ensure that.
11717                                  <1>       ;
11718                                  <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
11719                                  <1>       ; 25/04/2013 (Retro UNIX 8086 v1)
11720                                  <1>       ;
11721                                  <1>       ; 'anyi' is called if a file deleted while open.
11722                                  <1>       ; "anyi" checks to see if someone else has opened this file.
11723                                  <1>       ;
11724                                  <1>       ; INPUTS ->
11725                                  <1>       ;    r1 - contains an i-number
11726                                  <1>       ;    fsp - start of table containing open files
11727                                  <1>       ;
11728                                  <1>       ; OUTPUTS ->
11729                                  <1>       ;    "deleted" flag set in fsp entry of another occurrence of
11730                                  <1>       ;        this file and r2 points 1st word of this fsp entry.
11731                                  <1>       ;    if file not found - bit in i-node map is cleared
11732                                  <1>       ;                    (i-node is freed)
11733                                  <1>       ;              all blocks related to i-node are freed
11734                                  <1>       ;             all flags in i-node are cleared
11735                                  <1>       ; ((AX = R1)) input
11736                                  <1>       ;
11737                                  <1>       ;    (Retro UNIX Prototype : 02/12/2012, UNIXCOPY.ASM)
11738                                  <1>         ;    ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
11739                                  <1>       ;
11740                                  <1>       ; / r1 contains an i-number
11741                                  <1>
11742                                  <1>       ; TRDOS 386 (06/10/2016)
11743                                  <1>       ;
11744                                  <1>       ; INPUT:
11745                                  <1>       ;      EAX = First Cluster
11746                                  <1>       ;       DL = Logical DOS Drive Number
11747                                  <1>       ;
11748                                  <1>       ; OUTPUT:
11749                                  <1>       ;      CF = 1 -> EBX = File Handle/Number/Index
11750                                  <1>       ;      CF = 0 -> EBX = 0
11751                                  <1>       ;
11752                                  <1>       ; Modified Registers: EBX
11753                                  <1>
11754 00011173 31DB                    <1>       xor    ebx, ebx
11755                                  <1> anyi_0:
11756 00011175 80BB[E6990100]00        <1>       cmp    byte [ebx+OF_MODE], 0 ; 0 = empty entry
11757 0001117C 770A                    <1>       ja     short anyi_2 ; 1 (r), 2 (w) or 3 (r&w)
11758                                  <1> anyi_1:
11759 0001117E FEC3                    <1>       inc    bl
11760 00011180 80FB0A                  <1>       cmp    bl, OPENFILES ; max. count of open files
11761 00011183 72F0                    <1>       jb     short anyi_0
11762 00011185 31C0                    <1>       xor    eax, eax
11763 00011187 C3                      <1>       retn
11764                                  <1> anyi_2:
11765 00011188 3A93[DC990100]          <1>       cmp    dl, [ebx+OF_DRIVE]
11766 0001118E 75EE                    <1>       jne    short anyi_1
11767 00011190 66C1E302                <1>       shl    bx, 2 ; *4 (dword offset)
11768 00011194 3B83[B4990100]          <1>       cmp    eax, [ebx+OF_FCLUSTER]
11769 0001119A 7406                    <1>       je     short anyi_3
11770 0001119C 66C1EB02                <1>       shr    bx, 2 ; /4 (byte offset)
11771 000111A0 EBDC                    <1>       jmp    short anyi_1
11772                                  <1> anyi_3:
11773 000111A2 66C1EB02                <1>       shr    bx, 2 ; /4 (bytes offset) (index)
11774 000111A6 F9                      <1>       stc
11775 000111A7 C3                      <1>       retn
11776                                  <1>
11777                                  <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS9.INC
11778                                  <1> ; Last Modification: 09/12/2015
11779                                  <1>
11780                                  <1> syssleep:
11781                                  <1>       ; 29/06/2015 - (Retro UNIX 386 v1)
11782                                  <1>       ; 11/06/2014 - (Retro UNIX 8086 v1)
11783                                  <1>       ;
11784                                  <1>       ; Retro UNIX 8086 v1 feature only
11785                                  <1>       ; (INPUT -> none)
11786                                  <1>       ;
11787 000111A8 0FB61D[B3030300]       <1>       movzx  ebx, byte [u.uno] ; process number
11788 000111AF 8AA3[7F000300]          <1>       mov    ah, [ebx+p.ttyc-1] ; current/console tty
11789 000111B5 E881190000              <1>       call   sleep
11790 000111BA E93DC7FFFF              <1>       jmp    sysret
11791                                  <1>
11792                                  <1> _vp_clr:
11793                                  <1>       ; Reset/Clear Video Page
11794                                  <1>       ;
11795                                  <1>       ; 30/06/2015 - (Retro UNIX 386 v1)
11796                                  <1>       ; 21/05/2013 - 30/10/2013(Retro UNIX 8086 v1) (U0.ASM)
11797                                  <1>       ;
11798                                  <1>       ; Retro UNIX 8086 v1 feature only !
11799                                  <1>       ;
11800                                  <1>       ; INPUTS ->
11801                                  <1>       ;   BH = video page number
11802                                  <1>       ;
11803                                  <1>       ; OUTPUT ->
11804                                  <1>       ;   none
11805                                  <1>       ; ((Modified registers: eAX, BH, eCX, eDX, eSI, eDI))
11806                                  <1>       ;
11807                                  <1>       ; 04/12/2013
11808 000111BF 28C0                    <1>       sub    al, al
11809                                  <1>       ; al = 0 (clear video page)
11810                                  <1>       ; bh = video page ; 13/05/2016
11811 000111C1 B407                    <1>       mov    ah, 07h
```

```
11812                                  <1>       ; ah = 7 (attribute/color)
11813 000111C3 6631C9                  <1>       xor    cx, cx ; 0, left upper column (cl) & row (cl)
11814 000111C6 66BA4F18                <1>       mov    dx, 184Fh ; right lower column & row (dl=24, dh=79)
11815 000111CA E86E0EFFFF              <1>       call   _scroll_up
11816                                  <1>       ; bh = video page
11817 000111CF 6631D2                  <1>       xor    dx, dx ; 0 (cursor position)
11818 000111D2 E9C011FFFF              <1>       jmp    _set_cpos
11819                                  <1>
11820                                  <1> sysmsg:
11821                                  <1>       ; 07/12/2020
11822                                  <1>       ; 05/12/2020
11823                                  <1>       ; 13/05/2016
11824                                  <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
11825                                  <1>       ; 01/07/2015 - 11/11/2015 (Retro UNIX 386 v1)
11826                                  <1>       ; Print user-application message on user's console tty
11827                                  <1>       ;
11828                                  <1>       ; Input -> EBX = Message address
11829                                  <1>       ;         ECX = Message length (max. 255)
11830                                  <1>       ;         DL = Color (IBM PC Rombios color attributes)
11831                                  <1>       ;
11832 000111D7 81F9FF000000            <1>       cmp    ecx, MAX_MSG_LEN ; 255
11833 000111DD 0F8719C7FFFF            <1>       ja     sysret ; nothing to do with big message size
11834 000111E3 08C9                    <1>       or     cl, cl
11835 000111E5 0F8411C7FFFF            <1>       jz     sysret
11836 000111EB 20D2                    <1>       and    dl, dl
11837 000111ED 7502                    <1>       jnz    short sysmsg0
11838 000111EF B207                    <1>       mov    dl, 07h ; default color
11839                                  <1>              ; (black background, light gray character)
11840                                  <1> sysmsg0:
11841 000111F1 891D[84030300]          <1>       mov    [u.base], ebx
11842 000111F7 8815[EF890100]          <1>       mov    [ccolor], dl ; color attributes
11843 000111FD 89E5                    <1>       mov    ebp, esp
11844 000111FF 31DB                    <1>       xor    ebx, ebx ; 0
11845 00011201 891D[8C030300]          <1>       mov    [u.nread], ebx ; 0
11846                                  <1>       ;
11847 00011207 381D[C6030300]          <1>       cmp    [u.kcall], bl ; 0
11848 0001120D 776F                    <1>       ja     short sysmsgk ; Temporary (01/07/2015)
11849                                  <1>       ;
11850 0001120F 890D[88030300]          <1>       mov    [u.count], ecx
11851                                  <1>       ;inc   ecx ; + 00h ; ASCIIZ
11852                                  <1>       ;
11853                                  <1>       ; 07/12/2020
11854                                  <1>       ;add   ecx, 3
11855 00011215 6683C103                <1>       add    cx, 3
11856 00011219 80E1FC                  <1>       and    cl, ~3  ; not 3
11857                                  <1>       ;
11858 0001121C 29CC                    <1>       sub    esp, ecx
11859 0001121E 89E7                    <1>       mov    edi, esp
11860 00011220 89E6                    <1>       mov    esi, esp
11861 00011222 66891D[C4030300]        <1>       mov    [u.pcount], bx ; reset page (phy. addr.) counter
11862                                  <1>       ; 11/11/2015
11863 00011229 8A25[94030300]          <1>       mov    ah, [u.ttyp] ; recent open tty
11864                                  <1>       ; 0 = none
11865 0001122F FECC                    <1>       dec    ah
11866 00011231 790C                    <1>       jns    short sysmsg1
11867 00011233 8A1D[B3030300]          <1>       mov    bl, [u.uno] ; process number
11868 00011239 8AA3[7F000300]          <1>       mov    ah, [ebx+p.ttyc-1] ; user's (process's) console tty
11869                                  <1> sysmsg1:
11870 0001123F 8825[96030300]          <1>       mov    [u.ttyn], ah
11871                                  <1> sysmsg2:
11872 00011245 E848080000              <1>       call   cpass
11873 0001124A 7416                    <1>       jz     short sysmsg5
11874 0001124C AA                      <1>       stosb
11875 0001124D 20C0                    <1>       and    al, al
11876 0001124F 75F4                    <1>       jnz    short sysmsg2
11877                                  <1> sysmsg3:
11878 00011251 80FC07                  <1>       cmp    ah, 7 ; tty number
11879 00011254 7711                    <1>       ja     short sysmsg6 ; serial port
11880 00011256 E83E000000              <1>       call   print_cmsg ; 05/12/2020
11881                                  <1> sysmsg4:
11882 0001125B 89EC                    <1>       mov    esp, ebp
11883 0001125D E99AC6FFFF              <1>       jmp    sysret
11884                                  <1> sysmsg5:
11885 00011262 C60700                  <1>       mov    byte [edi], 0
11886 00011265 EBEA                    <1>       jmp    short sysmsg3
11887                                  <1> sysmsg6:
11888 00011267 8A06                    <1>       mov    al, [esi]
11889 00011269 E8CD180000              <1>       call   sndc
11890 0001126E 72EB                    <1>       jc     short sysmsg4
11891 00011270 803E00                  <1>       cmp    byte [esi], 0  ; 0 is stop character
11892 00011273 76E6                    <1>       jna    short sysmsg4
11893 00011275 46                      <1>       inc    esi
11894 00011276 8A25[96030300]          <1>       mov    ah, [u.ttyn]
11895 0001127C EBE9                    <1>       jmp    short sysmsg6
11896                                  <1>
11897                                  <1> sysmsgk: ; Temporary (01/07/2015)
11898                                  <1>       ; The message has been sent by Kernel (ASCIIZ string)
11899                                  <1>       ; (ECX -character count- will not be considered)
11900 0001127E 8B35[84030300]          <1>       mov    esi, [u.base]
11901 00011284 8A25[EE890100]          <1>       mov    ah, [ptty] ; present/current screen (video page)
11902 0001128A 8825[96030300]          <1>       mov    [u.ttyn], ah
11903 00011290 C605[C6030300]00        <1>       mov    byte [u.kcall], 0
11904 00011297 EBB8                    <1>       jmp    short sysmsg3
11905                                  <1>
11906                                  <1> print_cmsg:
11907                                  <1>       ; 08/12/2020
11908                                  <1>       ; 07/12/2020
11909                                  <1>       ; 05/12/2020
11910                                  <1>       ; 18/11/2017
11911                                  <1>       ; 13/05/2016 - TRDOS 386 (TRDOS v2.0)
11912                                  <1>       ; 01/07/2015 (Retro UNIX 386 v1)
11913                                  <1>       ;
11914                                  <1>       ; print message (on user's console tty)
11915                                  <1>       ;      with requested color
11916                                  <1>       ;
```

```
11917                                  <1>       ; INPUTS:
11918                                  <1>       ;      esi = message address
11919                                  <1>       ;      [u.ttyn] = tty number (0 to 7)
11920                                  <1>       ;      [ccolor] = color attributes (IBM PC BIOS colors)
11921                                  <1>       ;
11922                                  <1>       ; Modified registers: eax, ebx, ecx, edx, esi, edi
11923                                  <1>       ; (ebp must be preserved)
11924                                  <1>
11925                                  <1>       ;mov   bh, ah
11926 00011299 8A3D[96030300]         <1>       mov    bh, [u.ttyn]
11927 0001129F 8A1D[EF890100]         <1>       mov    bl, [ccolor] ; * ; 05/12/2020
11928                                  <1>
11929                                  <1>       ; 05/12/2020
11930 000112A5 803D[1C120300]00       <1>       cmp    byte [pmi32], 0 ; is vbios's 32 bit pmi enabled ?
11931 000112AC 772E                   <1>       ja     short pcmsg5 ; yes
11932                                  <1> pcmsg1:
11933                                  <1>       ; 08/12/2020
11934 000112AE 8A1D[EF890100]         <1>       mov    bl, [ccolor] ; * (video.s 'u11'&'beep' change BL)
11935                                  <1>
11936 000112B4 AC                     <1>       lodsb
11937 000112B5 20C0                   <1>       and    al, al  ; 0
11938 000112B7 743A                   <1>       jz     short pcmsg2
11939                                  <1> pcmsg7:
11940 000112B9 56                     <1>       push   esi
11941                                  <1>       ;mov   bl, [ccolor] ; * (video.s 'u11'&'beep' change BL)
11942                                  <1>       ; 05/12/2020
11943                                  <1>       ;;mov  bh, [u.ttyn]
11944                                  <1>       ;call  _write_tty
11945                                  <1>       ;pop   esi
11946                                  <1>       ;jmp   short pcmsg1
11947                                  <1> ;pcmsg2:
11948                                  <1>       ;retn
11949                                  <1>
11950                                  <1>       ; 07/12/2020
11951 000112BA 803D[CA6F0000]03       <1>       cmp     byte [CRT_MODE], 3
11952 000112C1 7708                   <1>       ja     short pcmsg4
11953                                  <1> pcmsg3:
11954 000112C3 E84810FFFF             <1>       call   _write_tty_m3
11955 000112C8 5E                     <1>       pop    esi
11956 000112C9 EBE3                   <1>       jmp    short pcmsg1
11957                                  <1> pcmsg4:
11958 000112CB 803D[CA6F0000]07       <1>         cmp     byte [CRT_MODE], 7
11959 000112D2 76EF                   <1>       jna    short pcmsg3
11960 000112D4 E85B1DFFFF             <1>       call   vga_write_teletype
11961 000112D9 5E                     <1>       pop    esi
11962 000112DA EBD2                   <1>       jmp    short pcmsg1
11963                                  <1> pcmsg5:
11964                                  <1>       ; 07/12/2020
11965 000112DC 803D[CA6F0000]07       <1>         cmp     byte [CRT_MODE], 7
11966 000112E3 76C9                   <1>       jna    short pcmsg1
11967                                  <1>
11968                                  <1>       ; 05/12/2020
11969                                  <1>       ; writing message by using
11970                                  <1>       ; VESA VBE3 video bios protected mode interface
11971                                  <1>
11972 000112E5 B40E                   <1>       mov    ah, 0Eh
11973                                  <1> pcmsg6:
11974 000112E7 AC                     <1>       lodsb
11975 000112E8 20C0                   <1>       and    al, al  ; 0
11976 000112EA 7407                   <1>       jz     short pcmsg2
11977                                  <1>       ; bh = video page
11978                                  <1>       ; ah = 0Eh
11979                                  <1>       ; al = character
11980                                  <1>       ; bl = color
11981 000112EC E81907FFFF             <1>       call   int10h_32bit_pmi
11982 000112F1 EBF4                   <1>       jmp    short pcmsg6
11983                                  <1> pcmsg2:
11984 000112F3 C3                     <1>       retn
11985                                  <1>
11986                                  <1> sysgeterr:
11987                                  <1>       ; 09/12/2015
11988                                  <1>       ; 21/09/2015 - (Retro UNIX 386 v1 feature only!)
11989                                  <1>       ; Get last error number or page fault count
11990                                  <1>       ; (for debugging)
11991                                  <1>       ;
11992                                  <1>       ; Input -> EBX = return type
11993                                  <1>       ;         0 = last error code (which is in 'u.error')
11994                                  <1>       ;         FFFFFFFFh = page fault count for running process
11995                                  <1>       ;         FFFFFFFEh = total page fault count
11996                                  <1>       ;         1 .. FFFFFFFDh = undefined
11997                                  <1>       ;
11998                                  <1>       ; Output -> EAX = last error number or page fault count
11999                                  <1>       ;         (depending on EBX input)
12000                                  <1>       ;
12001 000112F4 21DB                   <1>       and    ebx, ebx
12002 000112F6 750B                   <1>       jnz    short glerr_2
12003                                  <1> glerr_0:
12004 000112F8 A1[C8030300]           <1>       mov    eax, [u.error]
12005                                  <1> glerr_1:
12006 000112FD A3[64030300]           <1>       mov    [u.r0], eax
12007 00011302 C3                     <1>       retn
12008                                  <1> glerr_2:
12009 00011303 43                     <1>       inc    ebx ; FFFFFFFFh -> 0, FFFFFFFEh -> FFFFFFFFh
12010 00011304 74FD                   <1>       jz     short glerr_2 ; page fault count for process
12011 00011306 43                     <1>       inc    ebx ; FFFFFFFFh -> 0
12012 00011307 75EF                   <1>       jnz    short glerr_0
12013 00011309 A1[80050300]           <1>       mov    eax, [PF_Count] ; total page fault count
12014 0001130E EBED                   <1>         jmp     short glerr_1
12015                                  <1> glerr_3:
12016 00011310 A1[CC030300]           <1>       mov    eax, [u.pfcount]
12017 00011315 EBE6                   <1>       jmp    short glerr_1
12018                                  <1>
12019                                  <1> load_and_run_file:
12020                                  <1>       ; 18/11/2017
12021                                  <1>       ; 22/01/2017
```

```
12022                                  <1>       ; 04/01/2017, 07/01/2017
12023                                  <1>       ; 24/10/2016
12024                                  <1>       ; 24/04/2016, 02/05/2016, 03/05/2016, 06/05/2016
12025                                  <1>       ; 23/04/2016 (TRDOS 386 = TRDOS v2.0)
12026                                  <1>       ; 23/10/2015 (Retro UNIX 386 v1, 'sysexec')
12027                                  <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
12028                                  <1>       ; 03/06/2013 - 06/12/2013 (Retro UNIX 8086 v1)
12029                                  <1>       ; EAX = First Cluster number
12030                                  <1>       ; EDX = File Size
12031                                  <1>       ; ESI = Argument list address
12032                                  <1>       ; [argc] = argument count
12033                                  <1>       ; [u.nread] = argument list length
12034                                  <1>       ; [esp] = return address to the caller (*)
12035                                  <1>       ;
12036 00011317 8935[4C040300]         <1>       mov   [argv], esi
12037 0001131D 8915[55040300]         <1>       mov   [i.size], edx
12038 00011323 A3[51040300]           <1>       mov   [ii], eax
12039                                  <1>
12040                                  <1>       ;sti   ; 07/01/2017
12041                                  <1>       ;mov   eax, [k_page_dir]
12042                                  <1>       ;mov   [u.pgdir], eax
12043 00011328 31C0                   <1>       xor   eax, eax ; clc ; *** ; 04/01/2017
12044                                  <1>       ;mov   [u.r0], eax ; 0 ; 07/01/2017
12045                                  <1>
12046                                  <1>       ; 06/05/2016
12047                                  <1>       ; Set 'sysexit' return order to MainProg
12048                                  <1>       ;
12049 0001132A 58                     <1>       pop   eax ; * 'loc_load_and_run_file_8:' address
12050                                  <1>       ;; 22/01/2017
12051                                  <1>       ;;cli ; 07/01/2017
12052 0001132B 8B25[5C890100]         <1>       mov   esp, [tss.esp0]
12053                                  <1>       ;
12054                                  <1>       ; 'loc_load_run_file_8' address has
12055                                  <1>       ; 'jmp loc_file_rw_restore_retn' instruction
12056                                  <1>       ; 'loc_file_rw_restore_retn:' will return to
12057                                  <1>       ; [mainprog_return_addr]
12058                                  <1>       ; just after 'call command_interpreter'
12059                                  <1>       ;
12060 00011331 68[33750000]           <1>       push  _end_of_mainprog ; we must not return to here !
12061 00011336 FF35[40960100]         <1>       push  dword [mainprog_return_addr]
12062 0001133C 89E5                   <1>       mov   ebp, esp ; **
12063                                  <1>       ;
12064 0001133E 9C                     <1>       pushfd  ; EFLAGS      ; IRETD ; ***
12065 0001133F 6A08                   <1>       push  KCODE ; cs    ; IRETD
12066 00011341 50                     <1>       push  eax ; * (eip) ; IRETD
12067 00011342 8925[5C030300]         <1>       mov   [u.sp], esp
12068                                  <1>       ;mov  byte [u.quant], time_count
12069 00011348 1E                     <1>       push  ds
12070 00011349 06                     <1>       push  es
12071 0001134A 0FA0                   <1>       push  fs
12072 0001134C 0FA8                   <1>       push  gs
12073                                  <1>       ;mov  eax, [u.r0]
12074 0001134E 29C0                   <1>       sub   eax, eax
12075 00011350 60                     <1>       pushad
12076 00011351 68[FCD80000]           <1>       push  sysret
12077                                  <1>       ;push  sysrel1 ; 07/01/2017
12078 00011356 8925[60030300]         <1>       mov   [u.usp], esp
12079                                  <1>       ;
12080 0001135C E845060000             <1>       call  wswap ; Save MainProg (process 1) 'u' structure
12081                                  <1>                   ; and registers for return (from program)
12082 00011361 89EC                   <1>       mov   esp, ebp ; **
12083                                  <1>       ;;22/01/2017
12084                                  <1>       ;;sti ; 07/01/2017
12085 00011363 50                     <1>       push  eax  ; * 'loc_load_and_run_file_8:' address
12086                                  <1>       ;
12087                                  <1>       ;;; 02/05/2016
12088                                  <1>       ;;; Create a new process (parent: MainProg)
12089 00011364 31F6                   <1>       xor   esi, esi
12090                                  <1> cnpm_1: ; search p.stat table for unused process number
12091 00011366 46                     <1>       inc   esi
12092 00011367 80BE[AF000300]00       <1>       cmp   byte [esi+p.stat-1], 0 ; SFREE
12093                                  <1>                          ; is process active, unused, dead
12094 0001136E 760B                   <1>       jna   short cnpm_2 ; it's unused so branch
12095 00011370 6683FE10               <1>       cmp   si, nproc    ; all processes checked
12096 00011374 72F0                   <1>       jb    short cnpm_1    ; no, branch back
12097 00011376 E94062FFFF             <1>       jmp   panic
12098                                  <1> cnpm_2:
12099 0001137B A1[B8030300]           <1>       mov   eax, [u.pgdir] ; page directory of MainProg
12100 00011380 A3[BC030300]           <1>       mov   [u.ppgdir], eax ; parent's page directory
12101 00011385 E84948FFFF             <1>       call  allocate_page
12102 0001138A 0F822B62FFFF           <1>       jc    panic
12103                                  <1>       ; EAX = UPAGE (user structure page) address
12104 00011390 A3[B4030300]           <1>       mov   [u.upage], eax ; memory page for 'user' struct (child)
12105 00011395 89F7                   <1>       mov   edi, esi
12106 00011397 66C1E702               <1>       shl   di, 2
12107 0001139B 8987[BC000300]         <1>       mov   [edi+p.upage-4], eax ; memory page for 'user' struct
12108 000113A1 E8A748FFFF             <1>       call  clear_page ; 03/05/2016
12109                                  <1>       ;movzx eax, byte [p.ttyc] ; console tty (for MainProg)
12110 000113A6 6629C0                 <1>       sub   ax, ax ; 0
12111 000113A9 668986[7F000300]       <1>       mov    [esi+p.ttyc-1], ax ; al - set child's console tty
12112                                  <1>                               ; ah - reset child's wait channel
12113 000113B0 89F0                   <1>       mov   eax, esi
12114 000113B2 A2[B3030300]           <1>       mov   [u.uno], al ; child process number
12115 000113B7 FE86[AF000300]         <1>         inc    byte [esi+p.stat-1] ; 1, SRUN
12116 000113BD 66D1E6                 <1>       shl   si, 1 ; multiply si by 2 to get index into p.pid table
12117 000113C0 66FF05[4E030300]       <1>       inc   word [mpid] ; increment m.pid; get a new process name
12118 000113C7 66A1[4E030300]         <1>       mov   ax, [mpid]
12119 000113CD 668986[1E000300]       <1>       mov   [esi+p.pid-2], ax ; put new process name
12120                                  <1>                               ; in child process' name slot
12121                                  <1>       ;mov  ax, [p.pid]  ; get process name of MainProg
12122 000113D4 66B80100               <1>       mov   ax, 1
12123 000113D8 668986[3E000300]       <1>       mov   [esi+p.ppid-2], ax ; put parent process name
12124                                  <1>                                ; in parent process slot for child
12125 000113DF 6648                   <1>       dec   ax ; 0
12126 000113E1 66A3[94030300]         <1>       mov   [u.ttyp], ax ; 0
```

```
12127                                  <1>       ;;;
12128 000113E7 A1[51040300]            <1>       mov    eax,  [ii]
12129                                  <1>       ; Retro UNIX 386 v1, 'sysexec' (u2.s)
12130 000113EC E84A170000              <1>       call   iopen
12131                                  <1>       ; 06/06/2016
12132 000113F1 C605[A9030300]01        <1>       mov    byte [u.pri], 1 ; normal priority
12133                                  <1>       ;
12134 000113F8 EB16                    <1>       jmp    short sysexec_7 ; 02/05/2016
12135                                  <1>
12136                                  <1> sysexec_6:
12137                                  <1>       ; 19/11/2017
12138                                  <1>       ; 18/11/2017
12139                                  <1>       ; 14/11/2017
12140                                  <1>       ; 13/11/2017
12141 000113FA 8925[4C040300]          <1>       mov    [argv], esp ; *!* ; start address of argument list
12142                                  <1>
12143                                  <1>       ; 04/01/2017
12144                                  <1>       ; 24/10/2016
12145                                  <1>       ;;02/05/2016
12146                                  <1>       ; 23/04/2016 (TRDOS 386)
12147                                  <1>       ; 18/10/2015 ('sysexec_6')
12148                                  <1>       ; 23/06/2015
12149 00011400 A1[B8030300]            <1>       mov    eax, [u.pgdir] ; physical address of page directory
12150                                  <1>       ;cmp   eax, [k_page_dir] ; TRDOS MainProg ?
12151                                  <1>       ;je    short sysexec_7
12152                                  <1>       ; 19/11/2017
12153 00011405 8B1D[BC030300]          <1>       mov    ebx, [u.ppgdir] ; phy addr of the parent's page dir
12154 0001140B E8FC48FFFF              <1>       call   deallocate_page_dir
12155                                  <1> sysexec_7:
12156 00011410 E82C48FFFF              <1>       call   make_page_dir
12157 00011415 0F82A061FFFF            <1>       jc     panic  ; allocation error
12158                                  <1>                     ; after a deallocation would be nonsence !?
12159                                  <1>       ; 24/07/2015
12160                                  <1>       ; map kernel pages (1st 4MB) to PDE 0
12161                                  <1>       ;     of the user's page directory
12162                                  <1>       ;     (It is needed for interrupts!)
12163                                  <1>       ; 18/10/2015
12164 0001141B 8B15[C0890100]          <1>       mov    edx, [k_page_dir] ; Kernel's page directory
12165 00011421 8B02                    <1>       mov    eax, [edx] ; physical address of
12166                                  <1>                       ; kernel's first page table (1st 4 MB)
12167                                  <1>                       ; (PDE 0 of kernel's page directory)
12168 00011423 8B15[B8030300]          <1>       mov    edx, [u.pgdir]
12169 00011429 8902                    <1>       mov    [edx], eax ; PDE 0 (1st 4MB)
12170                                  <1>       ;
12171                                  <1>       ; 20/07/2015
12172 0001142B BB00004000              <1>       mov    ebx, CORE ; start address = 0 (virtual) + CORE
12173                                  <1>       ; 18/10/2015
12174 00011430 BE[3C040300]            <1>       mov    esi, pcore ; physical start address
12175                                  <1> sysexec_8:
12176 00011435 B907000000              <1>       mov    ecx, PDE_A_USER + PDE_A_WRITE + PDE_A_PRESENT
12177 0001143A E82048FFFF              <1>       call   make_page_table
12178 0001143F 0F827661FFFF            <1>       jc     panic
12179                                  <1>       ;mov   ecx, PTE_A_USER + PTE_A_WRITE + PTE_A_PRESENT
12180 00011445 E82348FFFF              <1>       call   make_page ; make new page, clear and set the pte
12181 0001144A 0F826B61FFFF            <1>       jc     panic
12182                                  <1>       ;
12183 00011450 8906                    <1>       mov    [esi], eax ; 24/06/2015
12184                                  <1>       ; ebx = virtual address (24/07/2015)
12185 00011452 E8BB4DFFFF              <1>       call   add_to_swap_queue
12186                                  <1>       ; 18/10/2015
12187 00011457 81FE[40040300]          <1>       cmp    esi, ecore ; user's stack (last) page ?
12188 0001145D 740C                    <1>       je     short sysexec_9 ; yes
12189 0001145F BE[40040300]            <1>       mov    esi, ecore  ; physical address of the last page
12190                                  <1>       ; 20/07/2015
12191 00011464 BB00F0FFFF              <1>       mov    ebx, (ECORE - PAGE_SIZE) + CORE
12192                                  <1>       ; ebx = virtual end address + segment base address - 4K
12193 00011469 EBCA                    <1>         jmp     short sysexec_8
12194                                  <1> sysexec_9:
12195                                  <1>       ; 19/11/2017
12196                                  <1>       ; 24/04/2016 (TRDOS 386 = TRDOS v2.0)
12197                                  <1>       ; 25/06/2015, 26/08/2015, 18/10/2015
12198                                  <1>       ; move arguments from kernel stack to [ecore]
12199                                  <1>       ; (argument list/line will be copied from kernel stack
12200                                  <1>       ; frame to the last (stack) page of user's core memory)
12201                                  <1>       ; 18/10/2015
12202 0001146B 8B3D[40040300]          <1>       mov    edi, [ecore]
12203 00011471 81C700100000            <1>       add    edi, PAGE_SIZE
12204                                  <1>       ; 19/11/2017
12205 00011477 83EF04                  <1>       sub    edi, 4
12206 0001147A C707000000000           <1>       mov    dword [edi], 0
12207 00011480 89FB                    <1>       mov    ebx, edi
12208                                  <1>       ;
12209 00011482 0FB705[4A040300]        <1>       movzx  eax, word [argc]
12210 00011489 09C0                    <1>       or     eax, eax
12211 0001148B 7445                    <1>       jz     short sysexec_13 ; 19/11/2017
12212                                  <1>       ;jnz   short sysexec_10
12213                                  <1>       ;mov   ebx, edi
12214                                  <1>       ;sub   ebx, 4
12215                                  <1>       ;mov   [ebx], eax ; 0
12216                                  <1>       ;jmp   short sysexec_13
12217                                  <1> sysexec_10:
12218 0001148D 8B0D[8C030300]          <1>       mov    ecx, [u.nread]
12219                                  <1>       ; 13/11/2017
12220                                  <1>       ;mov   esi, TextBuffer ; 'load_and_execute_file'
12221                                  <1>       ;mov   esi, esp     ; 'sysexec'
12222 00011493 8B35[4C040300]          <1>       mov    esi, [argv] ; 24/04/2016 (TRDOS 386  = TRDOS v2.0)
12223                                  <1>       ;sub   edi, ecx ; page end address - argument list length
12224 00011499 29CB                    <1>       sub    ebx, ecx ; 19/11/2017
12225 0001149B 89C2                    <1>       mov    edx, eax
12226 0001149D FEC2                    <1>       inc    dl ; argument count + 1 for argc value
12227 0001149F C0E202                  <1>       shl    dl, 2  ; 4 * (argument count + 1)
12228                                  <1>       ;mov   ebx, edi
12229 000114A2 89DF                    <1>       mov    edi, ebx ; 19//11/2017
12230 000114A4 80E3FC                  <1>       and    bl, 0FCh ; 32 bit (dword) alignment
12231 000114A7 29D3                    <1>       sub    ebx, edx
```

```
12232 000114A9 89FA                <1>       mov    edx, edi
12233 000114AB F3A4                <1>       rep    movsb
12234 000114AD 89D6                <1>       mov    esi, edx
12235 000114AF 89DF                <1>       mov    edi, ebx
12236 000114B1 BA00F0BFFF          <1>       mov    edx, ECORE - PAGE_SIZE ; virtual addr. of the last page
12237 000114B6 2B15[40040300]      <1>       sub    edx, [ecore] ; difference (virtual - physical)
12238 000114BC AB                  <1>       stosd  ; eax = argument count
12239                              <1> sysexec_11:
12240 000114BD 89F0                <1>       mov    eax, esi
12241 000114BF 01D0                <1>       add    eax, edx
12242 000114C1 AB                  <1>       stosd  ; eax = virtual address
12243                              <1>       ;dec   byte [argc]
12244 000114C2 66FF0D[4A040300]    <1>       dec    word [argc] ; 14/11/2017
12245 000114C9 7407                <1>       jz     short sysexec_13
12246                              <1> sysexec_12:
12247 000114CB AC                  <1>       lodsb
12248 000114CC 20C0                <1>       and    al, al
12249 000114CE 75FB                <1>       jnz    short sysexec_12
12250 000114D0 EBEB                <1>       jmp    short sysexec_11
12251                              <1> sysexec_13:
12252                              <1>       ; 24/10/2016
12253                              <1>       ; 24/04/2016 - TRDOS 386 (TRDOS v2.0)
12254                              <1>       ; 23/06/2015 - 19/10/2015 (Retro UNIX 386 v1, 'sysexec_13')
12255                              <1>       ;
12256                              <1>       ; moving arguments to [ecore] is OK here..
12257                              <1>       ;
12258                              <1>       ; ebx = beginning addres of argument list pointers
12259                              <1>             ;      in user's stack
12260 000114D2 2B1D[40040300]      <1>       sub    ebx, [ecore]
12261 000114D8 81C300F0BFFF        <1>       add     ebx, (ECORE - PAGE_SIZE)
12262                              <1>                    ; end of core - 4096 (last page)
12263                              <1>                    ; (virtual address)
12264 000114DE 891D[4C040300]      <1>       mov    [argv], ebx
12265 000114E4 891D[90030300]      <1>       mov    [u.break], ebx ; available user memory
12266                              <1>       ;
12267 000114EA 29C0                <1>       sub    eax, eax
12268 000114EC C705[88030300]2000- <1>       mov    dword [u.count], 32 ; Executable file header size
12268 000114F4 0000                <1>
12269 000114F6 C705[74030300]-     <1>       mov    dword [u.fofp], u.off
12269 000114FC [80030300]          <1>
12270 00011500 A3[80030300]        <1>       mov    [u.off], eax ; 0
12271 00011505 A3[84030300]        <1>       mov    [u.base], eax ; 0, start of user's core (virtual)
12272                              <1>       ; 24/10/2016
12273 0001150A A0[868A0100]        <1>       mov    al, [Current_Drv]
12274 0001150F A2[46030300]        <1>       mov    [cdev], al
12275                              <1>       ;
12276 00011514 A1[51040300]        <1>       mov    eax, [ii] ; Fist Cluster of the Program (PRG) file
12277                              <1>       ; EAX = First cluster of the executable file
12278 00011519 E80A010000          <1>       call   readi
12279                              <1>
12280 0001151E 8B0D[90030300]      <1>       mov    ecx, [u.break] ; top of user's stack (physical addr.)
12281 00011524 890D[88030300]      <1>       mov    [u.count], ecx ; save for overrun check
12282                              <1>       ;
12283 0001152A 8B0D[8C030300]      <1>       mov    ecx, [u.nread]
12284 00011530 890D[90030300]      <1>       mov    [u.break], ecx ; virtual address (offset from start)
12285 00011536 80F920              <1>       cmp    cl, 32
12286 00011539 7540                <1>         jne     short sysexec_15
12287                              <1>       ;:
12288                              <1>       ; Retro UNIX 386 v1 (32 bit) executable file header format
12289 0001153B 8B35[3C040300]      <1>       mov    esi, [pcore] ; start address of user's core memory
12290                              <1>                           ; (phys. start addr. of the exec. file)
12291 00011541 AD                  <1>       lodsd
12292 00011542 663DEB1E            <1>       cmp    ax, 1EEBh ; EBH, 1Eh -> jump to +32
12293 00011546 7533                <1>       jne    short sysexec_15
12294 00011548 AD                  <1>       lodsd
12295 00011549 89C1                <1>       mov    ecx, eax ; text (code) section size
12296 0001154B AD                  <1>       lodsd
12297 0001154C 01C1                <1>       add    ecx, eax ; + data section size (initialized data)
12298 0001154E 89CB                <1>       mov    ebx, ecx
12299 00011550 AD                  <1>       lodsd
12300 00011551 01C3                <1>       add    ebx, eax ; + bss section size (for overrun checking)
12301 00011553 3B1D[88030300]      <1>       cmp    ebx, [u.count]
12302 00011559 7711                <1>       ja     short sysexec_14  ; program overruns stack !
12303                              <1>       ;
12304                              <1>       ; add bss section size to [u.break]
12305 0001155B 0105[90030300]      <1>       add    [u.break], eax
12306                              <1>       ;
12307 00011561 83E920              <1>       sub    ecx, 32  ; header size (already loaded)
12308                              <1>       ;cmp   ecx, [u.count]
12309                              <1>       ;jnb   short sysexec_16
12310 00011564 890D[88030300]      <1>       mov    [u.count], ecx ; required read count
12311 0001156A EB29                <1>       jmp    short sysexec_16
12312                              <1> sysexec_14:
12313                              <1>       ; insufficient (out of) memory
12314 0001156C C705[C8030300]0400- <1>       mov    dword [u.error], ERR_MINOR_IM ; 1
12314 00011574 0000                <1>
12315 00011576 E961C3FFFF          <1>       jmp    error
12316                              <1> sysexec_15:
12317 0001157B 8B15[55040300]      <1>         mov  edx, [i.size] ; file size
12318 00011581 29CA                <1>       sub    edx, ecx ; file size - loaded bytes
12319 00011583 7626                <1>       jna    short sysexec_17 ; no need to next read
12320 00011585 01D1                <1>       add    ecx, edx ; [i.size]
12321 00011587 3B0D[88030300]      <1>       cmp    ecx, [u.count] ; overrun check (!)
12322 0001158D 77DD                <1>       ja     short sysexec_14
12323 0001158F 8915[88030300]      <1>       mov    [u.count], edx
12324                              <1> sysexec_16:
12325 00011595 A1[51040300]        <1>       mov    eax, [ii] ; first cluster
12326 0001159A E889000000          <1>       call   readi
12327 0001159F 8B0D[8C030300]      <1>       mov    ecx, [u.nread]
12328 000115A5 010D[90030300]      <1>       add    [u.break], ecx
12329                              <1> sysexec_17:
12330 000115AB A1[51040300]        <1>       mov    eax, [ii] ; first cluster
12331 000115B0 E886150000          <1>       call   iclose
12332 000115B5 31C0                <1>       xor     eax, eax
12333 000115B7 FEC0                <1>       inc    al
```

```
12334 000115B9 66A3[AA030300]      <1>       mov    [u.intr], ax ; 1 (interrupt/time-out is enabled)
12335 000115BF 66A3[AC030300]      <1>       mov    [u.quit], ax ; 1 ('crtl+brk' signal is enabled)
12336 000115C5 833D[BC030300]00    <1>         cmp  dword [u.ppgdir], 0  ; is the caller MainProg (kernel) ?
12337 000115CC 770C                <1>       ja     short sysexec_18 ; no, the caller is user process
12338                              <1>       ; If the caller is kernel (MainProg), 'sysexec' will come here
12339 000115CE 8B15[C0890100]      <1>       mov    edx, [k_page_dir] ; kernel's page directory
12340 000115D4 8915[BC030300]      <1>       mov    [u.ppgdir], edx ; next time 'sysexec' must not come here
12341                              <1> sysexec_18:
12342                              <1>       ; 02/05/2016
12343                              <1>       ; 24/04/2016 (TRDOS 386 = TRDOS v2.0)
12344                              <1>       ; 18/10/2015 (Retro UNIX 386 v1)
12345                              <1>       ; 05/08/2015
12346                              <1>       ; 29/07/2015
12347                              <1>
12348                              <1> ;     ; **** arguments list test start - 19/11/2017
12349                              <1> ;     mov    ebp, [argv]
12350                              <1> ;     sub    ebp, ECORE - 4096
12351                              <1> ;     add    ebp, [ecore]
12352                              <1> ;
12353                              <1> ;     mov    ebx, [ebp]
12354                              <1> ;     mov    [argc], bx
12355                              <1> ;     add    ebp, 4
12356                              <1> ;     mov    byte [ccolor], 1Fh
12357                              <1> ;_zx0:
12358                              <1> ;     cmp    word [argc], 0
12359                              <1> ;     jna    short _zx2
12360                              <1> ;_zx1:
12361                              <1> ;     push   ebp
12362                              <1> ;     mov    esi, [ebp]
12363                              <1> ;
12364                              <1> ;     sub    esi, ECORE - 4096
12365                              <1> ;     add    esi, [ecore]
12366                              <1> ;
12367                              <1> ;     call   print_cmsg
12368                              <1> ;
12369                              <1> ;     dec    word [argc]
12370                              <1> ;     jz     short _zx2
12371                              <1> ;
12372                              <1> ;     mov    al, '.'
12373                              <1> ;     mov    bl, 07h
12374                              <1> ;     mov    bh, [u.ttyn]
12375                              <1> ;     call   _write_tty
12376                              <1> ;
12377                              <1> ;     pop    ebp
12378                              <1> ;     add    ebp, 4
12379                              <1> ;     jmp    short _zx1
12380                              <1> ;_zx2:
12381                              <1> ;     pop    ebp
12382                              <1> ;     mov    byte [ccolor], 07h
12383                              <1> ;     mov    eax, 1
12384                              <1> ;     ; **** arguments list test stop
12385                              <1> ;     Test result is OK! (there is not a wrong thing) - 19/11/2017
12386                              <1>
12387 000115DA 8B2D[4C040300]      <1>       mov    ebp, [argv] ; user's stack pointer must point to argument
12388                              <1>                          ; list pointers (argument count)
12389 000115E0 FA                  <1>       cli
12390 000115E1 8B25[5C890100]      <1>         mov     esp, [tss.esp0] ; ring 0 (kernel) stack pointer
12391                              <1>       ;mov         esp, [u.sp] ; Restore Kernel stack
12392                              <1>                          ; for this process
12393                              <1>       ;add   esp, 20 ; --> EIP, CS, EFLAGS, ESP, SS
12394                              <1>       ;xor   eax, eax ; 0
12395 000115E7 FEC8                <1>       dec    al ; eax = 0
12396                              <1>       ;mov   edx, UDATA
12397                              <1>       ; 18/11/2017
12398 000115E9 6A23                <1>       push   UDATA ; user's stack segment
12399                              <1>       ;push  edx
12400 000115EB 55                  <1>       push   ebp ; user's stack pointer
12401                              <1>                  ; (points to number of arguments)
12402                              <1>
12403                              <1>       ; 04/01/2017
12404                              <1>       ; MainProg comes here while [sysflg]= 0FFh
12405                              <1>       ; (but sysexec comes here while [sysflg]= 0)
12406 000115EC C605[5B030300]00    <1>       mov    byte [sysflg], 0 ; 04/01/2017
12407                              <1>                               ; (timer_int sysflg control)
12408 000115F3 FB                  <1>       sti
12409 000115F4 9C                  <1>       pushfd ; EFLAGS
12410                              <1>              ; Set IF for enabling interrupts in user mode
12411                              <1>       ;or    dword [esp], 200h
12412                              <1>       ;
12413                              <1>       ;mov   bx, UCODE
12414                              <1>       ;push  bx ; user's code segment
12415 000115F5 6A1B                <1>       push   UCODE
12416                              <1>       ;push  0
12417 000115F7 50                  <1>       push   eax ; EIP (=0) - start address -
12418 000115F8 8925[5C030300]      <1>       mov    [u.sp], esp ; 29/07/2015
12419                              <1>       ; 05/08/2015
12420                              <1>       ; Remedy of a General Protection Fault during 'iretd' is here !
12421                              <1>       ; ('push dx' would cause to general protection fault,
12422                              <1>       ; after 'pop ds' etc.)
12423                              <1>       ;
12424                              <1>       ;; push dx ; ds (UDATA)
12425                              <1>       ;; push dx ; es (UDATA)
12426                              <1>       ;; push dx ; fs (UDATA)
12427                              <1>       ;; push dx ; gs (UDATA)
12428                              <1>       ;
12429                              <1>       ; This is a trick to prevent general protection fault
12430                              <1>       ; during 'iretd' intruction at the end of 'sysrele' (in u1.s):
12431 000115FE 66BA2300            <1>       mov    dx, UDATA ; 19/11/2017
12432 00011602 8EC2                <1>       mov    es, dx ; UDATA
12433 00011604 06                  <1>       push   es ; ds (UDATA)
12434 00011605 06                  <1>       push   es ; es (UDATA)
12435 00011606 06                  <1>       push   es ; fs (UDATA)
12436 00011607 06                  <1>       push   es ; gs (UDATA)
12437 00011608 66BA1000            <1>       mov    dx, KDATA
12438 0001160C 8EC2                <1>       mov    es, dx
```

```
12439                                  <1>       ;
12440                                  <1>       ;; pushad simulation
12441 0001160E 89E5                    <1>       mov    ebp, esp ; esp before pushad
12442 00011610 50                      <1>       push   eax ; eax (0)
12443 00011611 50                      <1>       push   eax ; ecx (0)
12444 00011612 50                      <1>       push   eax ; edx (0)
12445 00011613 50                      <1>       push   eax ; ebx (0)
12446 00011614 55                      <1>       push   ebp ; esp before pushad
12447 00011615 50                      <1>       push   eax ; ebp (0)
12448 00011616 50                      <1>       push   eax ; esi (0)
12449 00011617 50                      <1>       push   eax ; edi (0)
12450                                  <1>       ;
12451 00011618 A3[64030300]            <1>       mov    [u.r0], eax ; eax = 0
12452 0001161D 8925[60030300]          <1>       mov    [u.usp], esp
12453                                  <1>
12454                                  <1>       ; 14/11/2017
12455 00011623 E9D6C2FFFF              <1>       jmp    sysret0
12456                                  <1>
12457                                  <1> ;     ; 02/05/2016
12458                                  <1> ;     ;inc   byte [sysflg] ; 0FFh -> 0
12459                                  <1> ;     ;mov   byte [sysflg], 0 ; 04/01/2017
12460                                  <1> ;     movzx  ebx, byte [u.uno]
12461                                  <1> ;     shl    bl, 1 ; 13/11/2017
12462                                  <1> ;     cmp    word [ebx+p.ppid-2], 1 ; MainProg
12463                                  <1> ;     ja     sysret0 ; 03/05/2016
12464                                  <1> ;     push   sysret ; *
12465                                  <1> ;     mov    [u.usp], esp
12466                                  <1> ;     call   wswap ; save child process 'u' structure and
12467                                  <1> ;                  ; registers
12468                                  <1> ;     add    dword [u.usp], 4 ; 03/05/2016
12469                                  <1> ;sysexec_19: ; 02/05/2016
12470                                  <1> ;     retn ; * 'sysret' ; byte [sysflg] -> 0FFh
12471                                  <1>
12472                                  <1> readi:
12473                                  <1>       ; 01/05/2016
12474                                  <1>       ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)
12475                                  <1>       ; 20/05/2015 - Retro UNIX 386 v1
12476                                  <1>       ; 11/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
12477                                  <1>       ;
12478                                  <1>       ; Reads from a file whose the first cluster number in EAX
12479                                  <1>       ;
12480                                  <1>       ; INPUTS ->
12481                                  <1>       ;    EAX - First cluster number of the file
12482                                  <1>       ;    u.count - byte count user desires
12483                                  <1>       ;    u.base - points to user buffer
12484                                  <1>       ;    u.fofp - points to dword with current file offset
12485                                  <1>       ;    i.size - file size
12486                                  <1>       ;    cdev - logical dos drive number of the file
12487                                  <1>       ; OUTPUTS ->
12488                                  <1>       ;    u.count - cleared
12489                                  <1>       ;    u.nread - accumulates total bytes passed back
12490                                  <1>       ;
12491                                  <1>       ; ((EAX)) input/output
12492                                  <1>       ; (Retro UNIX Prototype : 14/12/2012 - 01/03/2013, UNIXCOPY.ASM)
12493                                  <1>         ; ((Modified registers: edx, ebx, ecx, esi, edi))
12494                                  <1>
12495 00011628 31D2                    <1>       xor    edx, edx ; 0
12496 0001162A 8915[8C030300]          <1>       mov    [u.nread], edx ; 0
12497 00011630 668915[C4030300]        <1>       mov    [u.pcount], dx ; 19/05/2015
12498 00011637 3915[88030300]          <1>       cmp    [u.count], edx ; 0
12499 0001163D 7701                    <1>       ja     short readi_1
12500 0001163F C3                      <1>       retn
12501                                  <1> readi_1:
12502                                  <1> dskr:
12503                                  <1>       ; 01/05/2016
12504                                  <1>       ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)
12505                                  <1>       ; 24/05/2015 - 12/10/2015 (Retro UNIX 386 v1)
12506                                  <1>       ; 26/04/2013 - 03/08/2013 (Retro UNIX 8086 v1)
12507                                  <1> dskr_0:
12508 00011640 8B15[55040300]          <1>         mov edx, [i.size]
12509 00011646 8B1D[74030300]          <1>       mov    ebx, [u.fofp]
12510 0001164C 2B13                    <1>       sub    edx, [ebx]
12511 0001164E 7647                    <1>       jna    short dskr_4
12512                                  <1>       ;
12513 00011650 50                      <1>       push   eax ; 01/05/2016
12514 00011651 3B15[88030300]          <1>       cmp     edx, [u.count]
12515 00011657 7306                    <1>       jnb    short dskr_1
12516 00011659 8915[88030300]          <1>       mov    [u.count], edx
12517                                  <1> dskr_1:
12518                                  <1>       ; EAX = First Cluster
12519                                  <1>       ; [Current_Drv] = Physical drive number
12520 0001165F E83B000000              <1>       call   mget_r
12521                                  <1>       ; NOTE: in 'mget_r', relevant sector will be read in buffer
12522                                  <1>       ; if it is not already in buffer !
12523 00011664 BB[8C050300]            <1>       mov    ebx, readi_buffer
12524 00011669 803D[C6030300]00        <1>       cmp    byte [u.kcall], 0 ; the caller is 'namei' sign (=1)
12525 00011670 770F                    <1>       ja     short dskr_3   ; zf=0 -> the caller is 'namei'
12526 00011672 66833D[C4030300]00      <1>       cmp    word [u.pcount], 0
12527 0001167A 7705                    <1>       ja     short dskr_3
12528                                  <1> dskr_2:
12529                                  <1>       ; [u.base] = virtual address to transfer (as destination address)
12530 0001167C E894010000              <1>       call   trans_addr_w ; translate virtual address to physical (w)
12531                                  <1> dskr_3:
12532                                  <1>       ; EBX (r5) = system (I/O) buffer address -physical-
12533 00011681 E8F7010000              <1>       call   sioreg
12534 00011686 87F7                    <1>       xchg   esi, edi
12535                                  <1>       ; EDI = file (user data) offset
12536                                  <1>       ; ESI = sector (I/O) buffer offset
12537                                  <1>       ; ECX = byte count
12538 00011688 F3A4                    <1>       rep    movsb
12539                                  <1>       ; eax = remain bytes in buffer
12540                                  <1>         ;       (check if remain bytes in the buffer > [u.pcount])
12541 0001168A 09C0                    <1>       or     eax, eax
12542 0001168C 75EE                    <1>       jnz    short dskr_2 ; (page end before system buffer end!)
12543 0001168E 58                      <1>       pop    eax  ; (first cluster number)
```

```
12544 0001168F 390D[88030300]      <1>       cmp    [u.count], ecx ; 0
12545 00011695 77A9                <1>       ja     short dskr_0
12546                              <1> dskr_4:
12547 00011697 C605[C6030300]00    <1>       mov    byte [u.kcall], 0
12548 0001169E C3                  <1>       retn
12549                              <1>
12550                              <1> mget_r:
12551                              <1>       ; 24/10/2016
12552                              <1>       ; 22/10/2016
12553                              <1>       ; 12/10/2016
12554                              <1>       ; 29/04/2016
12555                              <1>       ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)
12556                              <1>       ; 03/06/2015 (Retro UNIX 386 v1, 'mget', u.5s)
12557                              <1>       ; 22/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
12558                              <1>       ;
12559                              <1>       ; Get existing or (allocate) a new disk block for file
12560                              <1>       ;
12561                              <1>       ; INPUTS ->
12562                              <1>       ;    [u.fofp] = file offset pointer
12563                              <1>       ;    EAX = First Cluster
12564                              <1>       ;    [cdev] = Logical dos drive number
12565                              <1>       ;    ([u.off] = file offset)
12566                              <1>       ; OUTPUTS ->
12567                              <1>       ;    EAX = logical sector number
12568                              <1>       ;    ESI = Logical Dos Drive Description Table address
12569                              <1>       ;
12570                              <1>       ; Modified registers: EDX, EBX, ECX, ESI, EDI
12571                              <1>
12572 0001169F 8B35[74030300]      <1>       mov     esi, [u.fofp]
12573 000116A5 8B1E                <1>       mov    ebx, [esi] ; (u.off)
12574                              <1>
12575 000116A7 29C9                <1>       sub    ecx, ecx
12576 000116A9 8A2D[46030300]      <1>       mov    ch, [cdev]
12577                              <1>
12578 000116AF BE00010900          <1>       mov    esi, Logical_DOSDisks
12579 000116B4 01CE                <1>       add    esi, ecx
12580                              <1>
12581 000116B6 380D[F4950100]      <1>       cmp    [readi.valid], cl ; 0
12582 000116BC 7649                <1>       jna    short mget_r_0
12583                              <1>
12584 000116BE 3A2D[F5950100]      <1>       cmp    ch, [readi.drv]
12585 000116C4 7541                <1>       jne    short mget_r_0
12586                              <1>
12587 000116C6 3B05[08960100]      <1>       cmp    eax, [readi.fclust]
12588 000116CC 7565                <1>       jne    short mget_r_3
12589                              <1>
12590 000116CE 89D8                <1>       mov    eax, ebx ; file offset
12591 000116D0 668B0D[FC950100]    <1>       mov    cx, [readi.bpc]
12592 000116D7 41                  <1>       inc    ecx ; <= 65536
12593 000116D8 29D2                <1>       sub    edx, edx
12594 000116DA F7F1                <1>       div    ecx
12595                              <1>
12596 000116DC 8B3D[04960100]      <1>       mov    edi, [readi.c_index] ; cluster index
12597                              <1>
12598 000116E2 39F8                <1>       cmp    eax, edi
12599 000116E4 757A                <1>         jne     short mget_r_4  ; (*)
12600                              <1>
12601                              <1>       ; edx = byte offset in cluster (<= 65535)
12602 000116E6 668915[FE950100]    <1>       mov    [readi.offset], dx
12603 000116ED 66C1EA09            <1>       shr    dx, 9 ; / 512
12604 000116F1 8815[F7950100]      <1>       mov    [readi.s_index], dl ; sector index in cluster (0 to spc -1)
12605                              <1>
12606 000116F7 A1[00960100]        <1>       mov    eax, [readi.cluster]  ; > 0 if [readi.valid] = 1
12607 000116FC 8B15[0C960100]      <1>       mov    edx, [readi.fs_index]
12608 00011702 E99A000000          <1>         jmp     mget_r_7
12609                              <1>
12610                              <1> mget_r_0:
12611 00011707 882D[F5950100]      <1>       mov    [readi.drv], ch ; physical drive number
12612 0001170D 807E0300            <1>       cmp    byte [esi+LD_FATType], 0
12613 00011711 7707                <1>       ja     short mget_r_1
12614 00011713 8A4E12              <1>       mov    cl, [esi+LD_FS_BytesPerSec+1]
12615 00011716 D0E9                <1>       shr    cl, 1 ;  ; 1 for 512 bytes, 4 for 2048 bytes
12616 00011718 EB03                <1>       jmp    short mget_r_2
12617                              <1> mget_r_1:
12618 0001171A 8A4E13              <1>       mov    cl, [esi+LD_BPB+BPB_SecPerClust]
12619                              <1> mget_r_2:
12620 0001171D 880D[F6950100]      <1>       mov    [readi.spc], cl  ; sectors per cluster
12621                              <1>       ; NOTE: readi bytes per sector value is always 512 !
12622 00011723 66C1E109            <1>       shl    cx, 9 ; * 512
12623 00011727 6649                <1>       dec    cx ; bytes per cluster - 1
12624 00011729 66890D[FC950100]    <1>       mov    [readi.bpc], cx
12625 00011730 6629C9              <1>       sub    cx, cx
12626                              <1> mget_r_3:
12627 00011733 A3[08960100]        <1>       mov    [readi.fclust], eax ; first cluster (or FDT address)
12628 00011738 880D[F4950100]      <1>       mov    [readi.valid], cl ; 0
12629                              <1>       ;mov   [readi.s_index], cl ; 0
12630                              <1>       ;mov   [readi.offset], cx ; 0
12631 0001173E 890D[04960100]      <1>       mov    [readi.c_index], ecx ; 0
12632 00011744 890D[00960100]      <1>       mov    [readi.cluster], ecx ; 0
12633 0001174A 890D[F8950100]      <1>       mov    [readi.sector], ecx ; 0
12634                              <1>
12635 00011750 89D8                <1>       mov    eax, ebx ; file offset
12636 00011752 668B0D[FC950100]    <1>       mov    cx, [readi.bpc]
12637 00011759 41                  <1>       inc    ecx ; <= 65536
12638 0001175A 29D2                <1>       sub    edx, edx
12639 0001175C F7F1                <1>       div    ecx
12640                              <1>       ;mov   edi, [readi.c_index] ; previous cluster index
12641 0001175E 29FF                <1>       sub    edi, edi
12642                              <1> mget_r_4:
12643 00011760 A3[04960100]        <1>       mov    [readi.c_index], eax ; cluster index
12644                              <1>       ; edx = byte offset in cluster (<= 65535)
12645 00011765 668915[FE950100]    <1>       mov    [readi.offset], dx
12646 0001176C 66C1EA09            <1>       shr    dx, 9 ; / 512
12647 00011770 8815[F7950100]      <1>       mov    [readi.s_index], dl ; sector index in cluster (0 to spc -1)
12648                              <1>
```

```
12649 00011776 89C1               <1>       mov   ecx, eax ; current cluster index
12650 00011778 A1[08960100]       <1>       mov   eax, [readi.fclust]
12651 0001177D 09C9               <1>       or    ecx, ecx ; cluster index
12652 0001177F 741B               <1>       jz    short mget_r_6
12653                             <1>
12654 00011781 39CF               <1>       cmp   edi, ecx
12655 00011783 7710               <1>       ja    short mget_r_5 ; old cluster index is higher
12656 00011785 8B15[00960100]     <1>       mov   edx, [readi.cluster]
12657 0001178B 21D2               <1>       and   edx, edx
12658 0001178D 7406               <1>       jz    short mget_r_5
12659                             <1>       ; valid 'readi' parameters (*)
12660 0001178F 89D0               <1>       mov   eax, edx
12661 00011791 29F9               <1>       sub   ecx, edi
12662 00011793 740C               <1>       jz    short mget_r_7
12663                             <1> mget_r_5:
12664                             <1>       ; EAX = Beginning cluster
12665                             <1>       ; EDX = Sector index in disk/file section
12666                             <1>       ;       (Only for SINGLIX file system!)
12667                             <1>       ; ECX = Cluster sequence number after the beginning cluster
12668                             <1>       ; ESI = Logical DOS Drive Description Table address
12669 00011795 E8DBBFFFFF         <1>       call  get_cluster_by_index
12670 0001179A 724E               <1>       jc    short mget_r_err
12671                             <1>       ; EAX = Cluster number
12672                             <1> mget_r_6:
12673 0001179C A3[00960100]       <1>       mov   [readi.cluster], eax ; FDT number for Singlix File System
12674                             <1> mget_r_7:
12675 000117A1 807E0300           <1>       cmp   byte [esi+LD_FATType], 0
12676 000117A5 765F               <1>       jna   short mget_r_12
12677                             <1>
12678 000117A7 83E802             <1>       sub   eax, 2
12679 000117AA 0FB615[F6950100]   <1>       movzx edx, byte [readi.spc]
12680 000117B1 F7E2               <1>       mul   edx
12681                             <1>
12682 000117B3 034668             <1>       add   eax, [esi+LD_DATABegin]
12683 000117B6 8A15[F7950100]     <1>       mov   dl, [readi.s_index]
12684 000117BC 01D0               <1>       add   eax, edx
12685                             <1> mget_r_8:
12686                             <1>       ; eax = logical sector number
12687 000117BE 803D[F4950100]00   <1>       cmp   byte [readi.valid], 0
12688 000117C5 7608               <1>       jna   short mget_r_9
12689 000117C7 3B05[F8950100]     <1>       cmp   eax, [readi.sector]
12690 000117CD 7436               <1>       je    short mget_r_11 ; sector is already in 'readi' buffer
12691                             <1> mget_r_9:
12692 000117CF A3[F8950100]       <1>       mov   [readi.sector], eax
12693 000117D4 BB[8C050300]       <1>       mov   ebx, readi_buffer ; buffer address
12694 000117D9 B901000000         <1>       mov   ecx, 1
12695                             <1>       ; 29/04/2016
12696                             <1>       ;xor  dl, dl
12697                             <1>
12698                             <1>       ; EAX = Logical sector number
12699                             <1>       ; ECX = Sector count
12700                             <1>       ; EBX = Buffer address
12701                             <1>       ; (EDX = 0)
12702                             <1>       ; ESI = Logical DOS drive description table address
12703                             <1>
12704 000117DE E868130000         <1>       call  disk_read
12705 000117E3 7314               <1>       jnc   short mget_r_10
12706                             <1>
12707                             <1>       ; 22/10/2016 (15h -> 17)
12708 000117E5 B811000000         <1>       mov   eax, 17 ; Drive not ready or read error !
12709                             <1> mget_r_err:
12710 000117EA A3[C8030300]       <1>       mov   [u.error], eax
12711                             <1>       ; 12/10/2016
12712 000117EF A3[64030300]       <1>       mov   [u.r0], eax
12713 000117F4 E9E3C0FFFF         <1>       jmp   error
12714                             <1> mget_r_10:
12715 000117F9 C605[F4950100]01   <1>       mov   byte [readi.valid], 1 ; 24/10/2016
12716 00011800 A1[F8950100]       <1>       mov   eax, [readi.sector]
12717                             <1> mget_r_11:
12718 00011805 C3                 <1>       retn
12719                             <1> mget_r_12:
12720                             <1>       ; EAX = FDT number
12721                             <1>       ; EDX = Sector index from FDT sector (0,1,2,3,4...)
12722 00011806 40                 <1>       inc   eax ; the first data sector in FS disk section
12723 00011807 8915[0C960100]     <1>       mov   [readi.fs_index], edx
12724 0001180D 01D0               <1>       add   eax, edx
12725 0001180F EBAD               <1>       jmp   short mget_r_8
12726                             <1>
12727                             <1> trans_addr_r:
12728                             <1>       ; 12/10/2016
12729                             <1>       ; 02/05/2016 - TRDOS 386 (TRDOS v2.0)
12730                             <1>       ; Translate virtual address to physical address
12731                             <1>       ; for reading from user's memory space
12732                             <1>       ; 04/06/2015 - 18/10/2015 (Retro UNIX 386 v1)
12733                             <1>
12734 00011811 31D2               <1>       xor   edx, edx ; 0 (read access sign)
12735 00011813 EB04               <1>       jmp   short trans_addr_rw
12736                             <1>
12737                             <1> trans_addr_w:
12738                             <1>       ; 12/10/2016
12739                             <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
12740                             <1>       ; Translate virtual address to physical address
12741                             <1>       ; for writing to user's memory space
12742                             <1>       ; 04/06/2015 - 18/10/2015 (Retro UNIX 386 v1)
12743                             <1>
12744 00011815 29D2               <1>       sub   edx, edx
12745 00011817 FEC2               <1>       inc   dl ; 1 (write access sign)
12746                             <1> trans_addr_rw:
12747 00011819 50                 <1>       push  eax
12748 0001181A 53                 <1>       push  ebx
12749 0001181B 52                 <1>       push  edx ; r/w sign (in DL)
12750                             <1>       ;
12751 0001181C 8B1D[84030300]     <1>       mov   ebx, [u.base]
12752 00011822 E8C14AFFFF         <1>       call  get_physical_addr ; get physical address
12753 00011827 730F               <1>       jnc   short passc_0
```

```
12754 00011829 A3[C8030300]        <1>       mov   [u.error], eax
12755 0001182E A3[64030300]        <1>       mov   [u.r0], eax ; 12/10/2016
12756                              <1>       ;pop  edx
12757                              <1>       ;pop  ebx
12758                              <1>       ;pop  eax
12759 00011833 E9A4C0FFFF          <1>       jmp   error
12760                              <1> passc_0:
12761 00011838 F6C202              <1>       test  dl, PTE_A_WRITE ; writable page
12762 0001183B 5A                  <1>       pop   edx
12763 0001183C 751C                <1>       jnz   short passc_1
12764                              <1>
12765 0001183E 20D2                <1>       and   dl, dl
12766 00011840 7418                <1>       jz    short passc_1
12767                              <1>       ; read only (duplicated) page -must be copied to a new page-
12768                              <1>       ; EBX = linear address
12769 00011842 51                  <1>       push  ecx
12770 00011843 E83947FFFF          <1>       call  copy_page
12771 00011848 59                  <1>       pop   ecx
12772 00011849 721E                <1>       jc    short passc_2
12773 0001184B 50                  <1>       push  eax ; physical address of the new/allocated page
12774 0001184C E8C149FFFF          <1>       call  add_to_swap_queue
12775 00011851 58                  <1>       pop   eax
12776 00011852 81E3FF0F0000        <1>       and   ebx, PAGE_OFF ; 0FFFh
12777                              <1>       ;mov  ecx, PAGE_SIZE
12778                              <1>       ;sub  ecx, ebx
12779 00011858 01D8                <1>       add   eax, ebx
12780                              <1> passc_1:
12781 0001185A A3[C0030300]        <1>       mov   [u.pbase], eax ; physical address
12782 0001185F 66890D[C4030300]    <1>       mov   [u.pcount], cx ; remain byte count in page (1-4096)
12783 00011866 5B                  <1>       pop   ebx
12784 00011867 58                  <1>       pop   eax
12785 00011868 C3                  <1>       retn
12786                              <1> passc_2:
12787 00011869 B804000000          <1>       mov   eax, ERR_MINOR_IM ; "Insufficient memory !" error
12788 0001186E A3[64030300]        <1>       mov   [u.r0], eax ; 12/10/2016
12789 00011873 A3[C8030300]        <1>       mov   dword [u.error], eax
12790                              <1>       ;pop  ebx
12791                              <1>       ;pop  eax
12792 00011878 E95FC0FFFF          <1>       jmp   error
12793                              <1>
12794                              <1> sioreg:
12795                              <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
12796                              <1>       ; 19/05/2015 - 25/07/2015 (Retro UNIX 386 v1)
12797                              <1>       ; 12/03/2013 - 22/07/2013 (Retro UNIX 8086 v1)
12798                              <1>       ; INPUTS ->
12799                              <1>       ;     EBX = system buffer (data) address (r5)
12800                              <1>       ;     [u.fofp] = pointer to file offset pointer
12801                              <1>       ;     [u.base] = virtual address of the user buffer
12802                              <1>       ;     [u.pbase] = physical address of the user buffer
12803                              <1>       ;     [u.count] = byte count
12804                              <1>       ;     [u.pcount] = byte count within page frame
12805                              <1>       ; OUTPUTS ->
12806                              <1>       ;     ESI = user data offset (r1)
12807                              <1>       ;     EDI = system (I/O) buffer offset (r2)
12808                              <1>       ;     ECX = byte count (r3)
12809                              <1>       ;     EAX = remain bytes after byte count within page frame
12810                              <1>       ;      (If EAX > 0, transfer will continue from the next page)
12811                              <1>         ;
12812                              <1>       ; ((Modified registers:  EDX))
12813                              <1>
12814 0001187D 8B35[74030300]      <1>         mov     esi, [u.fofp]
12815 00011883 8B3E                <1>         mov     edi, [esi]
12816 00011885 89F9                <1>       mov   ecx, edi
12817 00011887 81C900FEFFFF        <1>       or    ecx, 0FFFFFE00h
12818 0001188D 81E7FF010000        <1>       and   edi, 1FFh
12819 00011893 01DF                <1>       add   edi, ebx ; EBX = system buffer (data) address
12820 00011895 F7D9                <1>       neg   ecx
12821 00011897 3B0D[88030300]      <1>       cmp   ecx, [u.count]
12822 0001189D 7606                <1>       jna   short sioreg_0
12823 0001189F 8B0D[88030300]      <1>       mov   ecx, [u.count]
12824                              <1> sioreg_0:
12825 000118A5 803D[C6030300]00    <1>       cmp   byte [u.kcall], 0
12826 000118AC 7613                <1>       jna   short sioreg_1
12827                              <1>        ; the caller is 'mkdir' or 'namei'
12828 000118AE A1[84030300]        <1>       mov   eax, [u.base]
12829 000118B3 A3[C0030300]        <1>       mov   [u.pbase], eax ; physical address = virtual address
12830 000118B8 66890D[C4030300]    <1>       mov   word [u.pcount], cx ; remain bytes in buffer (1 sector)
12831 000118BF EB0B                <1>       jmp   short sioreg_2
12832                              <1> sioreg_1:
12833 000118C1 0FB715[C4030300]    <1>       movzx  edx, word [u.pcount]
12834 000118C8 39D1                <1>       cmp   ecx, edx
12835 000118CA 772A                <1>       ja    short sioreg_4 ; transfer count > [u.pcount]
12836                              <1> sioreg_2: ; 2:
12837 000118CC 31C0                <1>       xor   eax, eax
12838                              <1> sioreg_3:
12839 000118CE 010D[8C030300]      <1>       add   [u.nread], ecx
12840 000118D4 290D[88030300]      <1>       sub   [u.count], ecx
12841 000118DA 010D[84030300]      <1>       add   [u.base], ecx
12842 000118E0 010E                <1>         add [esi], ecx
12843 000118E2 8B35[C0030300]      <1>       mov   esi, [u.pbase]
12844 000118E8 66290D[C4030300]    <1>       sub   [u.pcount], cx
12845 000118EF 010D[C0030300]      <1>       add   [u.pbase], ecx
12846 000118F5 C3                  <1>         retn
12847                              <1> sioreg_4:
12848                              <1>       ; transfer count > [u.pcount]
12849                              <1>       ; (ecx > edx)
12850 000118F6 89C8                <1>       mov   eax, ecx
12851 000118F8 29D0                <1>       sub   eax, edx ; remain bytes for 1 sector (block) transfer
12852 000118FA 89D1                <1>       mov   ecx, edx ; current transfer count = [u.pcount]
12853 000118FC EBD0                <1>       jmp   short sioreg_3
12854                              <1>
12855                              <1> tswitch: ; Retro UNIX 386 v1
12856                              <1> tswap:
12857                              <1>       ; 16/01/2017
12858                              <1>       ; 21/05/2016 - TRDOS 386 (TRDOS v2.0)
```

```
12859                                 <1>       ; 10/05/2015 - 01/09/2015 (Retro UNIX 386 v1)
12860                                 <1>       ; 14/04/2013 - 14/02/2014 (Retro UNIX 8086 v1)
12861                                 <1>       ; time out swap, called when a user times out.
12862                                 <1>       ; the user is put on the low priority queue.
12863                                 <1>       ; This is done by making a link from the last user
12864                                 <1>       ; on the low priority queue to him via a call to 'putlu'.
12865                                 <1>       ; then he is swapped out.
12866                                 <1>
12867                                 <1>       ; TRDOS 386 (TRDOS v2.0) modification ->  ** 21/05/2016 **
12868                                 <1>       ;     * when a high priority (event) process will be stopped
12869                                 <1>       ;      (swapped out, swithched out/off), 'tswap/tswitch' will
12870                                 <1>       ;      not add it to a run queue.
12871                                 <1>       ;      /// What for: Process may be already in a run queue,
12872                                 <1>       ;      it is unspeficied state because process might be started
12873                                 <1>       ;      by a timer event which does not regard previous priority
12874                                 <1>       ;      level and run queue of the process (for fast executing!).
12875                                 <1>       ;      After the 'run for event', process will be sequenced
12876                                 <1>       ;      to run by it's actual run queue. ///
12877                                 <1>       ;
12878                                 <1>       ; Retro UNIX 386 v1 modification ->
12879                                 <1>       ;       swap (software task switch) is performed by changing
12880                                 <1>       ;      user's page directory (u.pgdir) instead of segment change
12881                                 <1>       ;      as in Retro UNIX 8086 v1.
12882                                 <1>       ;
12883                                 <1>       ; RETRO UNIX 8086 v1 modification ->
12884                                 <1>       ;       'swap to disk' is replaced with 'change running segment'
12885                                 <1>       ;      according to 8086 cpu (x86 real mode) architecture.
12886                                 <1>       ;      pdp-11 was using 64KB uniform memory while IBM PC
12887                                 <1>       ;      compatibles was using 1MB segmented memory
12888                                 <1>       ;      in 8086/8088 times.
12889                                 <1>       ;
12890                                 <1>       ; INPUTS ->
12891                                 <1>       ;    u.uno - users process number
12892                                 <1>       ;    runq+4 - lowest priority queue
12893                                 <1>       ; OUTPUTS ->
12894                                 <1>       ;    r0 - users process number
12895                                 <1>       ;    r2 - lowest priority queue address
12896                                 <1>       ;
12897                                 <1>       ; ((AX = R0, BX = R2)) output
12898                                 <1>       ; ((Modified registers: EDX, EBX, ECX, ESI, EDI))
12899                                 <1>       ;
12900                                 <1>
12901                                 <1>       NOTE:
12902                                 <1>       ;* [u.pri] priority level is specified by run queue which is process
12903                                 <1>       ;  comes to run from.
12904                                 <1>       ;* Initial [u.pri] is 1 ('normal/regular') for programs
12905                                 <1>       ;  (which are launched by MainProg or 'sysexec'), it is changed
12906                                 <1>       ;  to 2 ('high') by timer event, if program uses 'systimer' system call.
12907                                 <1>       ;* Program (Process) also can change it's running priority
12908                                 <1>       ;  from 1 to 0 or up to 2 by using 'syspri' system call; but,
12909                                 <1>       ;  if program selects priority level 2 (high) for running, next time
12910                                 <1>       ;  it is reduced to 1 (normal/regular) because 'syspri' adds this
12911                                 <1>       ;  program to 'run for normal' queue while running duration is a bit
12912                                 <1>       ;  protected from swap/switch out immediate, behalf of other high
12913                                 <1>       ;  priority process in sequence. Program (with high priority) will not
12914                                 <1>       ;  be swapped/switched out (by timer event) before it's time quantum
12915                                 <1>       ;  will be elapsed, but, this will be temporary if program is not using
12916                                 <1>       ;  timer event function.
12917                                 <1>
12918                                 <1>       ;For example:
12919                                 <1>       ;If a process frequently gets a timer event, it runs at high priority
12920                                 <1>       ;level but when it returns from running it returns to actual run queue,
12921                                 <1>       ;not to 'run for event' queue again.
12922                                 <1>       ;'tswap' will not change the sequence at return/stop(swap out) stage.
12923                                 <1>       ;But if priority level not high (=2, 'run for event'), 'tswap/tswitch'
12924                                 <1>       ;will add the stopping process to relevant run queue according to
12925                                 <1>       ;[u.pri] priority level.
12926                                 <1>
12927                                 <1>       ; 16/01/2017
12928 000118FE BB[54030300]          <1>       mov    ebx, runq+2  ; 'runq_normal' ; normal/regular priority
12929                                 <1>       ; 21/05/2016
12930                                 <1>       ;cmp   byte [u.pri], 2     ; high priority (run for event) ?
12931                                 <1>       ;jnb   short swap
12932                                 <1>       ; 16/01/2017
12933                                 <1>       ; (Normal and also high/event priority processes will be added to
12934                                 <1>       ; normal priority run queue for ensuring circular running sequence!)
12935                                 <1>       ; (Timer interrupt or 'syspri' system call may change priority and run
12936                                 <1>       ; queue to high/event level.)
12937 00011903 803D[A9030300]00      <1>       cmp    byte [u.pri], 0
12938 0001190A 7702                  <1>       ja     short tswap_1; normal priority run queue
12939                                 <1>       ;
12940 0001190C 43                    <1>       inc    ebx
12941 0001190D 43                    <1>       inc    ebx          ; runq+4, 'runq_background', low priority
12942                                 <1> tswap_1:
12943 0001190E A0[B3030300]          <1>       mov    al, [u.uno]
12944                                 <1>                    ; movb u.uno,r1 / move users process number to r1
12945                                 <1>              ; mov  $runq+4,r2
12946                                 <1>                    ; / move lowest priority queue address to r2
12947                                 <1>              ; ebx = run queue
12948 00011913 E8FE000000            <1>       call   putlu
12949                                 <1>              ; jsr r0,putlu / create link from last user on Q to
12950                                 <1>                          ; / u.uno's user
12951                                 <1>
12952                                 <1> switch: ; Retro UNIX 386 v1
12953                                 <1> swap:
12954                                 <1>       ; 02/01/2017
12955                                 <1>       ; 21/05/2016
12956                                 <1>       ; 20/05/2016
12957                                 <1>       ; 02/05/2016
12958                                 <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
12959                                 <1>       ; 10/05/2015 - 02/09/2015 (Retro UNIX 386 v1)
12960                                 <1>       ; 14/04/2013 - 08/03/2014 (Retro UNIX 8086 v1)
12961                                 <1>       ;
12962                                 <1>       ; 'swap' is routine that controls the swapping of processes
12963                                 <1>       ; in and out of core.
```

```
12964                                  <1>       ;
12965                                  <1>       ; TRDOS 386 (TRDOS v2.0) modification ->  ** 20/05/2016 **
12966                                  <1>       ;     * 3 different priority level is applied
12967                                  <1>       ;      (just as original unix v1)
12968                                  <1>       ;      1) high priority (event) run queue, 'runq_event'
12969                                  <1>       ;      2) normal priority (regular) run queue, 'runq_normal'
12970                                  <1>       ;      3) low priority (background) run queue, 'runq_backgroud'
12971                                  <1>       ;      'swap' code will run a process which has max. priority
12972                                  <1>         ;       (for earliest event at first)
12973                                  <1>       ;
12974                                  <1>       ; Retro UNIX 386 v1 modification ->
12975                                  <1>       ;       swap (software task switch) is performed by changing
12976                                  <1>       ;      user's page directory (u.pgdir) instead of segment change
12977                                  <1>       ;      as in Retro UNIX 8086 v1.
12978                                  <1>       ;
12979                                  <1>       ; RETRO UNIX 8086 v1 modification ->
12980                                  <1>       ;       'swap to disk' is replaced with 'change running segment'
12981                                  <1>       ;      according to 8086 cpu (x86 real mode) architecture.
12982                                  <1>       ;      pdp-11 was using 64KB uniform memory while IBM PC
12983                                  <1>       ;      compatibles was using 1MB segmented memory
12984                                  <1>       ;      in 8086/8088 times.
12985                                  <1>       ;
12986                                  <1>       ; INPUTS ->
12987                                  <1>       ;    runq table - contains processes to run.
12988                                  <1>       ;    p.link - contains next process in line to be run.
12989                                  <1>       ;    u.uno - process number of process in core
12990                                  <1>       ;    s.stack - swap stack used as an internal stack for swapping.
12991                                  <1>       ; OUTPUTS ->
12992                                  <1>       ;    (original unix v1 -> present process to its disk block)
12993                                  <1>       ;    (original unix v1 -> new process into core ->
12994                                  <1>       ;        Retro Unix 8086 v1 -> segment registers changed
12995                                  <1>       ;        for new process)
12996                                  <1>       ;    u.quant = 3 (Time quantum for a process)
12997                                  <1>       ;      ((INT 1Ch count down speed -> 18.2 times per second)
12998                                  <1>       ;    RETRO UNIX 8086 v1 will use INT 1Ch (18.2 times per second)
12999                                  <1>       ;       for now, it will swap the process if there is not
13000                                  <1>       ;       a keyboard event (keystroke) (Int 15h, function 4Fh)
13001                                  <1>       ;       or will count down from 3 to 0 even if there is a
13002                                  <1>       ;         keyboard event locking due to repetitive key strokes.
13003                                  <1>       ;       u.quant will be reset to 3 for RETRO UNIX 8086 v1.
13004                                  <1>       ;
13005                                  <1>       ; ((Modified registers: EAX, EDX, EBX, ECX, ESI, EDI))
13006                                  <1>
13007                                  <1>       ;NOTE:
13008                                  <1>       ;High priority queue is the first for selecting a process to run.
13009                                  <1>       ;If there is not a process in high priority level run queue,
13010                                  <1>       ;a process in normal priority run queue will be selected
13011                                  <1>       ;or a proces in low priority run queue will be selected if normal
13012                                  <1>       ;priority level run queue is empty.
13013                                  <1>
13014                                  <1>       ; 21/05/2016 -(3 priority levels, 3 run queues)
13015 00011918 BE[52030300]            <1>       mov   esi, runq ; 'runq_event' ; high priority, 'run for event'
13016 0001191D C605[50960100]03        <1>       mov   byte [priority], 3 ; high priority + 1
13017 00011924 31DB                    <1>       xor   ebx, ebx ; 02/01/2017
13018                                  <1> swap_0: ; 1: / search runq table for highest priority process
13019 00011926 66AD                    <1>       lodsw  ; mov ax, [esi], add esi+2
13020                                  <1>       ;xor   ebx, ebx ; 02/05/2016
13021 00011928 6621C0                  <1>       and   ax, ax ; are there any processes to run in this Q entry
13022 0001192B 750E                    <1>       jnz   short swap_2
13023                                  <1>       ; 21/05/2026
13024                                  <1>       ; runq_normal = runq+2, runq_background = runq+4
13025 0001192D FE0D[50960100]          <1>       dec   byte [priority] ; 3 -> 3, 2 -> 1, 1-> 0
13026 00011933 75F1                    <1>       jnz   short swap_0
13027                                  <1>       ;cmp   esi, runq+6  ; if zero compare address to end of table
13028                                  <1>       ;jb    short swap_0 ; if not at end, go back
13029                                  <1> swap_1:
13030                                  <1>       ; 02/05/2016
13031                                  <1>       ; 29/04/2016 (TRDOS 386 = TRDOS v2.0)
13032                                  <1>       ; No user process to run...
13033                                  <1>       ; Run the kernel process... MainProg: Internal Command Interpreter
13034 00011935 FEC0                    <1>       inc   al ; mov al, 1  ; process number of MainProg
13035 00011937 FEC3                    <1>       inc   bl ; mov bl, al ; 1
13036 00011939 EB1E                    <1>       jmp   short swap_4
13037                                  <1> swap_2:
13038                                  <1>       ; 21/05/2016
13039 0001193B FE0D[50960100]          <1>       dec   byte [priority] ; priority level of present user/process
13040                                  <1>                             ; 0, 1, 2
13041 00011941 4E                      <1>       dec   esi
13042 00011942 4E                      <1>         dec     esi
13043                                  <1>       ;
13044 00011943 88C3                    <1>       mov   bl, al
13045 00011945 38E0                    <1>       cmp   al, ah ; is there only 1 process in the queue to be run
13046 00011947 740A                    <1>       je    short swap_3 ; yes
13047 00011949 8AA3[9F000300]          <1>       mov   ah, [ebx+p.link-1]
13048 0001194F 8826                    <1>             mov   [esi], ah ; move next process in line into run queue
13049 00011951 EB06                    <1>       jmp   short swap_4
13050                                  <1> swap_3:
13051 00011953 6631D2                  <1>       xor   dx, dx
13052 00011956 668916                  <1>       mov   [esi], dx ; zero the entry; no processes on the Q
13053                                  <1> swap_4:
13054 00011959 8A25[B3030300]          <1>       mov   ah, [u.uno]
13055 0001195F 38C4                    <1>       cmp   ah, al ; is this process the same as the process in core?
13056 00011961 743B                    <1>             je    short swap_8 ; yes, don't have to swap
13057 00011963 08E4                    <1>       or    ah, ah  ; is the process # = 0
13058 00011965 740D                    <1>             jz    short swap_6 ; 'sysexit'
13059                                  <1>       ;cmp   ah, al ;is this process the same as the process in core?
13060                                  <1>             ;je    short swap_8 ; yes, don't have to swap
13061 00011967 8925[60030300]          <1>       mov   [u.usp], esp ; return  address for 'syswait' & 'sleep'
13062 0001196D E834000000              <1>       call  wswap   ; write out core to disk
13063 00011972 EB1C                    <1>       jmp   short swap_7
13064                                  <1> swap_6:
13065                                  <1>       ; Deallocate memory pages belong to the process
13066                                  <1>       ; which is being terminated.
13067                                  <1>       ; (Retro UNIX 386 v1 modification !)
13068                                  <1>       ;
```

```
13069 00011974 53                  <1>       push   ebx
13070 00011975 A1[B8030300]        <1>       mov    eax, [u.pgdir]  ; page directory of the process
13071 0001197A 8B1D[BC030300]      <1>       mov    ebx, [u.ppgdir] ; page directory of the parent process
13072 00011980 E88743FFFF          <1>       call   deallocate_page_dir
13073 00011985 A1[B4030300]        <1>       mov    eax, [u.upage] ; 'user' structure page of the process
13074 0001198A E82244FFFF          <1>       call   deallocate_page
13075 0001198F 5B                  <1>       pop    ebx
13076                              <1> swap_7:
13077 00011990 C0E302              <1>       shl    bl, 2 ; * 4
13078 00011993 8B83[BC000300]      <1>       mov    eax, [ebx+p.upage-4] ; the 'u' page of the new process
13079 00011999 E840000000          <1>       call   rswap ; read new process into core
13080                              <1> swap_8:
13081                              <1>       ; Retro UNIX  8086 v1 modification !
13082 0001199E C605[A8030300]04    <1>       mov    byte [u.quant], time_count
13083 000119A5 C3                  <1>       retn
13084                              <1>
13085                              <1> wswap:  ; < swap out, swap to disk >
13086                              <1>       ; 28/02/2017 (fnsave)
13087                              <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
13088                              <1>       ; 09/05/2015 (Retro UNIX 386 v1)
13089                              <1>       ; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)
13090                              <1>       ; 'wswap' writes out the process that is in core onto its
13091                              <1>       ; appropriate disk area.
13092                              <1>       ;
13093                              <1>       ; Retro UNIX 386 v1 modification ->
13094                              <1>       ;       User (u) structure content and the user's register content
13095                              <1>       ;     will be copied to the process's/user's UPAGE (a page for
13096                              <1>       ;     saving 'u' structure and user registers for task switching).
13097                              <1>       ;     u.usp - points to kernel stack address which contains
13098                              <1>       ;             user's registers while entering system call.
13099                              <1>       ;     u.sp  - points to kernel stack address
13100                              <1>       ;             to return from system call -for IRET-.
13101                              <1>       ;     [u.usp]+32+16 = [u.sp]
13102                              <1>       ;     [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,
13103                              <1>       ;             edx, ecx, eax, gs, fs, es, ds, -> [u.sp].
13104                              <1>       ;
13105                              <1>       ; Retro UNIX 8086 v1 modification ->
13106                              <1>       ;       'swap to disk' is replaced with 'change running segment'
13107                              <1>       ;     according to 8086 cpu (x86 real mode) architecture.
13108                              <1>       ;     pdp-11 was using 64KB uniform memory while IBM PC
13109                              <1>       ;     compatibles was using 1MB segmented memory
13110                              <1>       ;     in 8086/8088 times.
13111                              <1>       ;
13112                              <1>       ; INPUTS ->
13113                              <1>       ;    u.break - points to end of program
13114                              <1>       ;    u.usp - stack pointer at the moment of swap
13115                              <1>       ;    core - beginning of process program
13116                              <1>       ;    ecore - end of core
13117                              <1>       ;    user - start of user parameter area
13118                              <1>       ;    u.uno - user process number
13119                              <1>       ;    p.dska - holds block number of process
13120                              <1>       ; OUTPUTS ->
13121                              <1>       ;    swp I/O queue
13122                              <1>       ;    p.break - negative word count of process
13123                              <1>       ;    r1 - process disk address
13124                              <1>       ;    r2 - negative word count
13125                              <1>       ;
13126                              <1>       ; RETRO UNIX 8086 v1 input/output:
13127                              <1>       ;
13128                              <1>       ; INPUTS ->
13129                              <1>       ;    u.uno - process number (to be swapped out)
13130                              <1>       ; OUTPUTS ->
13131                              <1>       ;    none
13132                              <1>       ;
13133                              <1>       ;   ((Modified registers: ECX, ESI, EDI))
13134                              <1>       ;
13135                              <1>
13136                              <1>       ; 28/02/2017
13137                              <1>       ;cmp   byte [multi_tasking], 0  ; Musti tasking mode ?
13138                              <1>       ;jna   short wswp
13139 000119A6 803D[DA030300]00    <1>       cmp    byte [u.fpsave], 0 ; 28/02/2017
13140 000119AD 7606                <1>       jna    short wswp
13141 000119AF DD35[DC030300]      <1>       fnsave [u.fpregs] ; save floating point registers (94 bytes)
13142                              <1> wswp:
13143 000119B5 8B3D[B4030300]      <1>       mov    edi, [u.upage] ; process's user (u) structure page addr
13144 000119BB B938000000          <1>       mov    ecx, (U_SIZE + 3) / 4
13145 000119C0 BE[5C030300]        <1>       mov    esi, user ; active user (u) structure
13146 000119C5 F3A5                <1>       rep    movsd
13147                              <1>       ;
13148 000119C7 8B35[60030300]      <1>       mov    esi, [u.usp] ; esp (system stack pointer,
13149                              <1>                           ;      points to user registers)
13150 000119CD 8B0D[5C030300]      <1>       mov    ecx, [u.sp]  ; return address from the system call
13151                              <1>                           ; (for IRET)
13152                              <1>                           ; [u.sp] -> EIP (user)
13153                              <1>                           ; [u.sp+4]-> CS (user)
13154                              <1>                           ; [u.sp+8] -> EFLAGS (user)
13155                              <1>                           ; [u.sp+12] -> ESP (user)
13156                              <1>                           ; [u.sp+16] -> SS (user)
13157 000119D3 29F1                <1>       sub    ecx, esi     ; required space for user registers
13158 000119D5 83C114              <1>       add    ecx, 20           ; +5 dwords to return from system call
13159                              <1>                           ; (for IRET)
13160 000119D8 C1E902              <1>       shr    ecx, 2
13161 000119DB F3A5                <1>       rep    movsd
13162 000119DD C3                  <1>       retn
13163                              <1>
13164                              <1> rswap:  ; < swap in, swap from disk >
13165                              <1>       ; 28/02/2017 (frstor)
13166                              <1>       ; 15/01/2017
13167                              <1>       ; 14/01/2017
13168                              <1>       ; 21/05/2016
13169                              <1>       ; 03/05/2016
13170                              <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
13171                              <1>       ; 09/05/2015 - 15/09/2015 (Retro UNIX 386 v1)
13172                              <1>       ; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)
13173                              <1>       ; 'rswap' reads a process whose number is in r1,
```

```
13174                                  <1>       ; from disk into core.
13175                                  <1>       ;
13176                                  <1>       ; Retro UNIX 386 v1 modification ->
13177                                  <1>       ;       User (u) structure content and the user's register content
13178                                  <1>       ;     will be restored from process's/user's UPAGE (a page for
13179                                  <1>       ;     saving 'u' structure and user registers for task switching).
13180                                  <1>       ;     u.usp - points to kernel stack address which contains
13181                                  <1>       ;             user's registers while entering system call.
13182                                  <1>       ;     u.sp  - points to kernel stack address
13183                                  <1>       ;             to return from system call -for IRET-.
13184                                  <1>       ;     [u.usp]+32+16 = [u.sp]
13185                                  <1>       ;     [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,
13186                                  <1>       ;             edx, ecx, eax, gs, fs, es, ds, -> [u.sp].
13187                                  <1>       ;
13188                                  <1>       ; RETRO UNIX 8086 v1 modification ->
13189                                  <1>       ;       'swap to disk' is replaced with 'change running segment'
13190                                  <1>       ;     according to 8086 cpu (x86 real mode) architecture.
13191                                  <1>       ;     pdp-11 was using 64KB uniform memory while IBM PC
13192                                  <1>       ;     compatibles was using 1MB segmented memory
13193                                  <1>       ;     in 8086/8088 times.
13194                                  <1>       ;
13195                                  <1>       ; INPUTS ->
13196                                  <1>       ;    r1 - process number of process to be read in
13197                                  <1>       ;    p.break - negative of word count of process
13198                                  <1>       ;    p.dska - disk address of the process
13199                                  <1>       ;    u.emt - determines handling of emt's
13200                                  <1>       ;    u.ilgins - determines handling of illegal instructions
13201                                  <1>       ; OUTPUTS ->
13202                                  <1>       ;    8 = (u.ilgins)
13203                                  <1>       ;    24 = (u.emt)
13204                                  <1>       ;    swp - bit 10 is set to indicate read
13205                                  <1>       ;             (bit 15=0 when reading is done)
13206                                  <1>       ;    swp+2 - disk block address
13207                                  <1>       ;    swp+4 - negative word count
13208                                  <1>       ;      ((swp+6 - address of user structure))
13209                                  <1>       ;
13210                                  <1>       ; RETRO UNIX 8086 v1 input/output:
13211                                  <1>       ;
13212                                  <1>       ; INPUTS ->
13213                                  <1>       ;    AL      - new process number (to be swapped in)
13214                                  <1>       ; OUTPUTS ->
13215                                  <1>       ;    none
13216                                  <1>       ;
13217                                  <1>       ;   ((Modified registers: EAX, ECX, ESI, EDI, ESP))
13218                                  <1>       ;
13219                                  <1>       ; Retro UNIX 386 v1 - modification ! 14/05/2015
13220 000119DE 89C6                   <1>       mov    esi, eax  ; process's user (u) structure page addr
13221 000119E0 B938000000             <1>       mov    ecx, (U_SIZE + 3) / 4
13222 000119E5 BF[5C030300]           <1>       mov    edi, user ; active user (u) structure
13223 000119EA F3A5                   <1>       rep    movsd
13224 000119EC 58                     <1>       pop    eax    ; 'rswap' return address
13225                                  <1>       ;
13226                                  <1>       ;cli
13227 000119ED 8B3D[60030300]         <1>       mov    edi, [u.usp] ; esp (system stack pointer,
13228                                  <1>                           ;     points to user registers)
13229 000119F3 89FC                   <1>       mov    esp, edi     ; 14/01/2017
13230 000119F5 8B0D[5C030300]         <1>       mov    ecx, [u.sp]  ; return address from the system call
13231                                  <1>                           ; (for IRET)
13232                                  <1>                           ; [u.sp] -> EIP (user)
13233                                  <1>                           ; [u.sp+4]-> CS (user)
13234                                  <1>                           ; [u.sp+8] -> EFLAGS (user)
13235                                  <1>                           ; [u.sp+12] -> ESP (user)
13236                                  <1>                           ; [u.sp+16] -> SS (user)
13237 000119FB 29F9                   <1>       sub    ecx, edi     ; required space for user registers
13238 000119FD 83C114                 <1>       add    ecx, 20           ; +5 dwords to return from system call
13239                                  <1>                           ; (for IRET)
13240 00011A00 C1E902                 <1>       shr    ecx, 2
13241 00011A03 F3A5                   <1>       rep    movsd
13242                                  <1>       ;mov   esp, [u.usp] ; 15/09/2015
13243                                  <1>       ;sti
13244                                  <1>       ; 28/02/2017
13245                                  <1>       ;cmp   byte [multi_tasking], 0  ; Musti tasking mode ?
13246                                  <1>       ;jna   short rswp_retn
13247 00011A05 803D[DA030300]00       <1>       cmp    byte [u.fpsave], 0
13248 00011A0C 7606                   <1>       jna    short rswp_retn
13249 00011A0E DD25[DC030300]         <1>       frstor [u.fpregs] ; restore floating point regs (94 bytes)
13250                                  <1> rswp_retn:
13251 00011A14 50                     <1>       push   eax    ; 'rswap' return address
13252 00011A15 C3                     <1>       retn
13253                                  <1>
13254                                  <1> putlu:
13255                                  <1>       ; 20/05/2016
13256                                  <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
13257                                  <1>       ; 10/05/2015 - 12/09/2015 (Retro UNIX 386 v1)
13258                                  <1>       ; 15/04/2013 - 23/02/2014 (Retro UNIX 8086 v1)
13259                                  <1>       ; 'putlu' is called with a process number in r1 and a pointer
13260                                  <1>       ; to lowest priority Q (runq+4) in r2. A link is created from
13261                                  <1>       ; the last process on the queue to process in r1 by putting
13262                                  <1>       ; the process number in r1 into the last process's link.
13263                                  <1>       ;
13264                                  <1>       ; INPUTS ->
13265                                  <1>       ;    r1 - user process number
13266                                  <1>       ;    r2 - points to lowest priority queue
13267                                  <1>       ;    p.dska - disk address of the process
13268                                  <1>       ;    u.emt - determines handling of emt's
13269                                  <1>       ;    u.ilgins - determines handling of illegal instructions
13270                                  <1>       ; OUTPUTS ->
13271                                  <1>       ;    r3 - process number of last process on the queue upon
13272                                  <1>       ;       entering putlu
13273                                  <1>       ;    p.link-1 + r3 - process number in r1
13274                                  <1>       ;    r2 - points to lowest priority queue
13275                                  <1>       ;
13276                                  <1>       ; ((Modified registers: EDX, EBX))
13277                                  <1>       ;
13278                                  <1>       ; / r1 = user process no.; r2 points to lowest priority queue
```

```
13279                                  <1>
13280                                  <1>       ; EBX = r2
13281                                  <1>       ; EAX = r1 (AL=r1b)
13282                                  <1>
13283                                  <1>       ; 20/05/2016
13284                                  <1>       ; AL = process number (1 to 16) // Retro UNIX 8086, 386 v1 //
13285                                  <1>       ;     (max. 16 processes available for current kernel version)
13286                                  <1>       ; EBX = run queue address ; 20/05/2016 (TRDOS 386)
13287                                  <1>             ; which is one of following addresses:
13288                                  <1>             ;  1) 'runq_event' high priority run queue
13289                                  <1>             ;  2) 'runq_normal' normal/regular priority run queue
13290                                  <1>             ;  3) 'runq_background' low priority run queue
13291                                  <1>
13292                                  <1>       ;mov   ebx, runq
13293 00011A16 0FB613                 <1>       movzx       edx, byte [ebx]
13294 00011A19 43                     <1>       inc    ebx
13295 00011A1A 20D2                   <1>       and    dl, dl
13296                                  <1>             ; tstb (r2)+ / is queue empty?
13297 00011A1C 740A                   <1>             jz     short putlu_1
13298                                  <1>             ; beq 1f / yes, branch
13299 00011A1E 8A13                   <1>       mov    dl, [ebx] ; 12/09/2015
13300                                  <1>             ; movb (r2),r3 / no, save the "last user" process number
13301                                  <1>                         ; / in r3
13302 00011A20 8882[9F000300]         <1>             mov    [edx+p.link-1], al
13303                                  <1>             ; movb r1,p.link-1(r3) / put pointer to user on
13304                                  <1>                         ; / "last users" link
13305 00011A26 EB03                   <1>       jmp    short putlu_2
13306                                  <1>             ; br 2f /
13307                                  <1> putlu_1: ; 1:
13308 00011A28 8843FF                 <1>       mov    [ebx-1], al
13309                                  <1>                     ; movb r1,-1(r2) / user is only user;
13310                                  <1>                         ; / put process no. at beginning and at end
13311                                  <1> putlu_2: ; 2:
13312 00011A2B 8803                   <1>       mov    [ebx], al
13313                                  <1>                     ; movb r1,(r2) / user process in r1 is now the last entry
13314                                  <1>                         ; / on the queue
13315 00011A2D 88C2                   <1>       mov    dl, al
13316 00011A2F 88B2[9F000300]         <1>         mov     [edx+p.link-1], dh ; 0
13317                                  <1>             ; dec r2 / restore r2
13318 00011A35 C3                     <1>         retn
13319                                  <1>             ; rts r0
13320                                  <1>
13321                                  <1> sysver:
13322                                  <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
13323 00011A36 C705[64030300]0002-    <1>       mov    dword [u.r0], 200h ; AH = major version, AL = minor version
13323 00011A3E 0000                   <1>
13324 00011A40 E9B7BEFFFF             <1>       jmp    sysret
13325                                  <1>
13326                                  <1>
13327                                  <1> syspri: ; change running priority (of the process)
13328                                  <1>       ; 21/05/2016
13329                                  <1>       ; 20/05/2026 - TRDOS 386 (TRDOS v2.0)
13330                                  <1>       ; INPUT ->
13331                                  <1>       ;     BL = priority level
13332                                  <1>       ;        0 = low running priority (running on background)
13333                                  <1>       ;        1 = normal/regular priority (running as regular)
13334                                  <1>       ;        2 = high/event priority (running for event)
13335                                  <1>       ;        >2 = invalid, it will accepted as 2 (event)
13336                                  <1>       ;        0FFh = get/return current running priority only
13337                                  <1>       ; OUTPUT ->
13338                                  <1>       ;     * if current [u.pri] < 2
13339                                  <1>       ;       if BL input < 0FFh ->
13340                                  <1>       ;          [u.pri] is updated as in BL input (0,1,2)
13341                                  <1>       ;       if BL input = 0FFh -> AL = [u.pri] (current)
13342                                  <1>       ;
13343                                  <1>       ;     * if current [u.pri] = 2
13344                                  <1>       ;       if BL input < 0FFh -> cf = 1 & AL = 2
13345                                  <1>       ;       if BL input = 0FFh -> cf = 0 & AL = 2
13346                                  <1>       ;
13347                                  <1>       ;     NOTE:
13348                                  <1>       ;     If [u.pri] = 2, it can not be changed to 1 or 0;
13349                                  <1>       ;     because, run queue of the running process is unspecified
13350                                  <1>       ;     at this      stage. Process might be started by a timer event
13351                                  <1>       ;     or priority might be changed to high by previous
13352                                  <1>       ;     'syspri' system     call. In both cases, the process is in
13353                                  <1>       ;     'runq_normal' or 'runq_background' queue.
13354                                  <1>       ;     As result of this fact, when the [u.quant] time quantum
13355                                  <1>       ;     of the process is elapsed or 'sysrele' system call is
13356                                  <1>       ;     instructed by the process, 'tswap' ('tswitch') procedure
13357                                  <1>       ;     will be called (to 'swap' or 'switch' out the procedure)
13358                                  <1>       ;     and it will not call 'putlu' to add the (stopping)
13359                                  <1>       ;     process to relevant run queue when [u.pri] = 2.
13360                                  <1>       ;     (Otherwise, it would be possible to add process to
13361                                  <1>       ;     a run queue while it is already in a run queue, wrongly.)
13362                                  <1>       ;
13363                                  <1>       ;     If [u.pri]< 2, 'tswap/tswitch' procedure will call
13364                                  <1>       ;     'putlu' to add process to relevant run queue
13365                                  <1>       ;     according to [u.pri] value. ('runq_normal' for 1,
13366                                  <1>       ;     'runq_background' for 0).
13367                                  <1>       ;
13368                                  <1>       ;     If BL input >= 2 and < 0FFh while [u.pri] < 2,
13369                                  <1>       ;     process will be added to 'runq_normal' queue and
13370                                  <1>       ;     [u.pri] will be set to 2. (in 'syspri' system call)
13371                                  <1>       ;
13372                                  <1>
13373 00011A45 29C0                   <1>       sub    eax, eax ; 0
13374 00011A47 A3[C8030300]           <1>       mov    [u.error], eax
13375                                  <1>
13376 00011A4C A0[A9030300]           <1>       mov    al, [u.pri]
13377 00011A51 A3[64030300]           <1>       mov    [u.r0], eax
13378                                  <1>
13379 00011A56 FEC3                   <1>       inc    bl
13380 00011A58 0F849EBEFFFF           <1>       jz     sysret ; 0FFh -> 0, get priority level
13381                                  <1>
13382 00011A5E 3C02                   <1>       cmp    al, 2
```

```
13383 00011A60 0F8376BEFFFF        <1>       jnb    error ; CF = 1 & AL = 2 (& last error = 0)
13384                              <1>
13385 00011A66 FECB                <1>       dec    bl
13386 00011A68 80FB02              <1>       cmp    bl, 2
13387 00011A6B 7602                <1>       jna    short syspri_1
13388 00011A6D B302                <1>       mov    bl, 2
13389                              <1> syspri_1:
13390 00011A6F 881D[A9030300]      <1>       mov    [u.pri], bl
13391 00011A75 80FB02              <1>       cmp    bl, 2
13392 00011A78 0F827EBEFFFF        <1>         jb      sysret
13393                              <1>
13394                              <1>       ; here...
13395                              <1>       ; Priority of current process has been changed to high
13396                              <1>       ; ('run for event') but current process will be added to
13397                              <1>       ; 'run as normal' queue. ('run for event' high priority
13398                              <1>       ; queue is under control of timer -& RTC- interrupt only!)
13399                              <1>       ;
13400                              <1>       ; (Otherwise, process can fall into black hole!
13401                              <1>       ; e.g. if it is not in waiting list and it has not got
13402                              <1>       ; a timer event and it is not in a run queue!
13403                              <1>       ; Because, when [u.pri] is 2, 'tswap/tswitch' will not
13404                              <1>       ; add the stopping process to a run queue.)
13405                              <1>
13406 00011A7E A0[B3030300]        <1>       mov    al, [u.uno]
13407 00011A83 BB[54030300]        <1>       mov    ebx, runq_normal ; normal priority !
13408                              <1>                               ; [u.pri] is set to high
13409                              <1>                               ; but 'runq_event' queue is set
13410                              <1>                               ; only by the kernel's timer
13411                              <1>                               ; event function (timer interrupt).
13412 00011A88 E889FFFFFF          <1>       call   putlu
13413 00011A8D E96ABEFFFF          <1>       jmp    sysret
13414                              <1>
13415                              <1> cpass: ; / get next character from user area of core and put it in AL (r1)
13416                              <1>       ; 02/05/2016 - TRDOS 386 (TRDOS v2.0)
13417                              <1>       ; 19/05/2015 - 18/10/2015 (Retro UNIX 386 v1)
13418                              <1>       ; 14/08/2013 - 20/09/2013 (Retro UNIX 8086 v1)
13419                              <1>       ; INPUTS ->
13420                              <1>       ;     [u.base] = virtual address in user area
13421                              <1>       ;     [u.count] = byte count (max.)
13422                              <1>       ;     [u.pcount] = byte count in page (0 = reset)
13423                              <1>       ; OUTPUTS ->
13424                              <1>       ;     AL = the character which is pointed by [u.base]
13425                              <1>       ;     zf = 1 -> transfer count has been completed
13426                              <1>         ;
13427                              <1>       ; ((Modified registers:  EAX, EDX, ECX))
13428                              <1>       ;
13429 00011A92 833D[88030300]00    <1>       cmp    dword [u.count], 0  ; have all the characters been transferred
13430                              <1>                                  ; i.e., u.count, # of chars. left
13431 00011A99 763F                <1>       jna    short cpass_3    ; to be transferred = 0?) yes, branch
13432 00011A9B FF0D[88030300]      <1>       dec    dword [u.count]         ; no, decrement u.count
13433                              <1>         ; 19/05/2015
13434                              <1>       ;(Retro UNIX 386 v1 - translation from user's virtual address
13435                              <1>       ;                     to physical address
13436 00011AA1 66833D[C4030300]00  <1>       cmp    word [u.pcount], 0 ; byte count in page = 0 (initial value)
13437                              <1>                               ; 1-4095 --> use previous physical base address
13438                              <1>                               ; in [u.pbase]
13439 00011AA9 770E                <1>       ja     short cpass_1
13440 00011AAB 833D[BC030300]00    <1>       cmp     dword [u.ppgdir], 0 ; is the caller os kernel
13441 00011AB2 7427                <1>         je      short cpass_k       ; (sysexec, '/etc/init') ?  (MainProg)
13442 00011AB4 E858FDFFFF          <1>       call   trans_addr_r
13443                              <1> cpass_1:
13444 00011AB9 66FF0D[C4030300]    <1>       dec    word [u.pcount]
13445                              <1> cpass_2:
13446 00011AC0 8B15[C0030300]      <1>       mov    edx, [u.pbase]
13447 00011AC6 8A02                <1>       mov    al, [edx]    ; take the character pointed to
13448                              <1>                             ; by u.base and put it in r1
13449 00011AC8 FF05[8C030300]      <1>       inc    dword [u.nread] ; increment no. of bytes transferred
13450 00011ACE FF05[84030300]      <1>       inc    dword [u.base]  ; increment the buffer address to point to the
13451                              <1>                               ; next byte
13452 00011AD4 FF05[C0030300]      <1>       inc    dword [u.pbase]
13453                              <1> cpass_3:
13454 00011ADA C3                  <1>       retn
13455                              <1> cpass_k:
13456                              <1>       ; 02/07/2015
13457                              <1>       ; The caller is os kernel
13458                              <1>       ; (get sysexec arguments from kernel's memory space)
13459 00011ADB 8B1D[84030300]      <1>       mov    ebx, [u.base]
13460 00011AE1 66C705[C4030300]00- <1>         mov     word [u.pcount], PAGE_SIZE ; 4096
13460 00011AE9 10                  <1>
13461 00011AEA 891D[C0030300]      <1>       mov    [u.pbase], ebx
13462 00011AF0 EBCE                <1>       jmp    short cpass_2
13463                              <1>
13464                              <1> transfer_to_user_buffer: ; fast transfer
13465                              <1>       ; 27/05/2016
13466                              <1>       ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
13467                              <1>       ;
13468                              <1>       ; INPUT ->
13469                              <1>       ;     ESI = source address in system space
13470                              <1>       ;     EDI = user's buffer address
13471                              <1>       ;     ECX = transfer (byte) count
13472                              <1>       ;     [u.pgdir] = user's page directory
13473                              <1>       ; OUTPUT ->
13474                              <1>       ;     ECX = actual transfer count
13475                              <1>       ;     cf = 1 -> error
13476                              <1>       ;     [u.count] = remain byte count
13477                              <1>       ;
13478                              <1>       ; Modified registers: eax, ecx
13479                              <1>       ;
13480                              <1>
13481 00011AF2 21C9                <1>       and    ecx, ecx
13482 00011AF4 743B                <1>       jz     short ttub_4
13483                              <1>
13484 00011AF6 890D[88030300]      <1>       mov    [u.count], ecx
13485                              <1>
13486 00011AFC 57                  <1>       push   edi
```

```
13487 00011AFD 56                <1>       push   esi
13488 00011AFE 53                <1>       push   ebx
13489 00011AFF 52                <1>       push   edx
13490 00011B00 51                <1>       push   ecx
13491                            <1>
13492 00011B01 89FB              <1>       mov    ebx, edi
13493 00011B03 81C300004000      <1>       add    ebx, CORE ; 27/05/2016
13494                            <1> ttub_1:
13495                            <1>       ; ebx = virtual (linear) address
13496                            <1>       ; [u.pgdir] = user's page directory
13497 00011B09 E8E047FFFF        <1>             call   get_physical_addr_x ; get physical address
13498 00011B0E 7222              <1>       jc     short ttub_5
13499                            <1>       ; eax = physical address
13500                            <1>       ; ecx = remain byte count in page (1-4096)
13501 00011B10 89C7              <1>       mov    edi, eax
13502 00011B12 A1[88030300]      <1>       mov    eax, [u.count]
13503 00011B17 39C1              <1>       cmp    ecx, eax
13504 00011B19 7602              <1>       jna    short ttub_2
13505 00011B1B 89C1              <1>       mov    ecx, eax
13506                            <1> ttub_2:
13507 00011B1D 29C8              <1>       sub    eax, ecx
13508 00011B1F 01CB              <1>       add    ebx, ecx
13509 00011B21 F3A4              <1>       rep    movsb
13510 00011B23 A3[88030300]      <1>       mov    [u.count], eax
13511 00011B28 09C0              <1>       or     eax, eax
13512 00011B2A 75DD              <1>       jnz    short ttub_1
13513                            <1> ttub_retn:
13514                            <1> tfub_retn:
13515 00011B2C 59                <1>       pop    ecx ; transfer count = actual transfer count
13516                            <1> ttub_3:
13517 00011B2D 5A                <1>       pop    edx
13518 00011B2E 5B                <1>       pop    ebx
13519 00011B2F 5E                <1>       pop    esi
13520 00011B30 5F                <1>       pop    edi
13521                            <1> ttub_4:
13522 00011B31 C3                <1>       retn
13523                            <1> ttub_5:
13524 00011B32 59                <1>       pop    ecx
13525 00011B33 2B0D[88030300]    <1>       sub    ecx, [u.count] ; actual transfer count
13526 00011B39 F9                <1>       stc
13527 00011B3A EBF1              <1>       jmp    short ttub_3
13528                            <1>
13529                            <1> transfer_from_user_buffer: ; fast transfer
13530                            <1>       ; 27/05/2016
13531                            <1>       ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
13532                            <1>       ;
13533                            <1>       ; INPUT ->
13534                            <1>       ;      ESI = user's buffer address
13535                            <1>       ;      EDI = destination address in system space
13536                            <1>       ;      ECX = transfer (byte) count
13537                            <1>       ;      [u.pgdir] = user's page directory
13538                            <1>       ; OUTPUT ->
13539                            <1>       ;      ecx = actual transfer count
13540                            <1>       ;      cf = 1 -> error
13541                            <1>       ;      [u.count] = remain byte count
13542                            <1>       ;
13543                            <1>       ; Modified registers: eax, ecx
13544                            <1>       ;
13545                            <1>
13546 00011B3C 21C9              <1>       and    ecx, ecx
13547                            <1>       ;jz    short tfub_4
13548 00011B3E 74F1              <1>       jz     short ttub_4
13549                            <1>
13550 00011B40 890D[88030300]    <1>       mov    [u.count], ecx
13551                            <1>
13552 00011B46 57                <1>       push   edi
13553 00011B47 56                <1>       push   esi
13554 00011B48 53                <1>       push   ebx
13555 00011B49 52                <1>       push   edx
13556 00011B4A 51                <1>       push   ecx
13557                            <1>
13558 00011B4B 89F3              <1>       mov    ebx, esi
13559 00011B4D 81C300004000      <1>       add    ebx, CORE ; 27/05/2016
13560                            <1> tfub_1:
13561                            <1>       ; ebx = virtual (linear) address
13562                            <1>       ; [u.pgdir] = user's page directory
13563 00011B53 E89647FFFF        <1>             call   get_physical_addr_x ; get physical address
13564                            <1>       ;jc    short tfub_5
13565 00011B58 72D8              <1>       jc     short ttub_5
13566                            <1>       ; eax = physical address
13567                            <1>       ; ecx = remain byte count in page (1-4096)
13568 00011B5A 89C6              <1>       mov    esi, eax
13569 00011B5C A1[88030300]      <1>       mov    eax, [u.count]
13570 00011B61 39C1              <1>       cmp    ecx, eax
13571 00011B63 7602              <1>       jna    short tfub_2
13572 00011B65 89C1              <1>       mov    ecx, eax
13573                            <1> tfub_2:
13574 00011B67 29C8              <1>       sub    eax, ecx
13575 00011B69 01CB              <1>       add    ebx, ecx
13576 00011B6B F3A4              <1>       rep    movsb
13577 00011B6D A3[88030300]      <1>       mov    [u.count], eax
13578 00011B72 09C0              <1>       or     eax, eax
13579 00011B74 75DD              <1>       jnz    short tfub_1
13580                            <1>
13581 00011B76 EBB4              <1>       jmp    short tfub_retn
13582                            <1>
13583                            <1> ;tfub_retn:
13584                            <1> ;     pop    ecx ; transfer count = actual transfer count
13585                            <1> ;tfub_3:
13586                            <1> ;     pop    edx
13587                            <1> ;     pop    ebx
13588                            <1> ;     pop    esi
13589                            <1> ;     pop    edi
13590                            <1> ;tfub_4:
13591                            <1> ;     retn
```

```
13592                                  <1> ;tfub_5:
13593                                  <1> ;       pop     ecx
13594                                  <1> ;       sub     ecx, [u.count] ; actual transfer count
13595                                  <1> ;       stc
13596                                  <1> ;       jmp     short tfub_3
13597                                  <1>
13598                                  <1> sysfff: ; <Find First File>
13599                                  <1>         ; 17/10/2016
13600                                  <1>         ; 16/10/2016
13601                                  <1>         ; 15/10/2016 TRDOS 386 (TRDOS v2.0) feature only !
13602                                  <1>         ;           -derived from TRDOS v1.0, INT_21H.ASM-
13603                                  <1>         ;            ("loc_INT21h_find_first_file")
13604                                  <1>         ; TRDOS 8086 (v1.0)
13605                                  <1>           ;    07/08/2011
13606                                  <1>           ;    Find First File
13607                                  <1>         ;      INPUT:
13608                                  <1>           ;        CX= Attributes
13609                                  <1>           ;        DS:DX= Pointer to filename
13610                                  <1>         ;      MSDOS OUTPUT:
13611                                  <1>         ;          DTA: (Default address: PSP offset 80h)
13612                                  <1>         ;          Offset  Descrription
13613                                  <1>         ;          0      Reserved for use find next file
13614                                  <1>         ;          21     Attribute of file found
13615                                  <1>         ;          22     Time stamp of file
13616                                  <1>         ;          24     Date stamp of file
13617                                  <1>         ;          26     File size in bytes
13618                                  <1>         ;          30     Filename and extension (zero terminated)
13619                                  <1>         ;      If cf = 1:
13620                                  <1>         ;          Error Codes: (in AX)
13621                                  <1>         ;             2 - File not found
13622                                  <1>         ;             18 - No more files
13623                                  <1>           ;
13624                                  <1>         ; TRDOS 386 (v2.0)
13625                                  <1>         ; 15/10/2016
13626                                  <1>         ;
13627                                  <1>           ; INPUT ->
13628                                  <1>           ;       CL = File attributes
13629                                  <1>         ;            bit 0 (1) - Read only file (R)
13630                                  <1>         ;             bit 1 (1) - Hidden file (H)
13631                                  <1>           ;            bit 2 (1) - System file (R)
13632                                  <1>         ;             bit 3 (1) - Volume label/name (V)
13633                                  <1>           ;            bit 4 (1) - Subdirectory (D)
13634                                  <1>         ;            bit 5 (1) - File has been archived (A)
13635                                  <1>         ;         CH = 0 -> Return basic parameters (24 bytes)
13636                                  <1>         ;         CH > 0 -> Return FindFile structure/table (128 bytes)
13637                                  <1>           ;          EBX = Pointer to filename (ASCIIZ) -path-
13638                                  <1>         ;         EDX = File parameters buffer address
13639                                  <1>         ;            (buffer size = 24 bytes if CH input = 0)
13640                                  <1>         ;            (buffer size = 128 bytes if CH input > 0)
13641                                  <1>         ;
13642                                  <1>         ; OUTPUT ->
13643                                  <1>         ;         EAX = 0 if CH input > 0
13644                                  <1>         ;         EAX = First cluster number of file if CH input = 0
13645                                  <1>         ;         EDX = File parameters table/structure address
13646                                  <1>         ;         Basic Parameters:
13647                                  <1>         ;            Offset  Description
13648                                  <1>         ;            ------ ---------------
13649                                  <1>         ;            0      File Attributes
13650                                  <1>         ;            1      Ambiguous filename chars are used sign
13651                                  <1>         ;                   (0 = filename fits exactly with request)
13652                                  <1>         ;                   (>0 = ambiguous filename chars are used)
13653                                  <1>         ;            2      Time stamp of file
13654                                  <1>         ;            4      Date stamp of file
13655                                  <1>         ;            6      File size in bytes
13656                                  <1>         ;            10     Short Filename (ASCIIZ, max. 13 bytes)
13657                                  <1>         ;            23     Longname Length (1-255) if existing
13658                                  <1>         ;
13659                                  <1>         ;          cf = 1 -> Error code in AL
13660                                  <1>         ;
13661                                  <1>         ; Modified Registers: EAX (at the return of system call)
13662                                  <1>         ;
13663                                  <1>         ; TR-DOS FindFile (FFF) Structure (128 bytes):
13664                                  <1>         ; 09/10/2011 (DIR.ASM) - 10/02/2016 (trdoskx.s)
13665                                  <1>         ;
13666                                  <1>         ; Offset     Parameter           Size
13667                                  <1>         ; ------     ------------------  --------
13668                                  <1>         ; 0          FindFile_Drv        1 byte
13669                                  <1>         ; 1          FindFile_Directory  65 bytes
13670                                  <1>         ; 66         FindFile_Name       13 bytes
13671                                  <1>         ; 79         FindFile_LongNameEntryLength 1 byte
13672                                  <1>         ;Above 80 bytes form
13673                                  <1>         ;TR-DOS Source/Destination File FullName Format/Structure
13674                                  <1>         ; 80         FindFile_AttributesMask 1 word
13675                                  <1>         ; 82         FindFile_DirEntry   32 bytes (*)
13676                                  <1>         ; 114        FindFile_DirFirstCluster 1 double word
13677                                  <1>         ; 118        FindFile_DirCluster 1 double word
13678                                  <1>         ; 122        FindFile_DirEntryNumber 1 word
13679                                  <1>         ; 124        FindFile_MatchCounter     1 word
13680                                  <1>         ; 126        FindFile_Reserved   1 word
13681                                  <1>         ; (*) MS-DOS, FAT 12-16-32 classic directory entry (32 bytes)
13682                                  <1>
13683                                  <1>         ;mov   [u.namep], ebx
13684                                  <1>         ; 16/10/2016
13685 00011B78 8915[70960100]        <1>         mov    [FFF_UBuffer], edx
13686 00011B7E 66890D[75960100]      <1>         mov    [FFF_Attrib], cx ; [FFF_RType] = ch
13687                                  <1>                    ; Attributes in CL, return data type in CH
13688 00011B85 89DE                  <1>         mov    esi, ebx
13689                                  <1>         ; file name is forced, change directory as temporary
13690                                  <1>         ;mov   ax, 1
13691                                  <1>         ;mov   [FFF_Valid], ah ; 0 ; reset ; 17/10/2016
13692                                  <1>         ;call  set_working_path
13693 00011B87 E8E2130000            <1>         call   set_working_path_x ; 17/10/2016
13694 00011B8C 731D                  <1>         jnc    short sysfff_0
13695                                  <1>
13696 00011B8E 21C0                  <1>         and    eax, eax  ; 0 -> Bad Path!
```

```
13697 00011B90 7505               <1>       jnz    short sysfff_err
13698                             <1>
13699                             <1>       ; eax = 0
13700 00011B92 B80C000000         <1>       mov    eax, ERR_DIR_NOT_FOUND ; Directory not found !
13701                             <1> sysfff_err:
13702 00011B97 A3[64030300]       <1>       mov    [u.r0], eax
13703 00011B9C A3[C8030300]       <1>       mov    [u.error], eax
13704 00011BA1 E89D140000         <1>         call       reset_working_path
13705 00011BA6 E931BDFFFF         <1>       jmp    error
13706                             <1>
13707                             <1> sysfff_0:
13708                             <1>       ;sub   ah, ah ; ah = 0
13709 00011BAB 8A0424             <1>       mov    al, [esp]
13710 00011BAE 08C0               <1>       or     al, al
13711 00011BB0 7412               <1>       jz     short sysfff_2
13712 00011BB2 B410               <1>       mov    ah, 10h
13713 00011BB4 A808               <1>       test   al, 08h
13714 00011BB6 7503               <1>       jnz    short sysfff_1
13715 00011BB8 80CC08             <1>       or     ah, 08h
13716                             <1> sysfff_1:
13717 00011BBB 2410               <1>       and    al, 10h ; Directory
13718 00011BBD 7405               <1>       jz     short sysfff_2
13719 00011BBF 80E408             <1>       and    ah, 08h
13720 00011BC2 30C0               <1>       xor    al, al ; When a directory is searched,
13721                             <1>                      ; filename will be returned even if
13722                             <1>                      ; it is not a directory!
13723                             <1>                      ; Because: (in order to prevent
13724                             <1>                      ; creating a dir with existing file name)
13725                             <1>                      ; Dir and file names must not be same!
13726                             <1>                      ; (return attribute must be checked)
13727                             <1> sysfff_2:
13728                             <1>       ; AX = Attributes mask
13729                             <1>             ; AL = AND mask (result must be equal to AL)
13730                             <1>             ; AH = Negative AND mask (result must be ZERO)
13731                             <1>       ; ESI = FindFile_Name address
13732                             <1>
13733 00011BC4 E8B878FFFF         <1>       call   find_first_file
13734 00011BC9 72CC               <1>       jc     short sysfff_err ; eax = 2 (File not found !)
13735                             <1>
13736                             <1>       ; ESI = Directory Entry (FindFile_DirEntry) Location
13737                             <1>       ; EDI = Directory Buffer Directory Entry Location
13738                             <1>       ; EAX = File Size
13739                             <1>       ;  BL = Attributes of The File/Directory
13740                             <1>       ;  BH = Long Name Yes/No Status (>0 is YES)
13741                             <1>       ;  DX > 0 : Ambiguous filename chars are used
13742                             <1>
13743                             <1> sysfff_3:
13744                             <1>       ; 16/10/2016
13745 00011BCB 668B0D[75960100]   <1>       mov    cx, [FFF_Attrib]
13746                             <1>       ; Attribs in CL, return data type in CH
13747                             <1>
13748                             <1>       ;or    cl, cl
13749                             <1>       ;jz    short sysfff_4 ; 0 = No filter
13750 00011BD2 80F1FF             <1>       xor    cl, 0FFh
13751 00011BD5 20D9               <1>       and    cl, bl
13752 00011BD7 7409               <1>       jz     short sysfff_4
13753                             <1>
13754                             <1>       ;mov   eax, 2 ; 'file not found !' error
13755                             <1>       ;jmp   short sysfff_err_1
13756                             <1>
13757                             <1>       ; 16/10/2016
13758 00011BD9 E85279FFFF         <1>       call   find_next_file
13759 00011BDE 72B7               <1>       jc     short sysfff_err ; eax = 12 (no more files !)
13760 00011BE0 EBE9               <1>       jmp    short sysfff_3
13761                             <1>
13762                             <1> sysfff_4:
13763 00011BE2 20ED               <1>       and    ch, ch ; [FFF_RType]
13764 00011BE4 7412               <1>       jz     short sysfff_5
13765 00011BE6 B980000000         <1>       mov    ecx, 128 ; ; transfer length
13766 00011BEB 880D[74960100]     <1>       mov    [FFF_Valid], cl
13767                             <1> sysfnf_11:
13768 00011BF1 BE[26930100]       <1>       mov    esi, FindFile_Drv
13769 00011BF6 EB44               <1>       jmp    short sysfff_6
13770                             <1> sysfff_5:
13771                             <1>       ;mov   esi, FindFile_DirEntry
13772 00011BF8 B918000000         <1>       mov    ecx, 24  ; transfer length
13773 00011BFD 880D[74960100]     <1>       mov    [FFF_Valid], cl
13774                             <1> sysfnf_12:
13775 00011C03 BF[309B0100]       <1>       mov    edi, DTA ; FFF data transfer address
13776                             <1>       ;mov   al, [esi+DirEntry_Attr] ; 11
13777 00011C08 88D8               <1>       mov    al, bl ; File/Dir Attributes
13778 00011C0A 887F17             <1>       mov    [edi+23], bh ; Longname length (0= none)
13779 00011C0D AA                 <1>       stosb
13780 00011C0E 88D0               <1>       mov    al, dl ; DL is for '?'
13781 00011C10 00F0               <1>       add    al, dh ; DH is for '*'
13782                             <1>       ; AL > 0 if ambiguous file name wildcards are used
13783 00011C12 AA                 <1>       stosb
13784 00011C13 8B4616             <1>       mov    eax, [esi+DirEntry_WrtTime] ; 22
13785 00011C16 AB                 <1>         stosd      ; DirEntry_WrtTime & DirEntry_WrtDate
13786 00011C17 8B461C             <1>         mov  eax, [esi+DirEntry_FileSize] ; 28
13787 00011C1A AB                 <1>         stosd
13788 00011C1B 668B4614           <1>       mov    ax, [esi+DirEntry_FstClusHI] ; 20
13789 00011C1F 66C1E010           <1>       shl    ax, 16
13790 00011C23 668B461A           <1>       mov    ax, [esi+DirEntry_FstClusLO] ; 26
13791 00011C27 A3[64030300]       <1>       mov    [u.r0], eax ; First Cluster
13792                             <1>
13793                             <1>         ;mov esi, FindFile_DirEntry
13794 00011C2C E84F140000         <1>       call   get_file_name
13795                             <1>
13796 00011C31 8A0D[74960100]     <1>       mov    cl, [FFF_Valid]
13797 00011C37 BE[309B0100]       <1>              mov    esi, DTA ; FFF data transfer address
13798                             <1> sysfff_6:
13799 00011C3C 8B3D[70960100]     <1>       mov    edi, [FFF_UBuffer] ; user's buffer address (edx)
13800 00011C42 E8ABFEFFFF         <1>       call   transfer_to_user_buffer
13801                             <1>
```

```
13802 00011C47 890D[64030300]      <1>       mov    [u.r0], ecx ; actual transfer count
13803 00011C4D E8F1130000          <1>         call       reset_working_path
13804 00011C52 E9A5BCFFFF          <1>       jmp    sysret
13805                              <1>
13806                              <1> sysfnf: ; <Find Next File>
13807                              <1>       ; 16/10/2016 TRDOS 386 (TRDOS v2.0) feature only !
13808                              <1>       ;           -derived from TRDOS v1.0, INT_21H.ASM-
13809                              <1>       ;            ("loc_INT21h_find_next_file")
13810                              <1>       ; TRDOS 8086 (v1.0)
13811                              <1>         ;    07/08/2011
13812                              <1>         ;    Find First File
13813                              <1>       ;      INPUT:
13814                              <1>         ;        none
13815                              <1>       ;      MSDOS OUTPUT:
13816                              <1>       ;          DTA: (Default address: PSP offset 80h)
13817                              <1>       ;          Offset  Descrription
13818                              <1>       ;          0      Reserved for use find next file
13819                              <1>       ;          21     Attribute of file found
13820                              <1>       ;          22     Time stamp of file
13821                              <1>       ;          24     Date stamp of file
13822                              <1>       ;          26     File size in bytes
13823                              <1>       ;          30     Filename and extension (zero terminated)
13824                              <1>       ;      If cf = 1:
13825                              <1>       ;          Error Codes: (in AX)
13826                              <1>       ;             18 - No more files
13827                              <1>         ;
13828                              <1>       ; TRDOS 386 (v2.0)
13829                              <1>       ; 16/10/2016
13830                              <1>       ;
13831                              <1>         ; INPUT ->
13832                              <1>             ;         none
13833                              <1>       ; OUTPUT ->
13834                              <1>       ;         EAX = 0 if CH input of 'Find First File' > 0
13835                              <1>       ;         EAX = First cluster number of file
13836                              <1>       ;             if CH input of 'Find First File' = 0
13837                              <1>       ;         EDX = File parameters table/structure address
13838                              <1>       ;
13839                              <1>       ;          cf = 1 -> Error code in AL
13840                              <1>       ;
13841                              <1>       ; Modified Registers: EAX (at the return of system call)
13842                              <1>
13843                              <1>       ;
13844                              <1>       ; Note: If byte [FFF_Valid] = 0
13845                              <1>       ;      'sysfnf' will return with 'no more files' error.
13846                              <1>       ;      If byte [FFF_Valid] = 24
13847                              <1>       ;      'sysfnf' will return with 32 bytes basic parameters
13848                              <1>       ;      at the address which is in EDX.
13849                              <1>       ;      If byte [FFF_Valid] = 128
13850                              <1>       ;      'sysfnf' will return with 128 bytes Find File
13851                              <1>       ;      Structure/Table at the address which is in EDX.
13852                              <1>
13853 00011C57 803D[74960100]00    <1>       cmp    byte [FFF_Valid], 0
13854 00011C5E 7714                <1>       ja     short stsfnf_0
13855                              <1>       ; 'no more files !' error
13856 00011C60 B80C000000          <1>       mov    eax, ERR_NO_MORE_FILES ; 12
13857 00011C65 A3[64030300]        <1>       mov    [u.r0], eax
13858 00011C6A A3[C8030300]        <1>       mov    [u.error], eax
13859 00011C6F E968BCFFFF          <1>       jmp    error
13860                              <1> stsfnf_0:
13861                              <1>       ;cmp   byte [FFF_Valid], 128
13862                              <1>       ;je    short stsfnf_1
13863                              <1>       ;cmp   byte [FFF_Valid], 24
13864                              <1>       ;je    short stsfnf_1
13865                              <1>       ;mov   [FFF_Valid], 24 ; Default
13866                              <1> stsfnf_1:
13867 00011C74 0FB61D[868A0100]    <1>       movzx  ebx, byte [Current_Drv]
13868 00011C7B 66891D[7A960100]    <1>       mov    [SWP_DRV], bx
13869 00011C82 8A15[26930100]      <1>       mov    dl, [FindFile_Drv]
13870 00011C88 38DA                <1>       cmp    dl, bl
13871 00011C8A 750B                <1>       jne    short stsfnf_2
13872 00011C8C 86FB                <1>       xchg   bh, bl
13873 00011C8E BE00010900          <1>       mov    esi, Logical_DOSDisks
13874 00011C93 01DE                <1>       add    esi, ebx
13875 00011C95 EB0D                <1>       jmp    short sysfnf_3
13876                              <1>
13877                              <1> stsfnf_2:
13878 00011C97 FE05[7B960100]      <1>       inc    byte [SWP_DRV_chg]
13879                              <1>
13880 00011C9D E83C64FFFF          <1>       call   change_current_drive
13881 00011CA2 7245                <1>       jc     short sysfnf_err_1 ; read error !
13882                              <1>                                 ; (do not stop, because
13883                              <1>                                 ; we don't have a
13884                              <1>                                 ; 'no more files'
13885                              <1>                                 ; -file not found- error,
13886                              <1>                                 ; next sysfnf system call
13887                              <1>                                 ; may solve the problem,
13888                              <1>                                 ; after re-placing the disk)
13889                              <1> sysfnf_3:
13890 00011CA4 A1[9C930100]        <1>       mov    eax, [FindFile_DirCluster]
13891 00011CA9 21C0                <1>       and    eax, eax
13892 00011CAB 7550                <1>       jnz    short sysfnf_6
13893                              <1>
13894 00011CAD 803D[858A0100]02    <1>       cmp    byte [Current_FATType], 2
13895 00011CB4 772C                <1>       ja     short sysfnf_err_0 ; invalid, we neeed to stop !?
13896 00011CB6 803D[858A0100]01    <1>       cmp    byte [Current_FATType], 1
13897 00011CBD 7223                <1>       jb     short sysfnf_err_0 ; invalid, we neeed to stop !?
13898                              <1>
13899 00011CBF 3805[AC910100]      <1>       cmp    byte [DirBuff_ValidData], al ; 0
13900 00011CC5 7608                <1>       jna    short sysfnf_4
13901                              <1>
13902 00011CC7 3B05[B1910100]      <1>       cmp    eax, [DirBuff_Cluster] ; 0 ?
13903 00011CCD 745E                <1>       je     short sysfnf_9
13904                              <1>
13905                              <1>       ;cmp   byte [Current_Dir_Level], 0
13906                              <1>         ;ja  short sysfnf_4
```

```
13907                                  <1>         ;jna short sysfnf_9
13908                                  <1>
13909                                  <1> sysfnf_4:
13910 00011CCF FE05[7B960100]          <1>         inc    byte [SWP_DRV_chg]
13911 00011CD5 E8E7B1FFFF              <1>         call   load_FAT_root_directory
13912 00011CDA 7351                    <1>         jnc    short sysfnf_9
13913                                  <1>         ; eax = error code (17, 'drv not ready or read error')
13914 00011CDC EB0B                    <1>         jmp    short sysfnf_err_1 ; read error ! (no FNF stop)
13915                                  <1>                                   ; (if you want, try again,
13916                                  <1>                                   ;  after re-placing the disk)
13917                                  <1> sysfnf_5:
13918 00011CDE 3C0C                    <1>         cmp    al, 12 ; 'no more files' error
13919 00011CE0 7507                    <1>         jne    short sysfnf_err_1 ; (no FNF stop -sysfnf will try
13920                                  <1>                                   ;  to read the directory again,
13921                                  <1>                                   ;  if the user calls sysfnf
13922                                  <1>                                   ;  just after this error return-)
13923                                  <1>         ; (FNF stop -sysfnf will not try
13924                                  <1>         ;  to read the directory again-)
13925                                  <1>
13926                                  <1> sysfnf_err_0:
13927 00011CE2 C605[74960100]00        <1>         mov    byte [FFF_Valid], 0 ; FNF stop sign
13928                                  <1> sysfnf_err_1:
13929 00011CE9 A3[64030300]            <1>         mov    [u.r0], eax
13930 00011CEE A3[C8030300]            <1>         mov    [u.error], eax
13931 00011CF3 E84B130000              <1>         call   reset_working_path
13932 00011CF8 E9DFBBFFFF              <1>         jmp    error
13933                                  <1>
13934                                  <1> sysfnf_6:
13935 00011CFD 803D[AC910100]00        <1>         cmp    byte [DirBuff_ValidData], 0
13936 00011D04 7608                    <1>         jna    short sysfnf_7
13937                                  <1>
13938 00011D06 3B05[B1910100]          <1>         cmp    eax, [DirBuff_Cluster]
13939 00011D0C 741F                    <1>         je     short sysfnf_9
13940                                  <1>
13941                                  <1> sysfnf_7:
13942 00011D0E FE05[7B960100]          <1>         inc    byte [SWP_DRV_chg]
13943 00011D14 803D[858A0100]01        <1>         cmp    byte [Current_FATType], 1
13944 00011D1B 7309                    <1>         jnb    short sysfnf_8
13945                                  <1>
13946                                  <1>         ; Singlix (TRFS) File System
13947                                  <1>         ; (access via compatibility buffer)
13948 00011D1D E867B2FFFF              <1>         call   load_FS_sub_directory
13949 00011D22 7309                    <1>         jnc    short sysfnf_9
13950                                  <1>
13951 00011D24 EBC3                    <1>         jmp    short sysfnf_err_1 ; read error (no FNF stop)
13952                                  <1>
13953                                  <1> sysfnf_8:
13954 00011D26 E821B2FFFF              <1>         call   load_FAT_sub_directory
13955 00011D2B 72BC                    <1>         jc     short sysfnf_err_1 ; read error (no FNF stop)
13956                                  <1>
13957                                  <1> sysfnf_9:
13958 00011D2D E8FE77FFFF              <1>         call   find_next_file
13959 00011D32 72AA                    <1>         jc     short sysfnf_5
13960                                  <1>
13961 00011D34 A0[75960100]            <1>         mov    al, [FFF_Attrib]
13962                                  <1>         ;or    al, al
13963                                  <1>         ;jz    short sysfnf_10 ; 0 = No filter
13964 00011D39 34FF                    <1>         xor    al, 0FFh
13965 00011D3B 20D8                    <1>         and    al, bl
13966 00011D3D 75EE                    <1>         jnz    short sysfnf_9 ; search for next file until
13967                                  <1>                               ; an error return from
13968                                  <1>                               ; find_next_file procedure
13969                                  <1> sysfnf_10:
13970 00011D3F 0FB60D[74960100]        <1>           movzx      ecx, byte [FFF_Valid]
13971 00011D46 80F980                  <1>         cmp    cl, 128 ; complete FindFile structure/table
13972 00011D49 0F84A2FEFFFF            <1>         je     sysfnf_11
13973                                  <1>         ;cmp   cl, 24  ; basic parameters
13974                                  <1>         ;je    sysfnf_12
13975 00011D4F E9AFFEFFFF              <1>         jmp    sysfnf_12
13976                                  <1>
13977                                  <1> writei:
13978                                  <1>         ; 26/10/2016
13979                                  <1>         ; 25/10/2016
13980                                  <1>         ; 23/10/2016
13981                                  <1>         ; 22/10/2016
13982                                  <1>         ; 19/10/2016 - TRDOS 386 (TRDOS v2.0)
13983                                  <1>         ; 19/05/2015 - 20/05/2015 (Retro UNIX 386 v1)
13984                                  <1>         ; 12/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
13985                                  <1>         ;
13986                                  <1>         ; Write data to file with first cluster number in EAX
13987                                  <1>         ;
13988                                  <1>         ; INPUTS ->
13989                                  <1>         ;    EAX - First cluster number of the file
13990                                  <1>         ;    EBX - File number (Open file index number)
13991                                  <1>         ;    u.count - byte count to be written
13992                                  <1>         ;    u.base - points to user buffer
13993                                  <1>         ;    u.fofp - points to dword with current file offset
13994                                  <1>         ;    i.size - file size
13995                                  <1>         ;    cdev - logical dos drive number of the file
13996                                  <1>         ; OUTPUTS ->
13997                                  <1>         ;    u.count - cleared
13998                                  <1>         ;    u.nread - accumulates total bytes passed back
13999                                  <1>         ;    i.size - new file size (if file byte offset overs file size)
14000                                  <1>         ;    u.fofp - points to u.off (with new offset value)
14001                                  <1>         ;
14002                                  <1>         ; (Retro UNIX Prototype : 11/11/2012 - 18/11/2012, UNIXCOPY.ASM)
14003                                  <1>         ; ((Modified registers: eax, edx, ebx, ecx, esi, edi, ebp))
14004                                  <1>
14005 00011D54 31C9                    <1>         xor    ecx, ecx
14006 00011D56 890D[8C030300]          <1>         mov    [u.nread], ecx  ; 0
14007 00011D5C 66890D[C4030300]        <1>         mov    [u.pcount], cx ; 19/05/2015
14008 00011D63 390D[88030300]          <1>         cmp    [u.count], ecx
14009 00011D69 7701                    <1>         ja     short writei_1
14010 00011D6B C3                      <1>         retn
14011                                  <1> writei_1:
```

```
14012 00011D6C 881D[34960100]      <1>       mov    [writei.ofn], bl ; Open file number
14013 00011D72 880D[6F960100]      <1>       mov    [setfmod], cl ; 0 ; reset 'update lm date&time' sign
14014                              <1> dskw_0:
14015                              <1>       ; 26/10/2016
14016                              <1>       ; 22/10/2016, 23/10/2016, 25/10/2016
14017                              <1>       ; 19/10/2016 - TRDOS 386 (TRDOS v2.0)
14018                              <1>       ; 31/05/2015 - 25/07/2015 (Retro UNIX 386 v1)
14019                              <1>       ; 26/04/2013 - 20/09/2013 (Retro UNIX 8086 v1)
14020                              <1>       ;
14021                              <1>       ; 01/08/2013 (mkdir_w check)
14022 00011D78 E8D7000000          <1>       call   mget_w
14023                              <1>       ; eax = sector/block number
14024                              <1>
14025 00011D7D 8B1D[74030300]      <1>       mov     ebx, [u.fofp]
14026 00011D83 8B13                <1>       mov    edx, [ebx]
14027 00011D85 81E2FF010000        <1>       and    edx, 1FFh  ; / test the lower 9 bits of the file offset
14028 00011D8B 750C                <1>       jnz    short dskw_1 ; / if its non-zero, branch
14029                              <1>                           ; if zero, file offset = 0,
14030                              <1>                                 ; / 512, 1024,...(i.e., start of new block)
14031 00011D8D 813D[88030300]0002- <1>       cmp    dword [u.count], 512
14031 00011D95 0000                <1>
14032                              <1>                             ; / if zero, is there enough data to fill
14033                              <1>                             ; / an entire block? (i.e., no. of
14034 00011D97 7337                <1>       jnb    short dskw_2 ; / bytes to be written greater than 512.?
14035                              <1>                           ; / Yes, branch. Don't have to read block
14036                              <1> dskw_1: ; in as no past info. is to be saved
14037                              <1>       ; (the entire block will be overwritten).
14038                              <1>       ; 23/10/2016
14039                              <1>
14040 00011D99 BB[94070300]        <1>       mov    ebx, writei_buffer
14041                              <1>       ; esi = logical dos drive description table address
14042                              <1>       ; eax = sector number
14043                              <1>       ; ebx = buffer address (in kernel's memory space)
14044                              <1>       ; ecx = sector count
14045 00011D9E B901000000          <1>       mov    ecx, 1
14046 00011DA3 E8A30D0000          <1>       call   disk_read
14047                              <1>       ;call  dskrd  ; / no, must retain old info..
14048                              <1>                             ; / Hence, read block 'r1' into an I/O buffer
14049 00011DA8 7326                <1>       jnc    short dskw_2
14050                              <1>
14051                              <1>       ; disk read error
14052 00011DAA B811000000          <1>       mov    eax, 17 ; drive not ready or READ ERROR !
14053                              <1> dskw_err: ; jump from disk write error
14054 00011DAF A3[64030300]        <1>       mov    [u.r0], eax
14055 00011DB4 A3[C8030300]        <1>       mov    [u.error], eax
14056                              <1>
14057 00011DB9 803D[6F960100]00    <1>       cmp    byte [setfmod], 0
14058 00011DC0 0F8616BBFFFF        <1>       jna    error
14059                              <1>
14060 00011DC6 E8AF030000          <1>       call   update_file_lmdt ; update last modif. date&time of the file
14061                              <1>       ;mov   byte [setfmod], 0
14062                              <1>
14063 00011DCB E90CBBFFFF          <1>       jmp    error
14064                              <1>
14065                              <1> dskw_2: ; 3:
14066                              <1>       ; 23/10/2016
14067 00011DD0 C605[10960100]01    <1>       mov    byte [writei.valid], 1 ; writei buffer contains valid data
14068 00011DD7 56                  <1>       push   esi ; logical dos drive description table address
14069                              <1>       ; EAX (r1) = block/sector number
14070                              <1>       ;call  wslot
14071                              <1>              ; jsr r0,wslot / set write and inhibit bits in I/O queue,
14072                              <1>                       ; / proc. status=0, r5 points to 1st word of data
14073 00011DD8 803D[C6030300]00    <1>       cmp    byte [u.kcall], 0
14074 00011DDF 770F                <1>       ja     short dskw_4 ; zf=0 -> the caller is 'mkdir'
14075                              <1>       ;
14076 00011DE1 66833D[C4030300]00  <1>       cmp    word [u.pcount], 0
14077 00011DE9 7705                <1>       ja     short dskw_4
14078                              <1> dskw_3:
14079                              <1>       ; [u.base] = virtual address to transfer (as source address)
14080 00011DEB E821FAFFFF          <1>       call   trans_addr_r ; translate virtual address to physical (r)
14081                              <1> dskw_4:
14082 00011DF0 BB[94070300]        <1>       mov    ebx, writei_buffer
14083                              <1>       ; EBX (r5) = system (I/O) buffer address
14084 00011DF5 E883FAFFFF          <1>       call   sioreg
14085                              <1>       ; ESI = file (user data) offset
14086                              <1>       ; EDI = sector (I/O) buffer offset
14087                              <1>       ; ECX = byte count
14088                              <1>       ;
14089 00011DFA F3A4                <1>       rep    movsb
14090                              <1>       ; 25/07/2015
14091                              <1>       ; eax = remain bytes in buffer
14092                              <1>         ;       (check if remain bytes in the buffer > [u.pcount])
14093 00011DFC 09C0                <1>       or     eax, eax
14094 00011DFE 75EB                <1>       jnz    short dskw_3 ; (page end before system buffer end!)
14095                              <1>
14096                              <1>       ; 23/10/2016
14097 00011E00 B101                <1>       mov    cl, 1
14098 00011E02 5E                  <1>       pop    esi
14099 00011E03 A1[14960100]        <1>       mov    eax, [writei.sector]
14100                              <1>       ; esi = logical dos drive description table address
14101                              <1>       ; eax = sector number
14102                              <1>       ; ebx = writei buffer address
14103                              <1>       ; ecx = sector count
14104 00011E08 E82F0D0000          <1>       call   disk_write ; / yes, write the block
14105 00011E0D 7307                <1>       jnc    short dskw_5
14106                              <1>
14107 00011E0F B812000000          <1>       mov    eax, 18 ; drive not ready or WRITE ERROR !
14108 00011E14 EB99                <1>       jmp    short dskw_err
14109                              <1>
14110                              <1> dskw_5:
14111                              <1>       ; 26/10/2016
14112 00011E16 0FB61D[34960100]    <1>       movzx  ebx, byte [writei.ofn] ; open file number
14113 00011E1D C0E302              <1>       shl    bl, 2 ; *4
14114 00011E20 8B83[049A0100]      <1>       mov    eax, [ebx+OF_POINTER]
14115 00011E26 3B83[2C9A0100]      <1>       cmp    eax, [ebx+OF_SIZE]
```

```
14116 00011E2C 7606                <1>       jna    short dskw_6
14117 00011E2E 8983[2C9A0100]      <1>       mov    [ebx+OF_SIZE], eax
14118                              <1> dskw_6:
14119                              <1>       ;shr   bl, 2
14120 00011E34 833D[88030300]00    <1>         cmp     dword [u.count], 0 ; / any more data to write?
14121 00011E3B 760A                <1>       jna    short dskw_7
14122 00011E3D A1[24960100]        <1>       mov    eax, [writei.fclust]
14123 00011E42 E931FFFFFF          <1>       jmp    dskw_0 ; / yes, branch
14124                              <1> dskw_7:
14125                              <1>       ; update last modif. date&time of the file
14126                              <1>       ; (also updates file size as OF_SIZE)
14127 00011E47 E82E030000          <1>       call   update_file_lmdt
14128                              <1>       ;mov   byte [setfmod], 0
14129                              <1>
14130                              <1>       ; 03/08/2013
14131 00011E4C C605[C6030300]00    <1>       mov    byte [u.kcall], 0
14132                              <1>       ; 23/10/2016
14133                              <1>       ;mov   eax, [writei.fclust]
14134 00011E53 C3                  <1>       retn
14135                              <1>
14136                              <1> mget_w:
14137                              <1>       ; 02/11/2016
14138                              <1>       ; 01/11/2016
14139                              <1>       ; 23/10/2016, 31/10/2016
14140                              <1>       ; 22/10/2016 - TRDOS 386 (TRDOS v2.0)
14141                              <1>       ; 03/06/2015 (Retro UNIX 386 v1, 'mget', u.5s)
14142                              <1>       ; 22/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
14143                              <1>       ;
14144                              <1>       ; Get existing or (allocate) a new disk block for file
14145                              <1>       ;
14146                              <1>       ; INPUTS ->
14147                              <1>       ;    [u.fofp] = file offset pointer
14148                              <1>       ;    [i.size] = file size
14149                              <1>       ;    [u.count] = byte count
14150                              <1>       ;    EAX = First cluster
14151                              <1>       ;    [cdev] = Logical dos drive number
14152                              <1>       ;    [writei.ofn] = File Number
14153                              <1>       ;                (Open file index, 0 based)
14154                              <1>       ;    ([u.off] = file offset)
14155                              <1>       ; OUTPUTS ->
14156                              <1>       ;    EAX = logical sector number
14157                              <1>       ;    ESI = Logical Dos Drive Description Table address
14158                              <1>       ;
14159                              <1>       ; Modified registers: EDX, EBX, ECX, ESI, EDI, EBP
14160                              <1>
14161 00011E54 8B35[74030300]      <1>         mov     esi, [u.fofp]
14162 00011E5A 8B2E                <1>       mov    ebp, [esi] ; u.off (or EBX*4+OF_POINTER)
14163                              <1>
14164 00011E5C 29C9                <1>       sub    ecx, ecx
14165 00011E5E 8A2D[46030300]      <1>       mov    ch, [cdev]
14166                              <1>
14167 00011E64 BE00010900          <1>       mov    esi, Logical_DOSDisks
14168 00011E69 01CE                <1>       add    esi, ecx
14169                              <1>
14170                              <1>       ; 31/10/2016
14171 00011E6B 89C3                <1>       mov    ebx, eax ; First Cluster or FDT address
14172                              <1>
14173 00011E6D 807E0300            <1>       cmp    byte [esi+LD_FATType], 0
14174 00011E71 0F86DD010000        <1>       jna    mget_w_14 ; Singlix FS
14175                              <1>
14176 00011E77 0FB74611            <1>       movzx  eax, word [esi+LD_BPB+BytesPerSec]
14177 00011E7B 0FB65613            <1>       movzx  edx, byte [esi+LD_BPB+SecPerClust]
14178 00011E7F 8815[12960100]      <1>       mov    [writei.spc], dl  ; sectors per cluster
14179 00011E85 F7E2                <1>       mul    edx
14180                              <1>       ; edx = 0
14181                              <1>       ; eax = bytes per cluster (<= 65536)
14182                              <1>
14183                              <1>       ; 02/11/2016
14184 00011E87 89C1                <1>       mov    ecx, eax
14185 00011E89 48                  <1>       dec    eax
14186 00011E8A 66A3[18960100]      <1>       mov    [writei.bpc], ax
14187                              <1>
14188 00011E90 89E8                <1>       mov    eax, ebp
14189 00011E92 0305[88030300]      <1>       add    eax, [u.count] ; next file position
14190 00011E98 3B05[55040300]      <1>       cmp    eax, [i.size] ; <= file size ?
14191 00011E9E 0F86FC000000        <1>       jna    mget_w_4 ; no
14192                              <1>
14193 00011EA4 F7F1                <1>       div    ecx
14194 00011EA6 A3[20960100]        <1>       mov    [writei.c_index], eax ; cluster index
14195                              <1>       ; edx = byte offset in cluster (<= 65535)
14196                              <1>       ;mov   [writei.offset], dx
14197                              <1>       ;shr   dx, 9 ; / 512
14198                              <1>       ;mov   [writei.s_index], dl ; sector index in cluster (0 to spc -1)
14199                              <1>
14200 00011EAB 29D2                <1>       sub    edx, edx ; 01/11/2016
14201 00011EAD 8915[14960100]      <1>       mov    [writei.sector], edx ; 0
14202 00011EB3 668915[1A960100]    <1>       mov    [writei.offset], dx  ; byte offset in cluster
14203 00011EBA 8815[13960100]      <1>       mov    [writei.s_index], dl ; sector index in cluster (0 to spc -1)
14204                              <1>
14205 00011EC0 89D8                <1>       mov    eax, ebx ; First Cluster
14206                              <1>
14207                              <1>       ; is this the 1st mget_w or a next mget_w call ? (by 'writei')
14208 00011EC2 3815[10960100]      <1>       cmp    byte [writei.valid], dl ; 0
14209 00011EC8 7624                <1>       jna    short mget_w_0
14210                              <1>
14211 00011ECA 8815[10960100]      <1>       mov    byte [writei.valid], dl ; 0 ; reset ('writei' will set it)
14212                              <1>
14213 00011ED0 3B05[24960100]      <1>       cmp    eax, [writei.fclust]
14214 00011ED6 7516                <1>       jne    short mget_w_0
14215                              <1>
14216 00011ED8 8A0D[46030300]      <1>       mov    cl, [cdev]
14217 00011EDE 3A0D[11960100]      <1>       cmp    cl, [writei.drv]
14218 00011EE4 7508                <1>       jne    short mget_w_0
14219                              <1>       ; [writei.l_clust] & [writei.l_index] are valid,
14220                              <1>       ;  we don't need to get last cluster & last cluster index
```

```
14221 00011EE6 8B0D[30960100]      <1>       mov    ecx, [writei.l_index]
14222 00011EEC EB64                <1>       jmp    short mget_w_2
14223                              <1> mget_w_0:
14224 00011EEE A3[24960100]        <1>       mov    [writei.fclust], eax ; first cluster
14225                              <1>       ; edx = 0
14226 00011EF3 A3[1C960100]        <1>       mov    [writei.cluster], eax ; first cluster ; 01/11/2016
14227 00011EF8 8915[28960100]      <1>       mov    [writei.fs_index], edx ; 0 ; curret cluster index
14228                              <1>
14229                              <1>       ; FAT file system (FAT12, FAT16, FAT32)
14230 00011EFE E85EB6FFFF          <1>       call   get_last_cluster
14231 00011F03 0F822B010000        <1>       jc     mget_w_err ; eax = error code
14232                              <1>
14233 00011F09 A3[2C960100]        <1>       mov    [writei.lclust], eax ; last cluster
14234                              <1>
14235 00011F0E 8B0D[50940100]      <1>       mov    ecx, [glc_index] ; last cluster index
14236 00011F14 890D[30960100]      <1>       mov    [writei.l_index], ecx
14237                              <1>
14238 00011F1A A0[34960100]        <1>       mov    al, [writei.ofn]
14239 00011F1F FEC0                <1>       inc    al
14240 00011F21 A2[6F960100]        <1>       mov    [setfmod], al ; update lm date&time sign
14241                              <1>
14242                              <1> mget_w_1:
14243 00011F26 3B0D[20960100]      <1>       cmp    ecx, [writei.c_index]  ; last cluster index
14244 00011F2C 7324                <1>       jnb    short mget_w_2 ; 01/11/2016
14245                              <1>
14246 00011F2E A1[2C960100]        <1>       mov    eax, [writei.lclust]
14247                              <1>       ; EAX = Last cluster
14248 00011F33 E837B7FFFF          <1>       call   add_new_cluster
14249 00011F38 0F82F6000000        <1>       jc     mget_w_err ; eax = error code
14250                              <1>       ; edx = 0
14251 00011F3E A3[2C960100]        <1>       mov    [writei.lclust], eax ; (new) last cluster
14252 00011F43 8B0D[30960100]      <1>       mov    ecx, [writei.l_index]
14253 00011F49 41                  <1>       inc    ecx ; add 1 to last cluster index
14254 00011F4A 890D[30960100]      <1>       mov    [writei.l_index], ecx ; current last cluster index
14255                              <1>
14256 00011F50 EBD4                <1>       jmp    short mget_w_1
14257                              <1>
14258                              <1> mget_w_2:
14259 00011F52 89E9                <1>       mov    ecx, ebp
14260 00011F54 030D[88030300]      <1>       add    ecx, [u.count]
14261 00011F5A 890D[55040300]      <1>       mov    [i.size], ecx ; save new file size
14262                              <1>       ;sub   edx, edx ; 0
14263                              <1>
14264 00011F60 A0[46030300]        <1>       mov    al, [cdev]
14265 00011F65 A2[11960100]        <1>       mov    [writei.drv], al ; physical drive number
14266                              <1>       ; edx = 0
14267 00011F6A 89E8                <1>       mov    eax, ebp ; file offset
14268 00011F6C 0FB70D[18960100]    <1>       movzx  ecx, word [writei.bpc] ; bytes per cluster - 1
14269 00011F73 41                  <1>       inc    ecx ; bytes per cluster
14270 00011F74 F7F1                <1>       div    ecx
14271                              <1>       ; edx = byte offset in cluster (<= 65535)
14272                              <1>       ; eax = cluster index
14273 00011F76 A3[20960100]        <1>       mov    [writei.c_index], eax
14274 00011F7B 668915[1A960100]    <1>       mov    [writei.offset], dx
14275 00011F82 66C1EA09            <1>       shr    dx, 9 ; / 512
14276 00011F86 8815[13960100]      <1>       mov    [writei.s_index], dl ; sector index in cluster (0 to spc -1)
14277                              <1>
14278                              <1> mget_w_3:
14279 00011F8C 3B05[30960100]      <1>       cmp    eax, [writei.l_index] ; last cluster index
14280 00011F92 752A                <1>       jne    short mget_w_5
14281                              <1>
14282 00011F94 A3[28960100]        <1>       mov    [writei.fs_index], eax ; cluster index (for next check)
14283 00011F99 A1[2C960100]        <1>       mov    eax, [writei.lclust] ; last cluster
14284 00011F9E EB60                <1>       jmp    short mget_w_10
14285                              <1>
14286                              <1> mget_w_4: ; 02/11/2016
14287                              <1>       ; eax = next file position
14288 00011FA0 2B05[88030300]      <1>       sub    eax, [u.count] ; current file position
14289                              <1>       ; edx = 0
14290                              <1>       ; ecx = bytes per cluster
14291 00011FA6 F7F1                <1>       div    ecx
14292 00011FA8 A3[20960100]        <1>       mov    [writei.c_index], eax ; cluster index
14293 00011FAD 668915[1A960100]    <1>       mov    [writei.offset], dx
14294 00011FB4 66C1EA09            <1>       shr    dx, 9 ; / 512
14295 00011FB8 8815[13960100]      <1>       mov    [writei.s_index], dl ; sector index in cluster (0 to spc -1)
14296                              <1>
14297                              <1> mget_w_5:
14298 00011FBE 21C0                <1>       and    eax, eax ; 0 = First Cluster's index number
14299 00011FC0 750C                <1>       jnz    short mget_w_6
14300                              <1>
14301 00011FC2 A3[28960100]        <1>       mov    [writei.fs_index], eax ; cluster index (for next check)
14302 00011FC7 A1[24960100]        <1>       mov    eax, [writei.fclust] ; first cluster
14303 00011FCC EB32                <1>       jmp    short mget_w_10
14304                              <1>
14305                              <1> mget_w_6:
14306 00011FCE 3B05[28960100]      <1>       cmp    eax, [writei.fs_index] ; current cluster index (>0)
14307 00011FD4 7507                <1>       jne    short mget_w_7
14308 00011FD6 A1[1C960100]        <1>       mov    eax, [writei.cluster] ; current cluster
14309 00011FDB EB3A                <1>       jmp    short mget_w_11
14310                              <1>
14311                              <1> mget_w_7:
14312 00011FDD 89C1                <1>       mov    ecx, eax
14313 00011FDF 2B0D[28960100]      <1>       sub    ecx, [writei.fs_index]
14314 00011FE5 730D                <1>       jnc    short mget_w_8
14315                              <1>       ; get cluster by index from the first cluster
14316 00011FE7 A1[24960100]        <1>       mov    eax, [writei.fclust]
14317 00011FEC 8B0D[20960100]      <1>       mov    ecx, [writei.c_index]
14318 00011FF2 EB05                <1>       jmp    short mget_w_9
14319                              <1>
14320                              <1> mget_w_8:
14321 00011FF4 A1[1C960100]        <1>       mov    eax, [writei.cluster] ; beginning cluster
14322                              <1>       ; ecx = cluster sequence number after the beginning cluster
14323                              <1>       ; sub  edx, edx ; 0
14324                              <1>
14325                              <1> mget_w_9:
```

```
14326                                  <1>       ; EAX = Beginning cluster
14327                                  <1>       ; EDX = Sector index in disk/file section
14328                                  <1>       ;       (Only for SINGLIX file system!)
14329                                  <1>       ; ECX = Cluster sequence number after the beginning cluster
14330                                  <1>       ; ESI = Logical DOS Drive Description Table address
14331 00011FF9 E877B7FFFF              <1>       call   get_cluster_by_index
14332 00011FFE 7234                    <1>       jc     short mget_w_err ; error code in EAX
14333                                  <1>       ; EAX = Cluster number
14334                                  <1> mget_w_10:
14335 00012000 A3[1C960100]            <1>       mov    [writei.cluster], eax ; FDT number for Singlix File System
14336                                  <1>
14337 00012005 807E0300                <1>       cmp    byte [esi+LD_FATType], 0
14338 00012009 7638                    <1>       jna    short mget_w_13
14339                                  <1>       ; 01/11/2016
14340 0001200B 8B15[20960100]          <1>       mov    edx, [writei.c_index]
14341 00012011 8915[28960100]          <1>       mov    [writei.fs_index], edx
14342                                  <1> mget_w_11:
14343 00012017 83E802                  <1>       sub    eax, 2
14344 0001201A 0FB615[12960100]        <1>       movzx  edx, byte [writei.spc]
14345 00012021 F7E2                    <1>       mul    edx
14346                                  <1>
14347 00012023 034668                  <1>       add    eax, [esi+LD_DATABegin]
14348 00012026 8A15[13960100]          <1>       mov    dl, [writei.s_index]
14349 0001202C 01D0                    <1>       add    eax, edx
14350                                  <1> mget_w_12:
14351 0001202E A3[14960100]            <1>       mov    [writei.sector], eax
14352                                  <1>       ;; buffer validation must be done in writei
14353                                  <1>       ;;mov  byte [writei.valid], 1
14354 00012033 C3                      <1>       retn
14355                                  <1>
14356                                  <1> mget_w_err:
14357 00012034 A3[C8030300]            <1>       mov    [u.error], eax
14358 00012039 A3[64030300]            <1>       mov    [u.r0], eax
14359 0001203E E999B8FFFF              <1>       jmp    error
14360                                  <1>
14361                                  <1> mget_w_13:
14362                                  <1>       ; EAX = FDT number (Current Section)
14363                                  <1>       ; EDX = Sector index from the first section (0,1,2,3,4...)
14364 00012043 2B15[28960100]          <1>       sub    edx, [writei.fs_index]
14365                                  <1>       ; EDX = Sector index from current section
14366 00012049 8915[28960100]          <1>       mov    [writei.fs_index], edx
14367 0001204F 40                      <1>       inc    eax ; the first data sector in FS disk section
14368 00012050 01D0                    <1>       add    eax, edx
14369 00012052 EBDA                    <1>       jmp    short mget_w_12
14370                                  <1>
14371                                  <1> mget_w_14:
14372 00012054 8A4E12                  <1>       mov    cl, [esi+LD_FS_BytesPerSec+1]
14373 00012057 D0E9                    <1>       shr    cl, 1 ;  ; 1 for 512 bytes, 4 for 2048 bytes
14374 00012059 880D[12960100]          <1>       mov    [writei.spc], cl  ; sectors per cluster
14375                                  <1>       ; NOTE: writei bytes per sector value is always 512 !
14376 0001205F 66C705[18960100]00-     <1>       mov    word [writei.bpc], 512
14376 00012067 02                      <1>
14377                                  <1>
14378 00012068 89E9                    <1>       mov    ecx, ebp
14379 0001206A 030D[88030300]          <1>       add    ecx, [u.count] ; next file position
14380 00012070 3B0D[55040300]          <1>       cmp    ecx, [i.size] ; <= file size ?
14381 00012076 0F86C8000000            <1>       jna    mget_w_19 ; no
14382                                  <1>
14383 0001207C 29D2                    <1>       sub    edx, edx ; 0
14384 0001207E 8915[14960100]          <1>       mov    [writei.sector], edx ; 0
14385 00012084 668915[1A960100]        <1>       mov    [writei.offset], dx  ; byte offset in cluster
14386 0001208B 8815[13960100]          <1>       mov    [writei.s_index], dl ; sector index in cluster (0 to spc -1)
14387                                  <1>
14388 00012091 C1E909                  <1>       shr    ecx, 9 ; 1 cluster = 512 bytes
14389 00012094 890D[20960100]          <1>       mov    [writei.c_index], ecx ; section/cluster index
14390                                  <1>
14391 0001209A 89D8                    <1>       mov    eax, ebx ; FDT number (First FDT address)
14392                                  <1>
14393                                  <1>       ; is this the 1st mget_w or a next mget_w call ? (by 'writei')
14394 0001209C 3815[10960100]          <1>       cmp    byte [writei.valid], dl ; 0
14395 000120A2 7624                    <1>       jna    short mget_w_15
14396                                  <1>
14397 000120A4 8815[10960100]          <1>       mov    byte [writei.valid], dl ; 0 ; reset ('writei' will set it)
14398                                  <1>
14399 000120AA 3B05[24960100]          <1>       cmp    eax, [writei.fclust]
14400 000120B0 7516                    <1>       jne    short mget_w_15
14401                                  <1>
14402 000120B2 8A0D[46030300]          <1>       mov    cl, [cdev]
14403 000120B8 3A0D[11960100]          <1>       cmp    cl, [writei.drv]
14404 000120BE 7508                    <1>       jne    short mget_w_15
14405                                  <1>       ; [writei.l_clust] & [writei.l_index] are valid,
14406                                  <1>       ;  we don't need to get last cluster & last cluster index
14407 000120C0 8B0D[30960100]          <1>       mov    ecx, [writei.l_index]
14408 000120C6 EB49                    <1>       jmp    short mget_w_17
14409                                  <1> mget_w_15:
14410 000120C8 A3[24960100]            <1>       mov    [writei.fclust], eax ; first section (FDT number)
14411                                  <1>       ; edx = 0
14412 000120CD 8915[1C960100]          <1>       mov    [writei.cluster], edx ; 0 ; current section
14413 000120D3 8915[28960100]          <1>       mov    [writei.fs_index], edx ; 0 ; curret section index
14414                                  <1>
14415                                  <1>       ; eax = FDT number (section 0 header address)
14416 000120D9 E8C1B6FFFF              <1>       call   get_last_section
14417 000120DE 0F8250FFFFFF            <1>       jc     mget_w_err ; eax = error code
14418                                  <1>
14419 000120E4 8915[28960100]          <1>       mov    [writei.fs_index], edx ; sector index in last section
14420                                  <1>
14421 000120EA A3[2C960100]            <1>       mov    [writei.lclust], eax ; last section address
14422                                  <1>
14423 000120EF 8B0D[50940100]          <1>       mov    ecx, [glc_index] ; last section index
14424 000120F5 890D[30960100]          <1>       mov    [writei.l_index], ecx
14425                                  <1>
14426 000120FB A0[34960100]            <1>       mov    al, [writei.ofn]
14427 00012100 FEC0                    <1>       inc    al
14428 00012102 A2[6F960100]            <1>       mov    [setfmod], al ; update lm date&time sign
14429                                  <1>
```

```
14430                                  <1> mget_w_16:
14431                                  <1>       ; edx = (existing) last section (sector) index
14432 00012107 8B0D[20960100]          <1>       mov    ecx, [writei.c_index] ; final section (sector) index
14433 0001210D 29D1                    <1>       sub    ecx, edx
14434 0001210F 7633                    <1>       jna    short mget_w_19
14435                                  <1>        ; ecx = sector count
14436                                  <1> mget_w_17:
14437 00012111 A1[2C960100]            <1>       mov    eax, [writei.lclust]
14438                                  <1>       ; ESI = Logical dos drv desc. table address
14439                                  <1>         ; EAX = Last section
14440                                  <1>         ; (ECX = 0 for directory)
14441                                  <1>         ; ECX = sector count (except FDT)
14442 00012116 E847ACFFFF              <1>       call   add_new_fs_section
14443 0001211B 7312                    <1>       jnc    short mget_w_18
14444                                  <1>
14445                                  <1>       ; If error number = 27h (insufficient disk space)
14446                                  <1>       ; it is needed to check free consequent sectors
14447                                  <1>       ; (1 data sector at least and +1 section header sector)
14448                                  <1>
14449 0001211D 83F827                  <1>       cmp    eax, 27h
14450 00012120 0F850EFFFFFF            <1>       jne    mget_w_err ; eax = error code
14451                                  <1>
14452                                  <1>       ; ecx = count of free consequent sectors
14453                                  <1>       ; ecx must be > 1 (1 data + 1 header sector)
14454 00012126 49                      <1>       dec    ecx
14455 00012127 0F8407FFFFFF            <1>       jz     mget_w_err
14456 0001212D EBE2                    <1>       jmp    short mget_w_17
14457                                  <1>
14458                                  <1> mget_w_18:
14459 0001212F A3[2C960100]            <1>       mov    [writei.lclust], eax ; (new) last section
14460                                  <1>       ; ecx = sector count (except section header)
14461 00012134 8B15[30960100]          <1>       mov    edx, [writei.l_index]
14462 0001213A 01CA                    <1>       add    edx, ecx ; add sector count to index
14463 0001213C 8915[30960100]          <1>       mov    [writei.l_index], edx
14464 00012142 EBC3                    <1>       jmp    short mget_w_16
14465                                  <1>
14466                                  <1> mget_w_19:
14467 00012144 89E9                    <1>       mov    ecx, ebp
14468 00012146 030D[88030300]          <1>       add    ecx, [u.count]
14469 0001214C 890D[55040300]          <1>       mov    [i.size], ecx ; save new file size
14470                                  <1>       ;sub   edx, edx ; 0
14471                                  <1>
14472 00012152 A0[46030300]            <1>       mov    al, [cdev]
14473 00012157 A2[11960100]            <1>       mov    [writei.drv], al ; physical drive number
14474                                  <1>       ; edx = 0
14475 0001215C 89E8                    <1>       mov    eax, ebp ; file offset
14476 0001215E 89C2                    <1>       mov    edx, eax
14477                                  <1>       ; 1 cluster = 512 bytes (for Singlix FS)
14478 00012160 C1E809                  <1>       shr    eax, 9  ; / 512
14479 00012163 81E2FF010000            <1>       and    edx, 1FFh
14480                                  <1>       ; edx = byte offset in cluster/sector (<= 511)
14481                                  <1>       ; eax = section (sector/cluster) index
14482 00012169 A3[20960100]            <1>       mov    [writei.c_index], eax
14483 0001216E 668915[1A960100]        <1>       mov    [writei.offset], dx
14484                                  <1>       ;mov   byte [writei.s_index], 0 ; sector index in cluster
14485 00012175 E912FEFFFF              <1>       jmp    mget_w_3
14486                                  <1>
14487                                  <1> update_file_lmdt: ; & update file size
14488                                  <1>       ; 26/10/2016
14489                                  <1>       ; 24/10/2016
14490                                  <1>       ; 23/10/2016
14491                                  <1>       ; 22/10/2016 - TRDOS 386 (TRDOS v2.0)
14492                                  <1>       ;
14493                                  <1>       ; Update last modification date&time of file
14494                                  <1>       ; (call from syswrite -> writei)
14495                                  <1>       ; ((also updates file size)) // 26/10/2016
14496                                  <1>       ;
14497                                  <1>       ; INPUT:
14498                                  <1>       ;      byte [setfmod] = open file number
14499                                  <1>       ; OUTPUT:
14500                                  <1>       ;      cf = 0 -> success !
14501                                  <1>       ;      cf = 1 -> lmdt update has been failed!
14502                                  <1>       ;
14503                                  <1>       ; Modified registers: eax, ebx, ecx, edx, esi, edi
14504                                  <1>       ;
14505                                  <1>
14506                                  <1>       ;cmp   byte [setfmod], 0
14507                                  <1>       ;jna   short uflmdt_2 ; nothing to do
14508                                  <1>
14509 0001217A 31C0                    <1>       xor    eax, eax
14510                                  <1>
14511 0001217C 0FB61D[6F960100]        <1>       movzx  ebx, byte [setfmod]
14512 00012183 FECB                    <1>       dec    bl ; open file index number (0 based)
14513                                  <1>
14514 00012185 8AA3[DC990100]          <1>       mov    ah, [ebx+OF_DRIVE]
14515 0001218B BE00010900              <1>       mov    esi, Logical_DOSDisks
14516 00012190 01C6                    <1>       add    esi, eax
14517 00012192 C0E302                  <1>       shl    bl, 2 ; *4
14518 00012195 8B8B[B4990100]          <1>       mov    ecx, [ebx+OF_FCLUSTER] ; first cluster
14519 0001219B 8B93[7C9A0100]          <1>       mov    edx, [ebx+OF_DIRCLUSTER] ; dir cluster
14520                                  <1>
14521 000121A1 D0EB                    <1>       shr    bl, 1 ; /2
14522 000121A3 0FB7BB[1C9B0100]        <1>       movzx  edi, word [ebx+OF_DIRENTRY]
14523                                  <1>
14524 000121AA 803D[AC910100]01        <1>       cmp    byte [DirBuff_ValidData], 1
14525 000121B1 726E                    <1>       jb     short uflmdt_4
14526                                  <1>
14527 000121B3 A0[AA910100]            <1>       mov    al, [DirBuff_DRV]
14528 000121B8 2C41                    <1>       sub    al, 'A'
14529 000121BA 38E0                    <1>       cmp    al, ah
14530 000121BC 7563                    <1>       jne    short uflmdt_4 ; different drive
14531 000121BE 8A4603                  <1>       mov    al, [esi+LD_FATType]
14532 000121C1 3A05[AB910100]          <1>       cmp    al, [DirBuff_FATType]
14533 000121C7 755B                    <1>       jne    short uflmdt_5 ; different FS type
14534 000121C9 3B15[B1910100]          <1>       cmp    edx, [DirBuff_Cluster]
```

```
14535 000121CF 7553               <1>       jne    short uflmdt_5 ; different cluster
14536                             <1>
14537                             <1> uflmdt_1:
14538                             <1>       ; Directory buffer is ready here!
14539                             <1>       ; OF_FCLUSTER must be compared/verified
14540 000121D1 BE00000800         <1>       mov    esi, Directory_Buffer
14541 000121D6 66C1E705           <1>       shl    di, 5 ; dir entry index * 32
14542 000121DA 01FE               <1>       add    esi, edi ; offset
14543                             <1>       ;
14544 000121DC F6460B18           <1>       test   byte [esi+DirEntry_Attr], 18h ; Vol & Dir
14545 000121E0 750F               <1>       jnz    short uflmdt_2 ; not a valid file !
14546 000121E2 668B4614           <1>       mov    ax, [esi+DirEntry_FstClusHI]
14547 000121E6 C1E010             <1>       shl    eax, 16
14548 000121E9 668B461A           <1>       mov    ax, [esi+DirEntry_FstClusLO]
14549 000121ED 39C8               <1>       cmp    eax, ecx ; same first cluster ?
14550 000121EF 7407               <1>       je     short uflmdt_3 ; yes, it is OK !!!
14551                             <1>
14552                             <1> uflmdt_2:
14553                             <1>       ; save directory buffer if has modified/changed sign
14554                             <1>       ; (It is good to save dir buff even if the searched
14555                             <1>       ; directory entry is not found !?)
14556 000121F1 E8C098FFFF         <1>       call   save_directory_buffer
14557 000121F6 F9                 <1>       stc    ; update failed
14558 000121F7 C3                 <1>       retn
14559                             <1>
14560                             <1> uflmdt_3:
14561                             <1>       ; Update directory entry
14562                             <1>       ; 26/10/2016
14563 000121F8 D0E3               <1>       shl    bl, 1 ; *2
14564 000121FA 8B83[2C9A0100]     <1>       mov    eax, [ebx+OF_SIZE] ; file size
14565 00012200 89461C             <1>       mov    [esi+DirEntry_FileSize], eax
14566                             <1>       ;
14567 00012203 E81098FFFF         <1>       call   convert_current_date_time
14568                             <1>       ; OUTPUT -> DX = Date in dos dir entry format
14569                             <1>         ;       AX = Time in dos dir entry format
14570 00012208 66894616           <1>       mov    [esi+DirEntry_WrtTime], ax
14571 0001220C 66895618           <1>       mov    [esi+DirEntry_WrtDate], dx
14572 00012210 66895612           <1>       mov    [esi+DirEntry_LastAccDate], dx
14573 00012214 C605[AC910100]02   <1>       mov    byte [DirBuff_ValidData], 2
14574 0001221B E89698FFFF         <1>       call   save_directory_buffer
14575 00012220 C3                 <1>       retn
14576                             <1>
14577                             <1> uflmdt_4:
14578                             <1>       ; Directory buffer sector read&write
14579                             <1>       ; 23/10/2016
14580                             <1>       ;
14581 00012221 8A4603             <1>       mov    al, [esi+LD_FATType]
14582                             <1> uflmdt_5:
14583 00012224 BB[9C090300]       <1>       mov    ebx, rw_buffer ; Common r/w sector buffer addr
14584                             <1>
14585 00012229 20C0               <1>       and    al, al ; 0 = Singlix FS
14586 0001222B 0F8492000000       <1>       jz     uflmdt_11
14587                             <1>
14588 00012231 21D2               <1>       and    edx, edx
14589 00012233 7521               <1>       jnz    short uflmdt_9
14590                             <1>
14591 00012235 3C02               <1>       cmp    al, 2  ; 3 = FAT32
14592 00012237 771A               <1>       ja     short uflmdt_8
14593                             <1>
14594 00012239 89F8               <1>       mov    eax, edi ; directory entry index number
14595 0001223B 66C1E804           <1>       shr    ax, 4 ; 16 entries per sector
14596 0001223F 034664             <1>       add    eax, [esi+LD_ROOTBegin]
14597                             <1>       ; eax = root directory sector
14598                             <1> uflmdt_6:
14599 00012242 50                 <1>       push   eax ; * ; disk sector address
14600 00012243 51                 <1>       push   ecx ; first cluster
14601 00012244 B901000000         <1>       mov    ecx, 1
14602                             <1>       ; ecx = sector count
14603 00012249 E8FD080000         <1>       call   disk_read
14604 0001224E 59                 <1>       pop    ecx
14605 0001224F 731A               <1>       jnc    short uflmdt_10
14606 00012251 58                 <1>       pop    eax ; *
14607                             <1> uflmdt_7:
14608 00012252 C3                 <1>       retn
14609                             <1>
14610                             <1> uflmdt_8:
14611 00012253 8B5632             <1>       mov    edx, [esi+LD_BPB+FAT32_RootFClust]
14612                             <1> uflmdt_9:
14613 00012256 83FA02             <1>       cmp    edx, 2
14614 00012259 72F7               <1>       jb     short uflmdt_7 ; invalid, nothing to do
14615                             <1>
14616 0001225B 83EA02             <1>       sub    edx, 2
14617 0001225E 89D0               <1>       mov    eax, edx
14618 00012260 0FB65613           <1>       movzx  edx, byte [esi+LD_BPB+SecPerClust]
14619 00012264 F7E2               <1>       mul    edx
14620 00012266 034668             <1>       add    eax, [esi+LD_DATABegin]
14621                             <1>       ; eax = sub directory (data) sector
14622 00012269 EBD7               <1>       jmp    short uflmdt_6
14623                             <1>
14624                             <1> uflmdt_10:
14625                             <1>       ; Directory sector buffer is ready here!
14626                             <1>       ; OF_FCLUSTER must be compared/verified
14627                             <1>       ; edi = dir entry index number (<= 2047)
14628 0001226B 6683E70F           <1>       and    di, 0Fh ; 16 entries per sector
14629 0001226F 66C1E705           <1>       shl    di, 5 ; dir entry index * 32
14630 00012273 81C7[9C090300]     <1>       add    edi, rw_buffer
14631                             <1>       ;
14632 00012279 F6470B18           <1>       test   byte [edi+DirEntry_Attr], 18h ; Vol & Dir
14633 0001227D 0F856EFFFFFF       <1>       jnz    uflmdt_2 ; not a valid file !
14634 00012283 668B5714           <1>       mov    dx, [edi+DirEntry_FstClusHI]
14635 00012287 C1E210             <1>       shl    edx, 16
14636 0001228A 668B571A           <1>       mov    dx, [edi+DirEntry_FstClusLO]
14637 0001228E 39CA               <1>       cmp    edx, ecx ; same first cluster ?
14638 00012290 0F855BFFFFFF       <1>       jne    uflmdt_2 ; no !?
14639                             <1>
```

```
14640                                   <1>       ; Update directory entry
14641 00012296 E87D97FFFF               <1>       call   convert_current_date_time
14642                                   <1>       ; OUTPUT -> DX = Date in dos dir entry format
14643                                   <1>         ;       AX = Time in dos dir entry format
14644 0001229B 66894716                 <1>       mov    [edi+DirEntry_WrtTime], ax
14645 0001229F 66895718                 <1>       mov    [edi+DirEntry_WrtDate], dx
14646 000122A3 66895712                 <1>       mov    [edi+DirEntry_LastAccDate], dx
14647                                   <1>
14648 000122A7 58                       <1>       pop    eax ; *
14649                                   <1>
14650 000122A8 BB[9C090300]             <1>       mov    ebx, rw_buffer ; Common r/w sector buffer addr
14651 000122AD B901000000               <1>       mov    ecx, 1
14652                                   <1>       ; esi = logical dos description table address
14653                                   <1>       ; eax = disk sector number/address (LBA)
14654                                   <1>       ; ecx = sector count
14655                                   <1>       ; ebx = buffer address
14656 000122B2 E885080000               <1>       call   disk_write
14657 000122B7 0F8234FFFFFF             <1>       jc     uflmdt_2
14658                                   <1>
14659                                   <1>       ; save directory buffer if has modified/changed sign
14660 000122BD E8F497FFFF               <1>       call   save_directory_buffer
14661 000122C2 C3                       <1>       retn
14662                                   <1>
14663                                   <1> uflmdt_11:
14664                                   <1>       ; 24/10/2016
14665                                   <1>       ; Update last modification date & time of a file
14666                                   <1>       ; on a disk with Singlix File System.
14667                                   <1>       ;
14668                                   <1>       ; (Method: Read the FDT -File Description Table-
14669                                   <1>       ; sector of the file and update the lmdt data fields,
14670                                   <1>       ; then write FDT sector to the disk.
14671                                   <1>       ; /// It is easy but there is compatibility buffer
14672                                   <1>       ; method also for changing directory entry data and
14673                                   <1>       ; also there are some programming issues for Singlix
14674                                   <1>       ; file system (TRFS), which are not completed yet!)
14675                                   <1>       ;
14676                                   <1>       ; Not ready yet ! (24/10/2016)
14677                                   <1>       ; /// Temporary code for error return ! ///
14678 000122C3 31C0                     <1>       xor    eax, eax
14679 000122C5 F9                       <1>       stc
14680 000122C6 C3                       <1>       retn
14681                                   <1>
14682                                   <1> sysalloc:
14683                                   <1>       ; 14/10/2017
14684                                   <1>       ; 20/08/2017, 01/09/2017
14685                                   <1>       ; 20/02/2017, 04/03/2017, 15/05/2017
14686                                   <1>       ; 19/02/2017 - TRDOS 386 (TRDOS v2.0)
14687                                   <1>       ; (TRDOS 386 feature only!)
14688                                   <1>       ;
14689                                   <1>       ; Allocate Contiguous Memory Block/Pages (for user)
14690                                   <1>       ; (System call for DMA Buffer allocation etc.)
14691                                   <1>       ;
14692                                   <1>       ; INPUT ->
14693                                   <1>       ;     EBX = Virtual address (for user)
14694                                   <1>       ;          (Physical memory block/aperture
14695                                   <1>       ;          will be mapped to this virtual address)
14696                                   <1>       ;     ECX = Byte Count
14697                                   <1>       ;          (will be rounded up to page border)
14698                                   <1>       ;     If ECX = 0
14699                                   <1>       ;         System call will return with an error (cf=1)
14700                                   <1>       ;         but ECX will contain maximum size of
14701                                   <1>       ;         available memory aperture and physical
14702                                   <1>       ;         (beginning) address of that aperture
14703                                   <1>       ;         (which have maximum size) will be in EAX.
14704                                   <1>       ;     EDX = Upper limit of the requested physical memory
14705                                   <1>       ;           block/pages.
14706                                   <1>       ;          (The last byte address of the memory aperture
14707                                   <1>       ;           must not be equal to or above this limit.)
14708                                   <1>       ;     If EDX = 0
14709                                   <1>       ;        there is NOLIMIT !
14710                                   <1>       ;     If EDX = 0FFFFFFFFh (-1)
14711                                   <1>       ;        ESI = Lower Limit !
14712                                   <1>       ;            (Beginning of the block must not be 'less'
14713                                   <1>       ;            than this.) (Must be equal to or above...)
14714                                   <1>       ;        EDI = Upper Limit !
14715                                   <1>       ;            (End of the block must be !less! than this)
14716                                   <1>       ;            (The last byte addr of the memory aperture
14717                                   <1>       ;            must not be equal to or above this limit.)
14718                                   <1>       ;
14719                                   <1>       ; OUTPUT ->
14720                                   <1>       ;     If CF = 0
14721                                   <1>       ;     EAX = Physical address of the allocated memory block
14722                                   <1>       ;     ECX = Allocated bytes (as rounded up to page borders)
14723                                   <1>       ;     EBX = Virtual address (as rounded up)
14724                                   <1>       ;     IF CF = 1
14725                                   <1>       ;         Requested (size of) Memory block could not be
14726                                   <1>       ;         allocated to the user!
14727                                   <1>       ;     IF CF = 1 & EAX = 0 (Insufficient memory error!)
14728                                   <1>       ;        ECX = Total number of free bytes
14729                                   <1>       ;              (not size of available contiguous bytes!)
14730                                   <1>       ;     If CF = 1 & EAX > 0
14731                                   <1>       ;        there is not a memory aperture with requested size
14732                                   <1>       ;        but total free mem is not less than requested size.
14733                                   <1>       ;        EAX = Physical addr of available memory aperture
14734                                   <1>       ;             with max size
14735                                   <1>       ;              (but it doesn't fit to the conditions!)
14736                                   <1>       ;        ECX = Size of available memory aperture in bytes.
14737                                   <1>       ;     If CF = 1 -> EAX = 0FFFFFFFFh
14738                                   <1>       ;        Conditions/Parameters are wrong !
14739                                   <1>       ;        ECX is same with input value.
14740                                   <1>       ;
14741                                   <1>       ; Note:      Previously allocated pages will be deallocated if
14742                                   <1>       ;      new allocation conditions are met.
14743                                   <1>       ;
14744                                   <1>       ; Note: u.break control may be included in future versions
```

```
14745                                  <1>       ;
14746                                  <1>
14747 000122C7 31C0                    <1>       xor    eax, eax ; 0
14748                                  <1>       ; 14/10/2017
14749 000122C9 4A                      <1>       dec    edx ; is there a limit ?
14750 000122CA 7810                    <1>       js     short sysalloc_1 ;  0 -> 0FFFFFFFFh -> NO LIMIT
14751 000122CC 42                      <1>       inc    edx ; > 0
14752                                  <1>       ; Check upper address limit
14753                                  <1>       ;(round up to page borders)
14754 000122CD 81C1FF0F0000            <1>       add    ecx, PAGE_SIZE-1 ; 4095
14755 000122D3 6681E100F0              <1>       and    cx, ~PAGE_OFF ; not 4095
14756 000122D8 39CA                    <1>       cmp    edx, ecx ; upper limit - block size
14757 000122DA 7224                    <1>       jb     short sysalloc_err
14758                                  <1> sysalloc_1:
14759                                  <1>       ; EAX = Beginning address (physical)
14760                                  <1>       ; EAX = 0 -> Allocate mem block from the 1st proper aperture
14761                                  <1>       ; ECX = Number of bytes to be allocated
14762 000122DC E89C41FFFF              <1>       call   allocate_memory_block
14763 000122E1 721D                    <1>       jc     short sysalloc_err
14764                                  <1>       ; 01/09/2017
14765 000122E3 29C2                    <1>       sub    edx, eax ; upper limit address - beginning address
14766 000122E5 760F                    <1>       jna    short sysalloc_3 ; begin addr not less than the limit
14767 000122E7 39CA                    <1>       cmp    edx, ecx
14768 000122E9 720B                    <1>       jb     short sysalloc_3 ; end address overs the limit
14769                                  <1> sysalloc_2:
14770                                  <1>       ; EAX = Beginning (physical) addr of the allocated mem block
14771                                  <1>       ; ECX = Num of allocated bytes (rounded up to page borders)
14772 000122EB 50                      <1>       push   eax ; * ; 04/03/2017
14773                                  <1>       ; Here, requested contiquous memory pages have been allocated
14774                                  <1>       ; on Memory Allocation Table but user's page directory
14775                                  <1>       ; and page tables have not been updated yet!
14776 000122EC 51                      <1>       push   ecx ; **
14777                                  <1>       ; ebx = virtual address (will be rounded up to page border)
14778                                  <1>       ; ecx = number of bytes to be deallocated
14779                                  <1>       ;      will be adjusted to ebx+ecx round down - ebx round up
14780 000122ED E8E644FFFF              <1>       call   deallocate_user_pages
14781 000122F2 731F                    <1>       jnc    short sysalloc_4 ; EAX = Deallocated memory bytes
14782 000122F4 59                      <1>       pop    ecx ; **
14783 000122F5 58                      <1>       pop    eax ; *
14784                                  <1> sysalloc_3:
14785                                  <1>       ; error !
14786                                  <1>       ; restore Memory Allocation Table Content
14787 000122F6 E88F43FFFF              <1>       call   deallocate_memory_block
14788 000122FB 31C0                    <1>       xor    eax, eax ; 0
14789 000122FD 48                      <1>       dec    eax ; 0FFFFFFFFh ; 15/05/2017
14790 000122FE EB09                    <1>       jmp    short sysalloc_wrong
14791                                  <1> sysalloc_err:
14792 00012300 8B2D[60030300]          <1>       mov    ebp, [u.usp]  ; ebp points to user's registers
14793 00012306 894D18                  <1>       mov    [ebp+24], ecx ; return to user with ecx value
14794                                  <1> sysalloc_wrong:
14795                                  <1>       ; eax = 0FFFFFFFFh
14796 00012309 A3[64030300]            <1>       mov    [u.r0], eax
14797 0001230E E9C9B5FFFF              <1>       jmp    error
14798                                  <1> sysalloc_4:
14799 00012313 8B2D[60030300]          <1>       mov    ebp, [u.usp]  ; ebp points to user's registers
14800 00012319 894518                  <1>       mov    [ebp+24], eax ; return to user with ecx value
14801 0001231C 895D10                  <1>       mov    [ebp+16], ebx ; new value of ebx (rounded up)
14802 0001231F 89C1                    <1>       mov    ecx, eax ; byte count (from 'deallocate_user_pages')
14803 00012321 5A                      <1>       pop    edx ; ** ; discard (another) byte count
14804 00012322 58                      <1>       pop    eax ; *
14805 00012323 A3[64030300]            <1>       mov    [u.r0], eax ; physical address
14806                                  <1>
14807 00012328 51                      <1>       push   ecx ; 20/08/2017
14808                                  <1>       ;
14809                                  <1>       ; Write newly allocated contiguous (physical) pages
14810                                  <1>       ; on page dir and page tables of current user/process
14811                                  <1>       ; as PRESENT, USER, WRITABLE
14812                                  <1>       ; (then clear allocated pages)
14813 00012329 E89F45FFFF              <1>       call   allocate_user_pages
14814                                  <1>       ;jnc   sysret ; OK! return to process with success...
14815                                  <1>
14816                                  <1>       ; 20/08/2017 ('sysdma' modification)
14817 0001232E 59                      <1>       pop    ecx
14818 0001232F A1[64030300]            <1>       mov    eax, [u.r0]   ; physical address (of the block)
14819                                  <1>
14820 00012334 721D                    <1>       jc     short sysalloc_6
14821                                  <1>
14822 00012336 833D[84A00100]FF        <1>       cmp    dword [dma_addr], 0FFFFFFFFh ; -1
14823 0001233D 0F82B9B5FFFF            <1>       jb     sysret
14824                                  <1>
14825 00012343 A3[84A00100]            <1>       mov    [dma_addr], eax ; save dma address for sysdma
14826 00012348 890D[88A00100]          <1>       mov    [dma_size], ecx ; save dma buff size for sysdma
14827                                  <1>
14828 0001234E E9A9B5FFFF              <1>       jmp    sysret
14829                                  <1>
14830                                  <1> sysalloc_6:
14831                                  <1>       ;
14832                                  <1>       ; unexpected error ! insufficient memory !? conflict !?
14833                                  <1>       ; (!!?there is not a free page for a new page table?!!)
14834                                  <1>       ; We need to terminate process with error message !!!
14835                                  <1>       ;
14836 00012353 8B2D[60030300]          <1>       mov    ebp, [u.usp]  ; ebp points to user's registers
14837 00012359 8B4D18                  <1>       mov    ecx, [ebp+24] ; byte count
14838                                  <1>
14839                                  <1>       ; 20/08/2017
14840                                  <1>       ;mov   eax, [u.r0]   ; physical address (of the block)
14841                                  <1>
14842                                  <1>       ;
14843                                  <1>       ; restore Memory Allocation Table Content
14844 0001235C E82943FFFF              <1>       call   deallocate_memory_block
14845                                  <1>       ;
14846 00012361 803D[CA6F0000]03        <1>       cmp    byte [CRT_MODE], 3 ; 80x25 text mode?
14847 00012368 7407                    <1>       je     short sysalloc_7 ; yes
14848                                  <1>       ; Current mode is VGA (or CGA graphics) mode,
14849                                  <1>       ; We need to return to text mode for displaying
```

```
14850                                  <1>       ; error message just before 'sysexit'.
14851 0001236A B003                    <1>       mov    al, 3
14852 0001236C E8F7F7FEFF              <1>       call   _set_mode
14853                                  <1> sysalloc_7:
14854 00012371 BE[B2430100]            <1>       mov    esi, beep_Insufficient_Memory ; error message
14855 00012376 E8DA51FFFF              <1>       call   print_msg ; print/display the message
14856 0001237B B801000000              <1>       mov    eax, 1 ; ax=1 is needed for 'sysexit' procedure
14857 00012380 E9FEB6FFFF              <1>       jmp    sysexit ; and terminate the process !
14858                                  <1>
14859                                  <1> sysdalloc:
14860                                  <1>       ; 19/02/2017 - TRDOS 386 (TRDOS v2.0)
14861                                  <1>       ; (TRDOS 386 feature only!)
14862                                  <1>       ;
14863                                  <1>       ; Deallocate Memory Block/Pages (for user)
14864                                  <1>       ; (Complementary call for sysalloc.)
14865                                  <1>       ;
14866                                  <1>       ; INPUT ->
14867                                  <1>       ;     EBX = Virtual address (for user)
14868                                  <1>       ;           (will be rounded up to page border)
14869                                  <1>       ;     ECX = Byte Count
14870                                  <1>       ;          (will be adjusted to page borders)
14871                                  <1>       ;     If ICX = 0
14872                                  <1>       ;        nothing to do
14873                                  <1>       ;     If EBX + ECX > User's ESP
14874                                  <1>       ;        nothing to do
14875                                  <1>       ;
14876                                  <1>       ; Note: u.break control may be included in future versions
14877                                  <1>       ;
14878                                  <1>       ; OUTPUT ->
14879                                  <1>       ;     If CF = 0
14880                                  <1>       ;        EAX = Deallocated memory bytes
14881                                  <1>       ;        EBX = Virtual address (as rounded up)
14882                                  <1>       ;     If CF = 1
14883                                  <1>       ;        EAX = 0
14884                                  <1>       ;
14885                                  <1>       ; Note:      Main purpose of this call is to deallocate/release
14886                                  <1>       ;     previously allocated (physically) contiguous memory
14887                                  <1>       ;     pages but beginning (virtual) address may not be
14888                                  <1>       ;     followed by physically contiguous pages. So, this
14889                                  <1>       ;     system call will deallocate user's virtually
14890                                  <1>       ;     contiguous memory pages. Also, there is not any
14891                                  <1>       ;     objections to use this system call without sysalloc
14892                                  <1>       ;     system call; only possible objection is to lost data
14893                                  <1>       ;     within user's memory space, if the beginning address
14894                                  <1>       ;     and size is not proper.
14895                                  <1>       ;
14896                                  <1>       ; Note: Empty page tables will not be deallocated!!!
14897                                  <1>       ;       (they will be deallocated at process termination)
14898                                  <1>       ;
14899                                  <1>       ; Note: When the program terminates itself or when it is
14900                                  <1>       ;     terminated by operating system kernel, all allocated
14901                                  <1>       ;     memory pages will be deallocated during termination
14902                                  <1>       ;     stage. So, 'sysdalloc' is not necessary except
14903                                  <1>       ;     forgiving memory block to other programs/processes.
14904                                  <1>       ;
14905 00012385 8B15[5C030300]          <1>       mov    edx, [u.sp]
14906 0001238B 8B420C                  <1>       mov    eax, [edx+12] ; user's stack pointer
14907 0001238E 29C8                    <1>       sub    eax, ecx ; esp - byte count
14908 00012390 24FC                    <1>       and    al, 0FCh ; dword alignment
14909 00012392 39D8                    <1>       cmp    eax, ebx
14910 00012394 7220                    <1>       jb     short sysdalloc_err ; deallocation overlaps with stack
14911                                  <1>
14912 00012396 31C0                    <1>       xor    eax, eax
14913 00012398 21C9                    <1>       and    ecx, ecx
14914 0001239A 7407                    <1>       jz     short sysdalloc_2
14915                                  <1>
14916 0001239C E83744FFFF              <1>       call   deallocate_user_pages
14917 000123A1 7213                    <1>       jc     short sysdalloc_err
14918                                  <1>
14919                                  <1> sysdalloc_2:
14920 000123A3 A3[64030300]            <1>       mov    [u.r0], eax
14921 000123A8 8B2D[60030300]          <1>       mov    ebp, [u.usp]
14922 000123AE 895D10                  <1>       mov    [ebp+16], ebx ; new value of ebx
14923 000123B1 E946B5FFFF              <1>       jmp    sysret
14924                                  <1>
14925                                  <1> sysdalloc_err:
14926 000123B6 A3[64030300]            <1>       mov    [u.r0], eax ; 0
14927 000123BB E91CB5FFFF              <1>       jmp    error
14928                                  <1>
14929                                  <1> syscalbac:
14930                                  <1>       ; SYS CALLBACK
14931                                  <1>       ; 03/08/2020
14932                                  <1>       ; 16/04/2017
14933                                  <1>       ; 14/04/2017
14934                                  <1>       ; 13/04/2017
14935                                  <1>       ; 28/02/2017
14936                                  <1>       ; 26/02/2017
14937                                  <1>       ; 24/02/2017
14938                                  <1>       ; 21/02/2017 - TRDOS 386 (TRDOS v2.0)
14939                                  <1>       ; (TRDOS 386 feature only!)
14940                                  <1>       ;
14941                                  <1>       ; Link or unlink IRQ callback service to/from user (ring 3)
14942                                  <1>       ;
14943                                  <1>       ; INPUT ->
14944                                  <1>       ;     BL = IRQ number (Hardware interrupt request number)
14945                                  <1>       ;          (0 t0 15 but IRQ 0,1,2,6,8,14,15 are prohibited)
14946                                  <1>       ;          IRQ numbers 3,4,5,7,9,10,11,12,13 are valid
14947                                  <1>       ;          (numbers >15 are invalid)
14948                                  <1>       ;
14949                                  <1>       ;     BH = 0 = Unlink IRQ (in BL) from user (ring 3) service
14950                                  <1>       ;          1 = Link IRQ by using Signal Response Byte method
14951                                  <1>       ;          2 = Link IRQ by using Callback service method
14952                                  <1>       ;          3 = Link IRQ by using Auto Increment S.R.B. method
14953                                  <1>       ;         >3 = invalid
14954                                  <1>       ;
```

```
14955                                  <1>       ;      CL = Signal Return/Response Byte value
14956                                  <1>       ;
14957                                  <1>       ;      If BH = 3, kernel will put a counter value  ; 03/08/2020
14958                                  <1>       ;                (into the S.R.B. addr)
14959                                  <1>       ;              between 0 to 255. (start value = CL+1)
14960                                  <1>       ;
14961                                  <1>       ;      NOTE: counter value, for example: even and odd numbers
14962                                  <1>       ;            may be used for -audio- DMA buffer switch
14963                                  <1>       ;            within double buffer method, etc.
14964                                  <1>       ;
14965                                  <1>       ;      EDX = Signal return (Response) byte address
14966                                  <1>       ;                           - or -
14967                                  <1>       ;            Interrupt/Callback service/routine address
14968                                  <1>       ;
14969                                  <1>       ;            (virtual address in user's memory space)
14970                                  <1>       ;
14971                                  <1>       ; OUTPUT ->
14972                                  <1>       ;      CF = 0 & EAX = 0 -> Successful setting
14973                                  <1>       ;      CF = 1 & EAX > 0 -> IRQ is prohibited or locked
14974                                  <1>       ;                    by another process
14975                                  <1>       ;              eax = ERR_PERM_DENIED -> prohibited or locked
14976                                  <1>       ;              eax = ERR_INV_PARAMETER ->
14977                                  <1>       ;                    invalid parameter/option or bad address
14978                                  <1>       ;
14979                                  <1>       ;      NOTE: Timer callbacks are set by using 'systimer'
14980                                  <1>       ;            system call (IRQ 0, PIT and IRQ 8, RTC)
14981                                  <1>       ;
14982                                  <1>       ;            Direct keyboard access is performed by using
14983                                  <1>       ;            Keyboard Interrupt (INT 32h)
14984                                  <1>       ;
14985                                  <1>       ;            It is prohibited here because:
14986                                  <1>       ;             1) Signal Response Byte method has not advantage
14987                                  <1>       ;                against INT 32h, function AH = 1. Also,
14988                                  <1>       ;                keyboard service interrupt will return with
14989                                  <1>       ;                ascii and scan codes (AL, AH) while
14990                                  <1>       ;                SRB method has only 1 byte space for ascii code
14991                                  <1>       ;                or scan code. One byte signal response is used
14992                                  <1>       ;                for ensuring very simple and very fast
14993                                  <1>       ;                virtual to physical memory address conversion
14994                                  <1>       ;                without any memory page crossover risk.
14995                                  <1>       ;                (Otherwise double page conversion or word
14996                                  <1>       ;                alignment would be needed.)
14997                                  <1>       ;             2) Badly written user code (callback code)
14998                                  <1>       ;                can prevent keyboard and timesharing functions
14999                                  <1>       ;                of the operating system via continuous and long
15000                                  <1>       ;                keyboard event handling by callback service.
15001                                  <1>       ;                (It can cause to lose immediate keystroke
15002                                  <1>       ;                response from hardware to user.)
15003                                  <1>       ;             3) If user will check any keyboard events, 'getkey'
15004                                  <1>       ;                (or 'getchar') must have more priority than other
15005                                  <1>       ;                (video etc.) events because only control ability
15006                                  <1>       ;                on a procedural infinite loop is a keyboard or
15007                                  <1>       ;                mouse event. So user can use keyboard function
15008                                  <1>       ;                at the end or at the beginning of a loop.
15009                                  <1>       ;                In this case, INT 32h is used for that purpose
15010                                  <1>       ;                and timer interrupt etc. callbacks can be used
15011                                  <1>       ;                for dynamic and synchronized data refresh/transfer
15012                                  <1>       ;                while cpu is in a static loop (without polling).
15013                                  <1>       ;                Keyboard Int callback is not more useful because
15014                                  <1>       ;                already a manual check (a key is pressed or not)
15015                                  <1>       ;                can be performed (via INT 32h, AH = 1) efficiently
15016                                  <1>       ;                in a loop to prevent a locked infinitive loop.
15017                                  <1>       ;
15018                                  <1>       ;           Disk IRQs (6,14,15) have been phohibited from ring 3
15019                                  <1>       ;           callback because, disk operations (file system services
15020                                  <1>       ;           etc.) are independent from user program, for fast disk r/w.
15021                                  <1>       ;           They are not more useful at ring 3 while they are in use
15022                                  <1>       ;           by standard diskio functions which are mandatory part of
15023                                  <1>       ;           (monolithic) OS kernel and mainprog command interpreter.
15024                                  <1>       ;           INT 33h diskio functions are enough for user level disk
15025                                  <1>       ;           r/w.
15026                                  <1>       ;
15027                                  <1>       ; TRDOS 386 - IRQ CALLBACK structures (parameters):
15028                                  <1>       ;
15029                                  <1>       ;           [u.irqlock] = 1 word, IRQ flags (0-15) that indicates
15030                                  <1>       ;                     which IRQs are locked by (that) user.
15031                                  <1>       ;                      Lock and unlock (by user) will change
15032                                  <1>       ;                     these flags or 'terminate process' (sysexit)
15033                                  <1>       ;                     will clear these flags and unlock those IRQs.
15034                                  <1>       ;
15035                                  <1>       ;                     Bit 0 is for IRQ 0 and Bit 15 is for IRQ 15
15036                                  <1>       ;
15037                                  <1>       ;           IRQ(x).owner      : 1 byte, user, [u.uno], 0 = free (unlocked)
15038                                  <1>       ;
15039                                  <1>       ;           IRQ(x).method : 1 byte for callback method & status
15040                                  <1>       ;                       0 = Signal Response Byte method
15041                                  <1>       ;                       1 = Callback service method
15042                                  <1>       ;                       >1 = invalid for current 'syscalback'.
15043                                  <1>       ;                     or(+) 80h = IRQ is in use by system (ring 0)
15044                                  <1>       ;                                 function (audio etc.) or
15045                                  <1>       ;                               a device driver.
15046                                  <1>       ;                     (system function will ignore the lock/owner)
15047                                  <1>       ;
15048                                  <1>       ;           IRQ(x).srb: 1 byte, Signal Return/Response byte value
15049                                  <1>       ;                       (a fixed value by user or a counter value
15050                                  <1>       ;                      from 0 to 255, which is increased by every
15051                                  <1>       ;                      interrupt just before putting it into
15052                                  <1>       ;                      the Signal Response byte address
15053                                  <1>       ;                      (This is not used in callback serv method)
15054                                  <1>       ;
15055                                  <1>       ;           IRQ(x).addr      : 1 dword
15056                                  <1>       ;                       Signal Response Byte address (physical)
15057                                  <1>       ;                                 -or-
15058                                  <1>       ;                       Callback service address (virtual)
15059                                  <1>       ;
```

```
15060                                  <1>       ;         IRQ(x).dev: 1 byte
15061                                  <1>       ;                     0 = Default device or kernel function
15062                                  <1>       ;                                 -or-
15063                                  <1>       ;                     1-255 = Assigned device driver number
15064                                  <1>       ;
15065                                  <1>       ;         (x) = 3,4,5,7,9,10,11,12,13
15066                                  <1>       ;
15067                                  <1>       ;
15068                                  <1>       ;      NOTE: If user's process/program calls the kernel (INT 40h)
15069                                  <1>       ;            while it is already running in a (ring 3) callback
15070                                  <1>       ;            service, kernel will force (convert) system call to
15071                                  <1>       ;            'sysrele' (sys release). So, this feature provides
15072                                  <1>       ;            easy and simple usage of callback services without
15073                                  <1>       ;            falling into deepless <please 'callback me' then
15074                                  <1>       ;            let me 'callback you'> cycles! (User must return
15075                                  <1>       ;            from callback service by using 'sysrele' system
15076                                  <1>       ;            call, without a significant delay. Otherwise user
15077                                  <1>       ;            process/program may be late to catch the next event
15078                                  <1>       ;            within same callback purpose.
15079                                  <1>       ;
15080                                  <1>
15081 000123C0 30C0                   <1>       xor    al, al ; the caller is 'syscalbac' sign/flag
15082 000123C2 E85F180000             <1>       call   set_irq_callback_service
15083                                  <1>       ; 16/04/2017
15084 000123C7 A3[64030300]           <1>       mov    [u.r0], eax
15085 000123CC 0F832AB5FFFF           <1>       jnc    sysret
15086 000123D2 A3[C8030300]           <1>       mov    dword [u.error], eax
15087 000123D7 E900B5FFFF             <1>       jmp    error
15088                                  <1>
15089                                  <1> sysfpstat:
15090                                  <1>       ; 28/02/2017 - TRDOS 386 (TRDOS v2.0)
15091                                  <1>       ; (TRDOS 386 feature only!)
15092                                  <1>       ;
15093                                  <1>       ; Set or reset FPU registers save/restore option (for user)
15094                                  <1>       ;             (during software task switching, wswap-rswap)
15095                                  <1>       ;
15096                                  <1>       ; INPUT ->
15097                                  <1>       ;     BL = 0 -> reset
15098                                  <1>       ;     BL = 1 -> set (FPU register will be saved and restored)
15099                                  <1>       ;
15100                                  <1>       ; OUTPUT ->
15101                                  <1>       ;     cf = 0 -> no error, FPU is ready...
15102                                  <1>       ;               (EAX = 0)
15103                                  <1>       ;     Cf = 1 -> error, 80387 FPU is not ready !
15104                                  <1>       ;               (EAX = 0FFFFFFFFh)
15105                                  <1>
15106 000123DC 31C0                   <1>       xor    eax, eax
15107 000123DE 803D[7C960100]00       <1>       cmp    byte [fpready], 0
15108 000123E5 7613                   <1>       jna    short sysfpstat_err
15109                                  <1>
15110 000123E7 80E301                 <1>       and    bl, 1 ; use BIT 0 only !
15111 000123EA 881D[DA030300]         <1>       mov    [u.fpsave], bl
15112 000123F0 A3[64030300]           <1>       mov    [u.r0], eax ; 0
15113 000123F5 E902B5FFFF             <1>       jmp    sysret
15114                                  <1>
15115                                  <1> sysfpstat_err:
15116 000123FA 48                     <1>       dec    eax ; 0FFFFFFFFh
15117 000123FB A3[64030300]           <1>       mov    [u.r0], eax ; -1
15118 00012400 E9D7B4FFFF             <1>       jmp    error
15119                                  <1>
15120                                  <1> sysdelete: ; Delete (Remove, Unlink) File
15121                                  <1>       ; 29/12/2017 (TRDOS 386 = TRDOS v2.0)
15122                                  <1>       ;
15123                                  <1>         ; INPUT ->
15124                                  <1>         ;          EBX = File name (ASCIIZ string) address
15125                                  <1>       ; OUTPUT ->
15126                                  <1>       ;          cf = 0 -> eax = 0
15127                                  <1>       ;          cf = 1 -> Error code in AL
15128                                  <1>       ;
15129                                  <1>       ; Modified Registers: EAX (at the return of system call)
15130                                  <1>       ;
15131                                  <1>
15132 00012405 89DE                   <1>       mov    esi, ebx
15133                                  <1>       ; file name is forced, change directory as temporary
15134                                  <1>       ;mov   ax, 1
15135                                  <1>       ;mov   [FFF_Valid], ah ; 0 ; reset
15136                                  <1>       ;call  set_working_path
15137 00012407 E8620B0000             <1>       call   set_working_path_x
15138 0001240C 731D                   <1>       jnc    short sysdelete_1
15139                                  <1>
15140 0001240E 21C0                   <1>       and    eax, eax  ; 0 -> Bad Path!
15141 00012410 7505                   <1>       jnz    short sysdelete_err
15142                                  <1>       ; eax = 0
15143                                  <1> sysdelete_path_err:
15144 00012412 B813000000             <1>       mov    eax, ERR_INV_PATH_NAME ; 'bad path name !'
15145                                  <1> sysdelete_err:
15146 00012417 A3[64030300]           <1>       mov    [u.r0], eax
15147 0001241C A3[C8030300]           <1>       mov    [u.error], eax
15148 00012421 E81D0C0000             <1>       call   reset_working_path
15149 00012426 E9B1B4FFFF             <1>       jmp    error
15150                                  <1> sysdelete_1:
15151                                  <1>       ;mov   esi, FindFile_Name
15152 0001242B 66B80018               <1>       mov    ax, 1800h ; Only files
15153 0001242F E84D70FFFF             <1>       call   find_first_file
15154 00012434 72E1                   <1>       jc     short sysdelete_err
15155                                  <1> sysdelete_2:
15156                                  <1>       ; check file attributes
15157                                  <1>
15158                                  <1>       ;test  bl, 17 ; system, hidden, readonly, directory
15159 00012436 F6C307                 <1>         test bl, 7 ; system, hidden, readonly
15160 00012439 7407                   <1>         jz   short sysdelete_3
15161                                  <1>
15162 0001243B B80B000000             <1>         mov  eax, ERR_FILE_ACCESS ; 11 = 'permission denied !'
15163 00012440 EBD5                   <1>         jmp  short sysdelete_err
15164                                  <1> sysdelete_3:
```

```
15165 00012442 6621D2              <1>       and    dx, dx ; Ambiguous filename chars used sign (DX>0)
15166 00012445 7407                <1>       jz     short sysdelete_4
15167 00012447 B81A000000          <1>       mov    eax, ERR_INV_FILE_NAME ; 26 = 'invalid file name !'
15168 0001244C EBC9                <1>         jmp  short sysdelete_err
15169                              <1> sysdelete_4:
15170                              <1>       ;mov   bh, [LongName_EntryLength]
15171 0001244E 883D[EE930100]      <1>       mov    [DelFile_LNEL], bh ; Long name entry length (if > 0)
15172                              <1>       ; edi = Directory Entry Offset (DirBuff)
15173                              <1>       ; esi = Directory Entry (FFF Structure)
15174 00012454 E8A599FFFF          <1>       call   remove_file
15175 00012459 72BC                <1>       jc     short sysdelete_err
15176                              <1> sysrmdir_5:
15177 0001245B 31C0                <1>       xor    eax, eax ; 0
15178 0001245D A3[64030300]        <1>       mov    [u.r0], eax
15179                              <1>       ;mov   [u.error], eax
15180 00012462 E8DC0B0000          <1>       call   reset_working_path
15181 00012467 E990B4FFFF          <1>       jmp    sysret
15182                              <1>
15183                              <1>
15184                              <1> sysrmdir: ; Remove (Unlink) Directory
15185                              <1>       ; 19/01/2021
15186                              <1>       ; 29/12/2017 (TRDOS 386 = TRDOS v2.0)
15187                              <1>       ;
15188                              <1>         ; INPUT ->
15189                              <1>         ;          EBX = Pointer to directory name
15190                              <1>       ; OUTPUT ->
15191                              <1>       ;          cf = 0 -> eax = 0
15192                              <1>       ;          cf = 1 -> Error code in AL
15193                              <1>       ;
15194                              <1>       ; Modified Registers: EAX (at the return of system call)
15195                              <1>       ;
15196                              <1>
15197                              <1>       ; 19/01/2021
15198 0001246C 803D[B0030300]00    <1>       cmp    byte [u.uid], 0  ; root (super user) ?
15199 00012473 7614                <1>       jna    short sysrmdir_0
15200                              <1>
15201                              <1>       ;mov   dword [u.r0], ERR_PERM_DENIED
15202 00012475 B80B000000          <1>       mov    eax, ERR_PERM_DENIED ; ERR_NOT_SUPERUSER
15203 0001247A A3[64030300]        <1>       mov    [u.r0], eax
15204 0001247F A3[C8030300]        <1>       mov    [u.error], eax
15205 00012484 E953B4FFFF          <1>       jmp    error
15206                              <1>
15207                              <1> sysrmdir_0:
15208 00012489 89DE                <1>       mov    esi, ebx
15209                              <1>       ; file name is forced, change directory as temporary
15210                              <1>       ;mov   ax, 1
15211                              <1>       ;mov   [FFF_Valid], ah ; 0 ; reset
15212                              <1>       ;call  set_working_path
15213 0001248B E8DE0A0000          <1>       call   set_working_path_x
15214 00012490 731D                <1>       jnc    short sysrmdir_1
15215                              <1>
15216 00012492 21C0                <1>       and    eax, eax  ; 0 -> Bad Path!
15217 00012494 7505                <1>       jnz    short sysrmdir_err
15218                              <1>       ; eax = 0
15219                              <1> sysrmdir_not_found:
15220 00012496 B80C000000          <1>       mov    eax, ERR_DIR_NOT_FOUND ; Directory not found !
15221                              <1> sysrmdir_err:
15222 0001249B A3[64030300]        <1>       mov    [u.r0], eax
15223 000124A0 A3[C8030300]        <1>       mov    [u.error], eax
15224 000124A5 E8990B0000          <1>       call   reset_working_path
15225 000124AA E92DB4FFFF          <1>       jmp    error
15226                              <1> sysrmdir_1:
15227                              <1>       ;mov   esi, FindFile_Name
15228 000124AF 66B81008            <1>       mov    ax, 0810h ; Only directories
15229 000124B3 E8C96FFFFF          <1>       call   find_first_file
15230 000124B8 7306                <1>       jnc    short sysrmdir_2
15231                              <1>
15232                              <1>       ; eax = 2 (File not found !)
15233 000124BA 3C02                <1>       cmp    al, 2 ; ERR_NOT_FOUND
15234 000124BC 74D8                <1>       je     short sysrmdir_not_found
15235 000124BE EBDB                <1>       jmp    short sysrmdir_err
15236                              <1> sysrmdir_2:
15237                              <1>       ; check directory attributes
15238                              <1>
15239 000124C0 F6C307              <1>         test bl, 7 ; system, hidden, readonly
15240 000124C3 7407                <1>         jz   short sysrmdir_3
15241                              <1>
15242 000124C5 B80B000000          <1>         mov  eax, ERR_DIR_ACCESS ; 11 = 'permission denied !'
15243 000124CA EBCF                <1>         jmp  short sysrmdir_err
15244                              <1> sysrmdir_3:
15245 000124CC 6621D2              <1>       and    dx, dx ; Ambiguous filename chars used sign (DX>0)
15246 000124CF 7407                <1>       jz     short sysrmdir_4
15247                              <1>       ;mov   eax, ERR_NOT_DIR ; 'not a valid directory !'
15248 000124D1 B813000000          <1>       mov    eax, ERR_INV_PATH_NAME ; 'bad path name !'
15249 000124D6 EBC3                <1>         jmp  short sysrmdir_err
15250                              <1> sysrmdir_4:
15251                              <1>       ;mov   bh, [LongName_EntryLength]
15252 000124D8 883D[EE930100]      <1>       mov    [DelFile_LNEL], bh ; Long name entry length (if > 0)
15253                              <1>       ; edi = Directory Entry Offset (DirBuff)
15254                              <1>       ; esi = Directory Entry (FFF Structure)
15255 000124DE E87376FFFF          <1>       call   delete_sub_directory
15256 000124E3 0F8372FFFFFF        <1>       jnc    sysrmdir_5
15257                              <1> ;     jc     short sysrmdir_6
15258                              <1> ;
15259                              <1> ;     xor    eax, eax ; 0
15260                              <1> ;sysrmdir_5:
15261                              <1> ;     mov    [u.r0], eax
15262                              <1> ;     ;mov   [u.error], eax
15263                              <1> ;     call   reset_working_path
15264                              <1> ;     jmp    sysret
15265                              <1> sysrmdir_6:
15266 000124E9 A3[64030300]        <1>       mov    [u.r0], eax
15267 000124EE A3[C8030300]        <1>       mov    [u.error], eax
15268                              <1>
15269 000124F3 09C0                <1>       or     eax, eax ; EAX = 0 -> Directory not empty!
```

```
15270 000124F5 741C                <1>       jz     short sysrmdir_9
15271                              <1>
15272                              <1>       ; EAX > 0 -> Error code in AL (or AX or EAX)
15273                              <1>
15274 000124F7 833D[A2910100]01    <1>       cmp    dword [FAT_ClusterCounter], 1
15275 000124FE 7209                <1>       jb     short sysrmdir_8
15276                              <1> sysrmdir_7:
15277                              <1>       ; ESI = Logical DOS Drive Description Table address
15278 00012500 66BB00FF            <1>       mov    bx, 0FF00h ; BH = FFh -> use ESI for Drive parameters
15279                              <1>                   ; BL = 0 -> Recalculate free cluster count
15280 00012504 E8D9AEFFFF          <1>       call   calculate_fat_freespace
15281                              <1> sysrmdir_8:
15282 00012509 E8350B0000          <1>       call   reset_working_path
15283 0001250E E9C9B3FFFF          <1>       jmp    error
15284                              <1>
15285                              <1> sysrmdir_9:
15286 00012513 A1[A2910100]        <1>       mov    eax, [FAT_ClusterCounter]
15287 00012518 09C0                <1>       or     eax, eax ; 0 ?
15288 0001251A 0F847BFFFFFF        <1>       jz     sysrmdir_err
15289                              <1>       ; ESI = Logical DOS Drive Description Table address
15290 00012520 66BB01FF            <1>       mov    bx, 0FF01h ; BH = FFh -> use ESI for Drive parameters
15291                              <1>                   ; BL = 1 -> add free clusters
15292 00012524 E8B9AEFFFF          <1>       call   calculate_fat_freespace
15293 00012529 09C9                <1>       or     ecx, ecx
15294 0001252B 74DC                <1>         jz   short sysrmdir_8 ; ecx = 0 -> OK
15295                              <1>       ; ecx > 0 -> Error (Recalculation is needed)
15296 0001252D EBD1                <1>       jmp    short sysrmdir_7
15297                              <1>
15298                              <1>
15299                              <1> syschdir: ; Change Current (Working) Drive & Directory (for user)
15300                              <1>       ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
15301                              <1>       ;
15302                              <1>         ; INPUT ->
15303                              <1>         ;          EBX = Directory name (ASCIIZ string) address
15304                              <1>       ; OUTPUT ->
15305                              <1>       ;          cf = 0 -> eax = 0
15306                              <1>       ;          cf = 1 -> Error code in AL
15307                              <1>       ;
15308                              <1>       ; Modified Registers: EAX (at the return of system call)
15309                              <1>       ;
15310                              <1>       ; NOTE: If drive name is not included, only the working
15311                              <1>       ; directory (for user, not for drive/OS) will be chanded.
15312                              <1>       ; If there is a drive name (as A:, B:, C:, D: etc.)
15313                              <1>       ; at the beginning of the ASCIIZ (directory) string,
15314                              <1>       ; working drive and working directory (for user)
15315                              <1>       ; will be changed together.
15316                              <1>       ; (When the program is terminated, MainProg -internal
15317                              <1>       ; shell- will reset working directory to the previous
15318                              <1>       ; -current- logical drive's current directory again.)
15319                              <1>
15320 0001252F 89DE                <1>       mov    esi, ebx
15321                              <1>       ; file name is not forced, change directory as temporary
15322 00012531 31C0                <1>       xor    eax, eax
15323                              <1>       ;mov   [FFF_Valid], ah ; 0 ; reset
15324                              <1>       ;call  set_working_path
15325 00012533 E83A0A0000          <1>       call   set_working_path_xx
15326 00012538 731D                <1>       jnc    short syschdir_ok
15327 0001253A 21C0                <1>       and    eax, eax  ; 0 -> Bad Path!
15328 0001253C 7505                <1>       jnz    short syschdir_err
15329                              <1>       ; eax = 0
15330                              <1> syschdir_not_found:
15331 0001253E B80C000000          <1>       mov    eax, ERR_DIR_NOT_FOUND ; Directory not found !
15332                              <1> syschdir_err:
15333 00012543 A3[64030300]        <1>       mov    [u.r0], eax
15334 00012548 A3[C8030300]        <1>       mov    [u.error], eax
15335 0001254D E8F10A0000          <1>       call   reset_working_path
15336 00012552 E985B3FFFF          <1>       jmp    error
15337                              <1> syschdir_ok:
15338 00012557 31C0                <1>       xor    eax, eax ; 0
15339 00012559 A3[64030300]        <1>       mov    [u.r0], eax
15340                              <1>       ;mov   [u.error], eax
15341 0001255E E999B3FFFF          <1>       jmp    sysret
15342                              <1>
15343                              <1>
15344                              <1> syschmod: ; Get & Change File (or Directory) Attributes
15345                              <1>       ; 19/01/2021
15346                              <1>       ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
15347                              <1>       ;
15348                              <1>         ; INPUT ->
15349                              <1>         ;          EBX = File/Directory (ASCIIZ) name address
15350                              <1>       ;          CL = New attributes (if CL < 40h)
15351                              <1>       ;          CL >= 40h -> Get File Attributes
15352                              <1>       ; OUTPUT ->
15353                              <1>       ;          cf = 0 -> EAX = File attributes (in AL)
15354                              <1>       ;          cf = 1 -> Error code in AL
15355                              <1>       ;
15356                              <1>       ; Modified Registers: EAX (at the return of system call)
15357                              <1>       ;
15358                              <1>       ; MSDOS File Attributes:    (bit value of attrib byte)
15359                              <1>       ;     ATTR_READ_ONLY      =      01h  (bit 0, 'R')
15360                              <1>       ;     ATTR_HIDDEN  =      02h  (bit 1, 'H')
15361                              <1>       ;     ATTR_SYSTEM  =      04h  (bit 2, 'S')
15362                              <1>       ;     ATTR_VOLUME_ID      =      08h  (bit 3)
15363                              <1>       ;     ATTR_DIRECTORY      =      10h  (bit 4)
15364                              <1>       ;     ATTR_ARCHIVE =      20h  (bit 5, 'A')
15365                              <1>       ;     ATTR_LONG_NAME      =      ATTR_READONLY |
15366                              <1>       ;                            ATTR_HIDDEN |
15367                              <1>       ;                            ATTR_SYSTEM |
15368                              <1>       ;                            ATTR_VOLUME_ID
15369                              <1>       ;     The upper two bits of attributes must be 0.
15370                              <1>
15371                              <1>       ; Note:      * If ATTR_DIRECTORY is set, only directory names
15372                              <1>       ;       will be searched (and S,H,R,A attributeds of
15373                              <1>       ;       the directory will be changed.)
15374                              <1>       ;     * If ATTR_VOLUME_ID is set, 'syschmod' system call
```

```
15375                                  <1>       ;       will return with 'permission denied' error.
15376                                  <1>       ;     * If ATTR_DIRECTORY is not set, only file names
15377                                  <1>       ;       will be searched (and S,H,R,A attributes of the
15378                                  <1>       ;       file will be changed.)
15379                                  <1>       ;
15380                                  <1>       ; (Ony Super User can change S,H,R attributes.)
15381                                  <1>
15382 00012563 80F940                 <1>       cmp   cl, 40h
15383 00012566 7327                   <1>       jnb   short syschmod_0
15384                                  <1>
15385 00012568 F6C108                 <1>       test  cl, 08h ; ATTR_VOLUME_ID
15386 0001256B 750E                   <1>       jnz   short syschmod_perm_err
15387                                  <1>
15388                                  <1>       ; 19/01/2021
15389 0001256D 803D[B0030300]00       <1>       cmp   byte [u.uid], 0  ; root (super user) ?
15390 00012574 7619                   <1>       jna   short syschmod_0
15391                                  <1>
15392                                  <1>       ; Not super user..
15393 00012576 F6C107                 <1>       test  cl, 07h      ; S,H,R attributes
15394 00012579 7414                   <1>       jz    short syschmod_0
15395                                  <1>
15396                                  <1> syschmod_perm_err:
15397                                  <1>       ;mov  dword [u.r0], ERR_PERM_DENIED
15398 0001257B B80B000000             <1>       mov   eax, ERR_PERM_DENIED ; 'permission denied !'
15399 00012580 A3[64030300]           <1>       mov   [u.r0], eax
15400 00012585 A3[C8030300]           <1>       mov   [u.error], eax
15401 0001258A E94DB3FFFF             <1>       jmp   error
15402                                  <1>
15403                                  <1> syschmod_0:
15404 0001258F 880D[3C940100]         <1>       mov   [Attributes], cl
15405 00012595 89DE                   <1>       mov   esi, ebx
15406                                  <1>       ; file name is forced, change directory as temporary
15407                                  <1>       ;mov  ax, 1
15408                                  <1>       ;mov  [FFF_Valid], ah ; 0 ; reset
15409                                  <1>       ;call set_working_path
15410 00012597 E8D2090000             <1>       call  set_working_path_x
15411 0001259C 731D                   <1>       jnc   short syschmod_1
15412 0001259E 21C0                   <1>       and   eax, eax  ; 0 -> Bad Path!
15413 000125A0 7505                   <1>       jnz   short syschmod_err
15414                                  <1>       ; eax = 0
15415                                  <1> syschmod_path_not_found:
15416 000125A2 B813000000             <1>       mov   eax, ERR_INV_PATH_NAME ; 'Bad path name !'
15417                                  <1> syschmod_err:
15418 000125A7 A3[64030300]           <1>       mov   [u.r0], eax
15419 000125AC A3[C8030300]           <1>       mov   [u.error], eax
15420 000125B1 E88D0A0000             <1>       call  reset_working_path
15421 000125B6 E921B3FFFF             <1>       jmp   error
15422                                  <1> syschmod_1:
15423 000125BB B008                   <1>       mov   al, 08h ; Except volume labels (& long names)
15424 000125BD A0[3C940100]           <1>       mov   al, [Attributes]
15425 000125C2 2410                   <1>       and   al, 10h ;
15426                                  <1>       ;mov  esi, FindFile_Name
15427                                  <1>       ;mov  ax, 1800h ; Only files
15428                                  <1>       ;mov  ax, 0810h ; Only directories
15429 000125C4 E8B86EFFFF             <1>       call  find_first_file
15430                                  <1>       ;jnc  short syschmod_2
15431 000125C9 72DC                   <1>       jc    short syschmod_err
15432                                  <1>
15433                                  <1>       ;; eax = 2 (File not found !)
15434                                  <1>       ;cmp  al, 2 ; ERR_NOT_FOUND
15435                                  <1>       ;jne  short syschmod_err
15436                                  <1>
15437                                  <1>       ;and  byte [Attributes], 10h
15438                                  <1>       ;jz   short syschmod_err
15439                                  <1>
15440                                  <1>       ;; Directory not found !
15441                                  <1>       ;mov  al, 3 ; ERR_PATH_NOT_FOUND
15442                                  <1>       ;jmp  short syschmod_err
15443                                  <1>
15444                                  <1> syschmod_2:
15445 000125CB 6621D2                 <1>       and   dx, dx ; Ambiguous filename chars used sign (DX>0)
15446 000125CE 7407                   <1>       jz    short syschmod_3
15447 000125D0 B81A000000             <1>       mov   eax, ERR_INV_FILE_NAME ; 'invalid file name !'
15448 000125D5 EBD0                   <1>         jmp short syschmod_err
15449                                  <1> syschmod_3:
15450                                  <1>       ; EDI = Directory buffer entry offset/address
15451                                  <1>       ; BL = File (or Directory) Attributes
15452                                  <1>       ; mov bl, [EDI+0Bh]
15453                                  <1>
15454                                  <1>       ; check directory attributes
15455 000125D7 8A3D[3C940100]         <1>       mov   bh, [Attributes] ; new attributes
15456 000125DD 80FF40                 <1>       cmp   bh, 40h  ;>=40 -> get file/directory attributes
15457 000125E0 732D                   <1>       jnb   short syschmod_6
15458                                  <1>
15459                                  <1>       ; set file/directory attributes
15460 000125E2 F6C307                 <1>       test  bl, 7 ; system, hidden, readonly
15461 000125E5 7409                   <1>         jz  short syschmod_4
15462                                  <1>
15463                                  <1>       ; 19/01/2021
15464 000125E7 803D[B0030300]00       <1>       cmp   byte [u.uid], 0  ; root (super user) ?
15465 000125EE 778B                   <1>       ja    short syschmod_perm_err
15466                                  <1> syschmod_4:
15467 000125F0 66817F0CA101           <1>       cmp   word [edi+DirEntry_NTRes], 01A1h ; Singlix FS
15468 000125F6 7424                   <1>       je    short syschmod_7
15469                                  <1>
15470 000125F8 887F0B                 <1>       mov   [edi+0Bh], bh    ; Attributes (New!)
15471                                  <1>
15472 000125FB C605[AC910100]02       <1>       mov   byte [DirBuff_ValidData], 2 ; modified sign
15473                                  <1>                                         ; to force write
15474 00012602 E8AF94FFFF             <1>       call  save_directory_buffer
15475 00012607 729E                   <1>       jc    short syschmod_err
15476                                  <1>
15477                                  <1> syschmod_5:
15478 00012609 8A1D[3C940100]         <1>       mov   bl, [Attributes]
15479                                  <1> syschmod_6:
```

```
15480 0001260F 0FB6C3              <1>       movzx  eax, bl
15481 00012612 A3[64030300]        <1>       mov    [u.r0], eax
15482                              <1>       ;mov   dword [u.error], 0
15483 00012617 E9E0B2FFFF          <1>       jmp    sysret
15484                              <1>
15485                              <1> syschmod_7:
15486 0001261C 29C0                <1>       sub    eax, eax
15487 0001261E 8A25[AA910100]      <1>         mov     ah, [DirBuff_DRV]
15488 00012624 BE00010900          <1>       mov    esi, Logical_DOSDisks
15489 00012629 01C6                <1>         add     esi, eax
15490 0001262B 807E04A1            <1>         cmp     byte [esi+LD_FSType], 0A1h
15491 0001262F 7307                <1>       jnc    short syschmod_8
15492 00012631 B01D                <1>       mov    al, ERR_INV_DATA ; 29 = Invalid Data
15493 00012633 E96FFFFFFF          <1>       jmp    syschmod_err
15494                              <1>
15495                              <1> syschmod_8:
15496                              <1>       ; BH = New MS-DOS File Attributes
15497 00012638 88F8                <1>       mov    al, bh ; File/Directory Attributes
15498 0001263A 30E4                <1>       xor    ah, ah ; Attributes in MS-DOS format sign
15499 0001263C E89E7FFFFF          <1>       call   change_fs_file_attributes
15500 00012641 0F8260FFFFFF        <1>       jc     syschmod_err
15501 00012647 EBC0                <1>       jmp    short syschmod_5
15502                              <1>
15503                              <1>
15504                              <1> sysdrive: ; Get/Set Current (Working) Drive (for user)
15505                              <1>       ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
15506                              <1>       ;
15507                              <1>         ; INPUT ->
15508                              <1>         ;          BL = Logical DOS Drive number (0=A: ... 2=C:)
15509                              <1>       ;         If BL = 0FFh -> Get Current Drive
15510                              <1>       ; OUTPUT ->
15511                              <1>       ;          cf = 0 ->
15512                              <1>       ;               AL = Current Drive number
15513                              <1>       ;               AH = The Last Logical DOS Drive no.
15514                              <1>       ;          cf = 1 -> Error code in AL
15515                              <1>       ;
15516                              <1>       ; Modified Registers: EAX (at the return of system call)
15517                              <1>       ;
15518                              <1>       ; NOTE: If the requested logical dos drive is ready,
15519                              <1>       ;     it's current current directory will be the user's
15520                              <1>       ;     (program's) current directory.
15521                              <1>       ;     (When the program is terminated, MainProg -internal
15522                              <1>       ;     shell- will reset the previous -current- logical drive
15523                              <1>       ;       as current drive again).
15524                              <1>
15525 00012649 80FBFF              <1>       cmp    bl, 0FFh
15526 0001264C 7435                <1>       je     short sysdrive_ok
15527 0001264E 3A1D[50400100]      <1>       cmp    bl, [Last_DOS_DiskNo]
15528 00012654 771E                <1>       ja     short sysdrive_err
15529                              <1>
15530                              <1>       ; Save current drive and reset mode
15531                              <1>       ; for 'reset_working_path' procedure (for MainProg)
15532 00012656 30C0                <1>       xor    al, al
15533 00012658 66A3[78960100]      <1>       mov    [SWP_Mode], ax ; ah = 0
15534 0001265E A0[868A0100]        <1>       mov    al, [Current_Drv]
15535 00012663 FEC4                <1>       inc    ah ; mov ah, 1
15536 00012665 66A3[7A960100]      <1>       mov    [SWP_DRV], ax
15537                              <1>
15538 0001266B 88DA                <1>       mov    dl, bl
15539 0001266D E86C5AFFFF          <1>       call   change_current_drive
15540 00012672 730F                <1>       jnc    short sysdrive_ok
15541                              <1> sysdrive_err:
15542 00012674 C705[64030300]0F00- <1>       mov    dword [u.r0], ERR_DRV_NOT_RDY ; 'drive not ready !'
15542 0001267C 0000                <1>
15543 0001267E E959B2FFFF          <1>       jmp    error
15544                              <1> sysdrive_ok:
15545 00012683 A0[868A0100]        <1>       mov    al, [Current_Drv]
15546 00012688 8A25[50400100]      <1>       mov    ah, [Last_DOS_DiskNo]
15547 0001268E A3[64030300]        <1>       mov    [u.r0], eax
15548 00012693 E964B2FFFF          <1>       jmp    sysret
15549                              <1>
15550                              <1>
15551                              <1> sysdir: ; Get Current (Working) Drive & Directory (for user)
15552                              <1>       ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
15553                              <1>       ;
15554                              <1>         ; INPUT ->
15555                              <1>         ;          EBX = Current directory name buffer address
15556                              <1>       ;            (Buffer length = 92 bytes)
15557                              <1>       ; OUTPUT ->
15558                              <1>       ;          AL = Current drive (0=A: .. 2=C:)
15559                              <1>       ;         If CF = 1 -> AL = error code
15560                              <1>       ;
15561                              <1>       ; Modified Registers: EAX (at the return of system call)
15562                              <1>       ;
15563                              <1>       ; Note: Required directory name buffer length may be
15564                              <1>       ;     <= 92 bytes for current TRDOS 386 version.
15565                              <1>       ;     (7*12 name chars + 7 slash + 0)
15566                              <1>
15567 00012698 89E5                <1>       mov    ebp, esp
15568 0001269A 83EC60              <1>       sub    esp, 96
15569 0001269D 53                  <1>       push   ebx ; User's buffer address
15570 0001269E 30D2                <1>       xor    dl, dl ; 0 = current drive
15571 000126A0 E83589FFFF          <1>       call   get_current_directory
15572 000126A5 72CD                <1>       jc     short sysdrive_err ; 'drive not ready !' error
15573 000126A7 89E6                <1>       mov    esi, esp ; System's buffer address
15574 000126A9 5F                  <1>       pop    edi  ; User's buffer address
15575                              <1>       ; ecx = transfer (byte) count (<=92)
15576 000126AA E843F4FFFF          <1>       call   transfer_to_user_buffer
15577 000126AF 89EC                <1>       mov    esp, ebp
15578 000126B1 730F                <1>       jnc    short sysdir_ok
15579                              <1> sysdir_err:
15580 000126B3 C705[64030300]2E00- <1>       mov    dword [u.r0], ERR_BUFFER  ; 'buffer error !'
15580 000126BB 0000                <1>
15581 000126BD E91AB2FFFF          <1>       jmp    error
15582                              <1> sysdir_ok:
```

```
15583 000126C2 8A0D[868A0100]      <1>       mov    cl, [Current_Drv]
15584 000126C8 890D[64030300]      <1>       mov    [u.r0], ecx
15585 000126CE E929B2FFFF          <1>       jmp    sysret
15586                              <1>
15587                              <1>
15588                              <1> sysldrvt: ; Get copy of Logical DOS Drive Description Table
15589                              <1>       ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
15590                              <1>       ;
15591                              <1>         ; INPUT ->
15592                              <1>       ;          BL = Logical DOS drive number (zero based)
15593                              <1>         ;           ECX = Logical DOS drv desc table buffer addr
15594                              <1>       ;            (Buffer length = 256 bytes)
15595                              <1>       ; OUTPUT ->
15596                              <1>       ;          cf = 0 ->
15597                              <1>       ;               AL = Current Drive number
15598                              <1>       ;               AH = The Last Logical DOS Drive no.
15599                              <1>       ;          cf = 1 -> Error code in AL
15600                              <1>       ;               AH = The Last Logical DOS Drive no.
15601                              <1>       ;
15602                              <1>       ; Modified Registers: EAX (at the return of system call)
15603                              <1>       ;
15604                              <1>       ; Note: Required description table buffer length is
15605                              <1>       ;     256 bytes for current TRDOS 386 version.
15606                              <1>
15607 000126D3 89CF               <1>       mov    edi, ecx ; Destination address (user space)
15608 000126D5 88DC               <1>       mov    ah, bl
15609 000126D7 30C0               <1>       xor    al, al
15610 000126D9 BE00010900         <1>       mov    esi, Logical_DOSDisks
15611 000126DE 01C6               <1>       add    esi, eax ; Source address (system space)
15612 000126E0 B900010000         <1>       mov    ecx, 256 ; Byte count
15613                              <1>                       ; Logical Dos Drv Desc Table size
15614 000126E5 E808F4FFFF         <1>       call   transfer_to_user_buffer
15615 000126EA 72C7               <1>       jc     short sysdir_err
15616 000126EC 8A2D[50400100]      <1>       mov    ch, [Last_DOS_DiskNo]
15617 000126F2 EBCE               <1>       jmp    short sysdir_ok
15618                              <1>
15619                              <1>
15620                              <1> systime: ; Get System Date&Time
15621                              <1>       ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
15622                              <1>       ;
15623                              <1>         ; INPUT -> BL =
15624                              <1>       ;          0 = Get Date&Time in Unix/Epoch format
15625                              <1>       ;          1 = Get Time in MSDOS format
15626                              <1>       ;          2 = Get Date in MSDOS format
15627                              <1>       ;          3 = Get Date&Time in MSDOS format
15628                              <1>       ;          4 & other values =
15629                              <1>       ;             System timer ticks will be returned
15630                              <1>       ;             in EAX and Carry Flag will be set.
15631                              <1>       ;             (CF will not be set if BL = 4)
15632                              <1>       ; OUTPUT ->
15633                              <1>       ;      For BL input = 3
15634                              <1>       ;           EAX = Current Time (RTC)
15635                              <1>       ;             AL = Second (DL in MSDOS)
15636                              <1>       ;             AH = Minute (CL in MSDOS)
15637                              <1>       ;             HW of EAX = Hour (CH in MSDOS)
15638                              <1>       ;         EDX = Current System Date (RTC)
15639                              <1>       ;             DL = Day (DL in MSDOS)
15640                              <1>       ;             DH = Month (DH in MSDOS)
15641                              <1>       ;             HW of EDX = Year (CX in MSDOS)
15642                              <1>       ;
15643                              <1>       ;      For BL input = 2
15644                              <1>       ;         EAX = Current System Date (RTC)
15645                              <1>       ;             DL = Day (DL in MSDOS)
15646                              <1>       ;             DH = Month (DH in MSDOS)
15647                              <1>       ;             HW of EDX = Year (CX in MSDOS)
15648                              <1>       ;
15649                              <1>       ;      For BL input = 1
15650                              <1>       ;           EAX = Current Time (RTC)
15651                              <1>       ;             AL = Second (DL in MSDOS)
15652                              <1>       ;             AH = Minute (CL in MSDOS)
15653                              <1>       ;             HW of EAX = Hour (CH in MSDOS)
15654                              <1>       ;
15655                              <1>       ;      For BL input = 0
15656                              <1>       ;           EAX = Unix (Epoch) Time Ticks/Seconds
15657                              <1>       ;
15658                              <1>       ;      For BL input  = 4
15659                              <1>       ;         EAX = System timer ticks
15660                              <1>       ;
15661                              <1>       ;      If CF = 1 (for other values of BL input)
15662                              <1>       ;         EAX = System timer ticks (no error code!)
15663                              <1>       ;
15664                              <1>       ; Modified Registers: EAX, (EDX)
15665                              <1>       ;              (at the return of system call)
15666                              <1>       ;
15667                              <1>
15668 000126F4 20DB               <1>       and    bl, bl
15669 000126F6 750F               <1>       jnz    short systime_1
15670 000126F8 E8D14FFFFF         <1>       call   epoch
15671                              <1> systime_0:
15672 000126FD A3[64030300]        <1>       mov    [u.r0], eax
15673 00012702 E9F5B1FFFF         <1>       jmp    sysret
15674                              <1> systime_1:
15675 00012707 80FB04              <1>       cmp    bl, 4
15676 0001270A 7211               <1>       jb     short systime_2
15677 0001270C A1[408A0100]        <1>       mov    eax, [TIMER_LH] ; 18.2 Hz timer ticks
15678                              <1>                               ; Note: [TIMER_LH] may be set
15679                              <1>                               ; to wrong timer value due to
15680                              <1>                               ; program functions.
15681                              <1>                               ; (This value must not be
15682                              <1>                               ; accepted as [TIMER_LH]/18.2
15683                              <1>                               ; seconds since the midnight.)
15684 00012711 76EA               <1>       jna    short systime_0
15685 00012713 A3[64030300]        <1>       mov    [u.r0], eax
15686 00012718 E9BFB1FFFF         <1>       jmp    error ; cf = 1 & [u.r0] = eax = timer ticks
15687                              <1>
```

```
15688                                  <1> systime_2:
15689                                  <1>       ;push  ebx
15690 0001271D E80E4FFFFF              <1>       call   get_rtc_date_time
15691                                  <1>       ;pop   ebx
15692 00012722 F6C301                  <1>       test   bl, 1
15693 00012725 7429                    <1>       jz     short systime_4
15694 00012727 30E4                    <1>       xor    ah, ah
15695 00012729 A0[8E860100]            <1>       mov    al, [hour]
15696 0001272E 88C2                    <1>       mov    dl, al
15697 00012730 C1E010                  <1>       shl    eax, 16
15698 00012733 A0[92860100]            <1>       mov    al, [second]
15699 00012738 8A25[90860100]          <1>       mov    ah, [minute]
15700 0001273E F6C302                  <1>       test   bl, 2
15701 00012741 74BA                    <1>       jz     short systime_0
15702                                  <1>       ; Check time & date match risk
15703                                  <1>       ; (23:59:59 may cause to wrong
15704                                  <1>       ; date -new day with previous date-...)
15705 00012743 80FA17                  <1>       cmp    dl, 23
15706 00012746 7206                    <1>       jb     short systime_3
15707 00012748 663D3B3B                <1>       cmp    ax, (59*256)+59 ; if hour is 23:59:59
15708 0001274C 73CF                    <1>       jnb    short systime_2 ; wait for 1 second
15709                                  <1> systime_3:
15710                                  <1>       ; eax = time
15711 0001274E 89C6                    <1>       mov    esi, eax
15712                                  <1> systime_4:
15713 00012750 66A1[88860100]          <1>       mov    ax, [year]
15714 00012756 C1E010                  <1>       shl    eax, 16
15715 00012759 A0[8C860100]            <1>       mov    al, [day]
15716 0001275E 8A25[8A860100]          <1>       mov    ah, [month]
15717                                  <1>       ; eax = date
15718 00012764 80E301                  <1>       and    bl, 1
15719 00012767 7494                    <1>       jz     short systime_0
15720 00012769 96                      <1>       xchg   esi, eax
15721                                  <1>       ; eax = time, esi = date
15722 0001276A 8B2D[60030300]          <1>       mov    ebp, [u.usp]  ; EBP points to user's registers
15723                                  <1>       ; (user) edx <-- (system) esi
15724 00012770 897514                  <1>       mov    [ebp+20], esi ; return to user with EDX value
15725 00012773 EB88                    <1>       jmp    short systime_0
15726                                  <1>
15727                                  <1>
15728                                  <1> sysstime: ; Set System Date&Time
15729                                  <1>       ; 31/12/2017
15730                                  <1>       ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
15731                                  <1>       ;
15732                                  <1>         ; INPUT -> BL =
15733                                  <1>       ;          0 = Set Date&Time in Unix/Epoch format
15734                                  <1>       ;          1 = Set Time in MSDOS format
15735                                  <1>       ;          2 = Set Date in MSDOS format
15736                                  <1>       ;          3 = Set Date&Time in MSDOS format
15737                                  <1>       ;          4 = Set System Timer (Ticks)
15738                                  <1>       ;          5 = Convert/Save current time to/as
15739                                  <1>       ;             18.2 Hz system timer ticks
15740                                  <1>       ;          6 = Convert MSDOS Date&Time to UNIX format
15741                                  <1>       ;             without setting system date&time ; (test)
15742                                  <1>       ;          7 = Convert UNIX Date&Time to MSDOS format
15743                                  <1>       ;             without setting system date&time ; (test)
15744                                  <1>       ;         8-0FFh = invalid !
15745                                  <1>       ;        ECX = Time (or Timer) value in selected format
15746                                  <1>       ;        EDX = Date value in MSDOS format if BL=2,3,6
15747                                  <1>       ;
15748                                  <1>       ; OUTPUT ->
15749                                  <1>       ;      If CF = 0 ->
15750                                  <1>       ;           EAX = Set value
15751                                  <1>       ;      If CF = 1 -> (invalid BL input)
15752                                  <1>       ;         EAX = Ticks count [TIMER_LH]
15753                                  <1>       ;
15754                                  <1>
15755 00012775 20DB                    <1>       and    bl, bl ; 0
15756 00012777 7511                    <1>       jnz    short sysstime_0
15757 00012779 89C8                    <1>       mov    eax, ecx
15758 0001277B E8D84FFFFF              <1>       call   convert_from_epoch
15759 00012780 E88450FFFF              <1>       call   set_rtc_date_time
15760 00012785 E972B1FFFF              <1>       jmp    sysret
15761                                  <1> sysstime_0:
15762 0001278A 80FB08                  <1>       cmp    bl, 8
15763 0001278D 722D                    <1>       jb     short sysstime_1
15764                                  <1>       ; invalid input (>7)
15765 0001278F A1[408A0100]            <1>       mov    eax, [TIMER_LH] ; 18.2 Hz timer ticks
15766                                  <1>                              ; Note: [TIMER_LH] may be set
15767                                  <1>                              ; to wrong timer value due to
15768                                  <1>                              ; program functions.
15769                                  <1>                              ; (This value must not be
15770                                  <1>                              ; accepted as [TIMER_LH]/18.2
15771                                  <1>                              ; seconds since the midnight.)
15772 00012794 A3[64030300]            <1>       mov    [u.r0], eax
15773 00012799 E93EB1FFFF              <1>       jmp    error ; cf = 1 & [u.r0] = eax = timer ticks
15774                                  <1>
15775                                  <1> sysstime_8:
15776                                  <1>       ; BL = 7
15777 0001279E 89C8                    <1>       mov    eax, ecx ; seconds since 1/1/1970 00:00:00
15778 000127A0 E8B34FFFFF              <1>       call   convert_from_epoch
15779 000127A5 30E4                    <1>       xor    ah, ah
15780 000127A7 A0[8E860100]            <1>       mov    al, [hour]
15781 000127AC C1E010                  <1>       shl    eax, 16
15782 000127AF A0[92860100]            <1>       mov    al, [second]
15783 000127B4 8A25[90860100]          <1>       mov    ah, [minute]
15784 000127BA EB92                    <1>       jmp    short systime_3
15785                                  <1>
15786                                  <1> sysstime_1:
15787 000127BC 80FB04                  <1>       cmp    bl, 4
15788 000127BF 743F                    <1>       je     short sysstime_2 ; set system timer ticks
15789 000127C1 80FB05                  <1>       cmp    bl, 5
15790 000127C4 754B                    <1>       jne    short sysstime_4
15791                                  <1>       ; convert current time to system timer ticks (18.2Hz)
15792 000127C6 E8654EFFFF              <1>       call   get_rtc_date_time
```

```
15793 000127CB 0FB60D[8E860100]    <1>       movzx  ecx, byte [hour]
15794 000127D2 B8100E0000          <1>       mov    eax, 60*60 ; 1 hour = 3600 seconds
15795 000127D7 F7E1                <1>       mul    ecx
15796 000127D9 89C3                <1>       mov    ebx, eax
15797 000127DB B13C                <1>       mov    cl, 60  ; 1 minute = 60 seconds
15798 000127DD 0FB605[90860100]    <1>       movzx  eax, byte [minute]
15799 000127E4 F7E1                <1>       mul    ecx
15800 000127E6 01D8                <1>       add    eax, ebx
15801 000127E8 8A0D[92860100]      <1>       mov    cl, [second]
15802 000127EE 01C8                <1>       add    eax, ecx
15803 000127F0 B1B6                <1>       mov    cl, 182
15804 000127F2 F7E1                <1>       mul    ecx
15805 000127F4 83C009              <1>       add    eax, 9
15806 000127F7 83D200              <1>       adc    edx, 0
15807 000127FA B10A                <1>       mov    cl, 10
15808 000127FC F7F1                <1>       div    ecx
15809                              <1>       ; eax = ((182*seconds)+9)/10
15810 000127FE 89C1                <1>       mov    ecx, eax
15811                              <1> sysstime_2:
15812 00012800 890D[408A0100]      <1>       mov    [TIMER_LH], ecx ; 18.2 * seconds
15813                              <1> sysstime_3:
15814 00012806 890D[64030300]      <1>       mov    [u.r0], ecx
15815 0001280C E9EBB0FFFF          <1>       jmp    sysret
15816                              <1> sysstime_4:
15817 00012811 80FB06              <1>       cmp    bl, 6
15818 00012814 7788                <1>       ja     short sysstime_8
15819                              <1>
15820 00012816 890D[64030300]      <1>       mov    [u.r0], ecx
15821                              <1>
15822 0001281C 880D[92860100]      <1>       mov    [second], cl
15823 00012822 882D[90860100]      <1>       mov    [minute], ch
15824 00012828 C1E910              <1>       shr    ecx, 16
15825 0001282B 880D[8E860100]      <1>       mov    [hour], cl
15826                              <1>       ; BL = 1,2,3,6
15827 00012831 80FB01              <1>       cmp    bl, 1
15828 00012834 762A                <1>       jna    short sysstime_5
15829                              <1>       ; BL = 2,3,6
15830 00012836 8815[8C860100]      <1>       mov    [day], dl
15831 0001283C 8835[8A860100]      <1>       mov    [month], dh
15832 00012842 C1EA10              <1>       shr    edx, 16
15833 00012845 668915[88860100]    <1>       mov    [year], dx
15834 0001284C 80E303              <1>       and    bl, 3
15835 0001284F 742D                <1>       jz     short sysstime_7 ; 6
15836                              <1>       ; BL = 2,3
15837 00012851 F6C301              <1>       test   bl, 1
15838 00012854 7419                <1>       jz     short sysstime_6 ; 2
15839                              <1>       ; BL = 3
15840 00012856 E8AE4FFFFF          <1>       call   set_rtc_date_time
15841 0001285B E99CB0FFFF          <1>       jmp    sysret
15842                              <1> sysstime_5:
15843                              <1>       ; BL = 1
15844 00012860 E8E54FFFFF          <1>       call   set_time_bcd
15845 00012865 E86F42FFFF          <1>       call   set_rtc_time
15846 0001286A E98DB0FFFF          <1>       jmp    sysret
15847                              <1> sysstime_6:
15848                              <1>       ; BL = 2
15849 0001286F E8A94FFFFF          <1>       call   set_date_bcd
15850 00012874 E8CF42FFFF          <1>       call   set_rtc_date
15851 00012879 E97EB0FFFF          <1>       jmp    sysret
15852                              <1> sysstime_7:
15853                              <1>       ; BL = 6
15854                              <1>       ; [year], [month], [day],
15855                              <1>       ; [hour], [minute], [second]
15856 0001287E E8504EFFFF          <1>       call   convert_to_epoch
15857 00012883 89C1                <1>       mov    ecx, eax ; seconds since 1/1/1970 00:00:00
15858 00012885 E97CFFFFFF          <1>       jmp    sysstime_3
15859                              <1>
15860                              <1> sysrename: ; Rename File (or Directory)
15861                              <1>       ; 19/01/2021
15862                              <1>       ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)
15863                              <1>       ;
15864                              <1>         ; INPUT ->
15865                              <1>         ;          EBX = File/Directory (ASCIIZ) name address
15866                              <1>       ;         ECX = New name (in same dir, no path name)
15867                              <1>       ; OUTPUT ->
15868                              <1>       ;          cf = 0 -> EAX = 0
15869                              <1>       ;          cf = 1 -> Error code in AL
15870                              <1>
15871                              <1>       ; 19/01/2021
15872 0001288A 803D[B0030300]00    <1>       cmp    byte [u.uid], 0  ; root (super user) ?
15873 00012891 7614                <1>       jna    short sysrename_0
15874                              <1>
15875                              <1> sysrename_perm_err:
15876                              <1>       ;mov   dword [u.r0], ERR_PERM_DENIED
15877 00012893 B80B000000          <1>       mov    eax, ERR_PERM_DENIED ; 'permission denied !'
15878 00012898 A3[64030300]        <1>       mov    [u.r0], eax
15879 0001289D A3[C8030300]        <1>       mov    [u.error], eax
15880 000128A2 E935B0FFFF          <1>       jmp    error
15881                              <1>
15882                              <1> sysrename_0:
15883 000128A7 51                  <1>       push   ecx ; new file name address (in user space)
15884 000128A8 89DE                <1>       mov    esi, ebx
15885                              <1>       ; file name is forced, change directory as temporary
15886                              <1>       ;mov   ax, 1
15887                              <1>       ;mov   [FFF_Valid], ah ; 0 ; reset
15888                              <1>       ;call  set_working_path
15889 000128AA E8BF060000          <1>       call   set_working_path_x
15890 000128AF 731E                <1>       jnc    short sysrename_1
15891 000128B1 21C0                <1>       and    eax, eax  ; 0 -> Bad Path!
15892 000128B3 7505                <1>       jnz    short sysrename_err
15893                              <1>       ; eax = 0
15894                              <1> sysrename_path_not_found:
15895 000128B5 B813000000          <1>       mov    eax, ERR_INV_PATH_NAME ; 'Bad path name !'
15896                              <1> sysrename_err:
15897 000128BA 59                  <1>       pop    ecx ; new file name address (in user space)
```

```
15898                                   <1> sysrename_error:
15899 000128BB A3[64030300]               <1>       mov   [u.r0], eax
15900 000128C0 A3[C8030300]               <1>       mov   [u.error], eax
15901 000128C5 E879070000                 <1>       call  reset_working_path
15902 000128CA E90DB0FFFF                 <1>       jmp   error
15903                                     <1> sysrename_1:
15904 000128CF B008                       <1>       mov   al, 08h ; Except volume labels (& long names)
15905 000128D1 A0[3C940100]               <1>       mov   al, [Attributes]
15906 000128D6 2410                       <1>       and   al, 10h ;
15907                                     <1>       ;mov  esi, FindFile_Name
15908                                     <1>       ;mov  ax, 1800h ; Only files
15909                                     <1>       ;mov  ax, 0810h ; Only directories
15910 000128D8 66B80008                   <1>       mov   ax, 0800h ; Find File or Directory
15911 000128DC E8A06BFFFF                 <1>       call  find_first_file
15912                                     <1>       ;jnc  short sysrename_2
15913 000128E1 72D7                       <1>       jc    short sysrename_err
15914                                     <1> sysrename_2:
15915                                     <1>       ; ESI = Directory Entry (FindFile_DirEntry) Location
15916                                     <1>       ; EDI = Directory Buffer Directory Entry Location
15917                                     <1>       ; EAX = File Size
15918                                     <1>       ;  BL = Attributes of The File/Directory
15919                                     <1>       ;  BH = Long Name Yes/No Status (>0 is YES)
15920                                     <1>       ;  DX > 0 : Ambiguous filename chars are used
15921                                     <1>
15922 000128E3 6621D2                     <1>       and   dx, dx ; Ambiguous filename chars used sign (DX>0)
15923 000128E6 7407                       <1>       jz    short sysrename_3
15924 000128E8 B81A000000                 <1>       mov   eax, ERR_INV_FILE_NAME ; 'invalid file name !'
15925 000128ED EBCB                       <1>         jmp  short sysrename_err
15926                                     <1> sysrename_3:
15927                                     <1>       ; EDI = Directory buffer entry offset/address
15928                                     <1>       ; BL = File (or Directory) Attributes
15929                                     <1>       ; mov  bl, [EDI+0Bh]
15930                                     <1>
15931 000128EF 5A                         <1>       pop   edx ; new file name address (in user space)
15932                                     <1>
15933                                     <1>       ; check file/directory attributes
15934 000128F0 F6C307                     <1>       test  bl, 7 ; system, hidden, readonly
15935 000128F3 759E                       <1>         jnz  short sysrename_perm_err
15936                                     <1> sysrename_4:
15937 000128F5 66817F0CA101               <1>       cmp   word [edi+DirEntry_NTRes], 01A1h ; Singlix FS
15938 000128FB 7496                       <1>       je    short sysrename_perm_err ; -temporary!-
15939                                     <1>
15940                                     <1>       ; save old file name & file info (FFF structure)
15941 000128FD BE[26930100]               <1>       mov   esi, FindFile_Drv
15942 00012902 BF[6C940100]               <1>       mov   edi, SourceFile_Drv
15943 00012907 B920000000                 <1>       mov   ecx, 128/4
15944 0001290C F3A5                       <1>       rep   movsd
15945                                     <1>
15946 0001290E 89D6                       <1>       mov   esi, edx ; new file name address (in user space)
15947 00012910 BF[EC940100]               <1>       mov   edi, DestinationFile_Drv
15948 00012915 E8388DFFFF                 <1>       call  parse_path_name
15949 0001291A 729F                       <1>       jc    short sysrename_error ; eax = 1 (Bad file name)
15950                                     <1>
15951                                     <1>       ; same drive ?
15952 0001291C A0[26930100]               <1>       mov   al, [FindFile_Drv]
15953 00012921 3A05[EC940100]             <1>       cmp   al, [DestinationFile_Drv]
15954                                     <1>       ;jne  short sysrename_perm_err ; Permission denied
15955 00012927 7509                       <1>       jne   short sysrename_5 ; Bad file name
15956                                     <1>
15957                                     <1>       ; no path name !? (rename file in same directory)
15958 00012929 803D[ED940100]20           <1>       cmp   byte [DestinationFile_Directory], 20h
15959 00012930 7607                       <1>       jna   short sysrename_6
15960                                     <1> sysrename_5:
15961 00012932 B801000000                 <1>       mov   eax, ERR_BAD_CMD_ARG ; 1 = Bad file name
15962                                     <1>                                  ; (Bad argument)
15963 00012937 EB82                       <1>       jmp   short sysrename_error
15964                                     <1> sysrename_6:
15965 00012939 803D[2E950100]20           <1>       cmp   byte [DestinationFile_Name], 20h
15966 00012940 76F0                       <1>       jna   short sysrename_5
15967                                     <1>
15968 00012942 BE[2E950100]               <1>       mov   esi, DestinationFile_Name
15969 00012947 E8F36EFFFF                 <1>       call  check_filename ; is it a valid msdos file name?
15970 0001294C 0F8269FFFFFF               <1>       jc    sysrename_error ; 26 = ERR_INV_FILE_NAME
15971                                     <1>
15972                                     <1>       ;mov  esi, DestinationFile_Name
15973 00012952 66B80008                   <1>       mov   ax, 0800h ; Find File or Directory
15974 00012956 E8266BFFFF                 <1>       call  find_first_file
15975 0001295B 720A                       <1>       jc    short sysrename_7
15976                                     <1>
15977 0001295D B80E000000                 <1>       mov   eax, ERR_FILE_EXISTS  ; file already exists !
15978 00012962 E954FFFFFF                 <1>       jmp   sysrename_error
15979                                     <1> sysrename_7:
15980                                     <1>       ; eax = 2 (File not found !)
15981 00012967 3C02                       <1>       cmp   al, 2 ; ERR_NOT_FOUND
15982 00012969 0F854CFFFFFF               <1>       jne   sysrename_error
15983                                     <1>
15984                                     <1>       ; 31/12/2017
15985                                     <1>       ; Following code is also part of 'rename_file' in
15986                                     <1>       ; 'trdosk3.s' (MainProg's 'rename' command) ; 13/11/2017
15987 0001296F BE[2E950100]               <1>       mov   esi, DestinationFile_Name ; (Rename_NewName)
15988 00012974 668B0D[E6940100]           <1>       mov   cx, [SourceFile_DirEntryNumber]
15989 0001297B 66A1[D2940100]             <1>       mov   ax, [SourceFile_DirEntry+20] ; First Cluster, HW
15990 00012981 C1E010                     <1>       shl   eax, 16
15991 00012984 66A1[D8940100]             <1>       mov   ax, [SourceFile_DirEntry+26] ; First Cluster, LW
15992 0001298A 0FB61D[BB940100]           <1>       movzx ebx, byte [SourceFile_LongNameEntryLength]
15993 00012991 E80495FFFF                 <1>       call  rename_directory_entry
15994 00012996 0F821FFFFFFF               <1>       jc    sysrename_error
15995                                     <1>       ;xor  eax, eax
15996 0001299C A3[64030300]               <1>       mov   [u.r0], eax ; 0
15997                                     <1>       ;mov  [u.error], eax
15998 000129A1 E89D060000                 <1>       call  reset_working_path
15999 000129A6 E951AFFFFF                 <1>       jmp   sysret
16000                                     <1>
16001                                     <1> sysmem: ; Get Total&Free Memory amount
16002                                     <1>       ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)
```

```
16003                                  <1>       ;
16004                                  <1>         ; INPUT ->
16005                                  <1>       ;     none
16006                                  <1>       ; OUTPUT ->
16007                                  <1>       ;     EAX = Total memory count (in bytes)
16008                                  <1>       ;     EBX = Virtually available memory amount (in bytes)
16009                                  <1>       ;           = 4GB - CORE (4MB)
16010                                  <1>       ;     ECX = Free memory count (in bytes)
16011                                  <1>       ;     EDX = Calculated free memory count (in bytes)
16012                                  <1>
16013 000129AB A1[C4890100]            <1>       mov   eax, [memory_size] ; in pages
16014 000129B0 C1E00C                  <1>       shl   eax, 12                ; in bytes
16015 000129B3 A3[64030300]            <1>       mov   [u.r0], eax
16016 000129B8 E89019FFFF              <1>       call  calc_free_mem
16017                                  <1>       ; edx = calculated free pages
16018                                  <1>       ; ecx = 0
16019 000129BD 8B2D[60030300]          <1>       mov   ebp, [u.usp]  ; EBP points to user's registers
16020 000129C3 C745100000C0FF          <1>       mov   dword [ebp+16], ECORE ; EBX (for user)
16021                                  <1>                           ; 0FFC00000h ; 4GB - 4MB
16022 000129CA C1E20C                  <1>       shl   edx, 12
16023 000129CD 895514                  <1>       mov   [ebp+20], edx ; EDX (for user)
16024 000129D0 8B0D[C8890100]          <1>       mov   ecx, [free_pages]
16025 000129D6 C1E10C                  <1>       shl   ecx, 12      ; free bytes
16026 000129D9 894D18                  <1>       mov   [ebp+24], ecx ; ECX (for user)
16027                                  <1>       ;mov  [free_pages], edx
16028 000129DC E91BAFFFFF              <1>       jmp   sysret
16029                                  <1>
16030                                  <1> sysprompt:
16031                                  <1>       ; Set TRDOS 386 Command Interpreter (MainProg) prompt
16032                                  <1>       ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)
16033                                  <1>       ;
16034                                  <1>         ; INPUT ->
16035                                  <1>       ;     EBX = 0 -> use default prompt
16036                                  <1>       ;     EBX > 0 -> prompt string (ASCIIZ) address
16037                                  <1>       ;                (Max. 11 characters except ZERO tail)
16038                                  <1>       ; OUTPUT ->
16039                                  <1>       ;     (EAX = 0)
16040                                  <1>       ;     CF = 0 -> Successful
16041                                  <1>       ;     CF = 1 -> Failed
16042                                  <1>
16043 000129E1 21DB                   <1>       and   ebx, ebx
16044 000129E3 750A                   <1>       jnz   short sysprompt_0
16045                                  <1>
16046 000129E5 E89B64FFFF             <1>       call  default_command_prompt ; '['+'TRDOS'+']'
16047 000129EA E90DAFFFFF             <1>       jmp   sysret
16048                                  <1>
16049                                  <1> sysprompt_0:
16050 000129EF 31C0                   <1>       xor   eax, eax
16051 000129F1 A3[64030300]           <1>       mov   [u.r0], eax
16052 000129F6 89DE                   <1>       mov   esi, ebx
16053 000129F8 B90C000000             <1>       mov   ecx, 12
16054 000129FD 89E5                   <1>       mov   ebp, esp
16055 000129FF 29CC                   <1>       sub   esp, ecx
16056 00012A01 49                     <1>       dec   ecx ; 11
16057 00012A02 89E7                   <1>       mov   edi, esp
16058 00012A04 E833F1FFFF             <1>       call  transfer_from_user_buffer
16059 00012A09 7211                   <1>       jc    short sysprompt_err
16060 00012A0B 803E20                 <1>       cmp   byte [esi], 20h
16061 00012A0E 760C                   <1>       jna   short sysprompt_err
16062 00012A10 E88264FFFF             <1>       call  set_command_prompt
16063 00012A15 89EC                   <1>       mov   esp, ebp
16064 00012A17 E9E0AEFFFF             <1>       jmp   sysret
16065                                  <1> sysprompt_err:
16066                                  <1> syspath_err:
16067 00012A1C 89EC                   <1>       mov   esp, ebp
16068 00012A1E E9B9AEFFFF             <1>       jmp   error
16069                                  <1>
16070                                  <1> syspath:
16071                                  <1>       ; Get/Set Run Path
16072                                  <1>       ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)
16073                                  <1>       ;
16074                                  <1>         ; INPUT ->
16075                                  <1>       ;     EBX = 0 -> get path (to buffer address in ECX)
16076                                  <1>       ;     EBX > 0 -> set path
16077                                  <1>       ;         EBX = Path string buffer address (ASCIIZ)
16078                                  <1>       ;             (Path description except 'PATH=')
16079                                  <1>       ;     ECX = Buffer address (if EBX = 0)
16080                                  <1>       ;           (ECX will not be used if EBX > 0)
16081                                  <1>       ;     DL = Buffer size (0 = 256 byte)
16082                                  <1>       ;
16083                                  <1>       ; OUTPUT ->
16084                                  <1>       ;     CF = 0 -> Successful (EAX = String length)
16085                                  <1>       ;     CF = 1 -> Failed (EAX = 0)
16086                                  <1>       ;
16087                                  <1>       ; NOTE: 'PATH=' or 'PATH' must be excluded
16088                                  <1>       ;  (It must not be at the beginning of the string.)
16089                                  <1>
16090 00012A23 89E5                   <1>       mov   ebp, esp
16091 00012A25 81EC00010000           <1>       sub   esp, 256
16092 00012A2B 89E7                   <1>       mov   edi, esp
16093                                  <1>
16094 00012A2D 31C0                   <1>       xor   eax, eax
16095 00012A2F A3[64030300]           <1>       mov   [u.r0], eax
16096                                  <1>
16097 00012A34 21DB                   <1>       and   ebx, ebx
16098 00012A36 752E                   <1>       jnz   short syspath_0
16099                                  <1>
16100                                  <1>       ; EBX = 0 -> get run path
16101 00012A38 89CB                   <1>       mov   ebx, ecx ; buffer addr (in user's mem space)
16102 00012A3A BE[1D410100]           <1>       mov   esi, Cmd_Path  ; 'PATH' address
16103 00012A3F 0FB6CA                 <1>       movzx ecx, dl
16104 00012A42 80E901                 <1>       sub   cl, 1 ; 0 -> 255, 1 -> 0
16105 00012A45 6683D101               <1>       adc   cx, 1 ; 255 -> 256, 0 -> 1
16106                                  <1>       ; EDI = Output buffer
16107                                  <1>       ; CX = Buffer length
```

```
16108                                  <1>       ; AL = 0 -> use ASCIIZ word in [ESI]
16109                                  <1>       ; ESI = 'PATH' address (with zero tail)
16110 00012A49 E87D7CFFFF              <1>       call   get_environment_string
16111 00012A4E 72CC                    <1>       jc     short syspath_err
16112 00012A50 89DF                    <1>       mov    edi, ebx ; User's buffer address
16113 00012A52 89E6                    <1>       mov    esi, esp
16114                                  <1>       ; EDI = User's buffer address
16115                                  <1>       ; ECX = transfer (byte) count
16116 00012A54 E899F0FFFF              <1>       call   transfer_to_user_buffer
16117 00012A59 72C1                    <1>       jc     short syspath_err
16118 00012A5B 890D[64030300]          <1>       mov    [u.r0], ecx
16119 00012A61 E996AEFFFF              <1>       jmp    sysret
16120                                  <1>
16121                                  <1> syspath_0:
16122 00012A66 89DE                    <1>       mov    esi, ebx
16123 00012A68 0FB6CA                  <1>       movzx  ecx, dl
16124 00012A6B 80E901                  <1>       sub    cl, 1 ; 0 -> 255, 1 -> 0
16125 00012A6E 6683D101                <1>       adc    cx, 1 ; 255 -> 256, 0 -> 1
16126 00012A72 E8C5F0FFFF              <1>       call   transfer_from_user_buffer
16127 00012A77 72A3                    <1>       jc     short syspath_err
16128                                  <1>       ;(*) 'PATH=' will be added to
16129                                  <1>       ;         the head of the string
16130 00012A79 83EC08                  <1>       sub    esp, 8 ;(*)
16131 00012A7C 89FE                    <1>       mov    esi, edi ;(*)
16132 00012A7E E82C7CFFFF              <1>       call   set_path_x ;(*)
16133 00012A83 7297                    <1>       jc     short syspath_err
16134 00012A85 8915[64030300]          <1>       mov    [u.r0], edx ; run path string length
16135 00012A8B E96CAEFFFF              <1>       jmp    sysret
16136                                  <1>
16137                                  <1> sysenv:
16138                                  <1>       ; Get/Set Environment Variables
16139                                  <1>       ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)
16140                                  <1>       ;
16141                                  <1>         ; INPUT ->
16142                                  <1>       ;     EBX = 0 -> get (all) environment variables
16143                                  <1>       ;           (Required Buffer length = 512 bytes)
16144                                  <1>       ;     EBX > 0 -> set (one) environment variable
16145                                  <1>       ;           (If there is not a '=' after
16146                                  <1>       ;           the environment variable name, it will
16147                                  <1>       ;           accepted as 'get environment variable'.)
16148                                  <1>       ;            EBX = Buffer address
16149                                  <1>       ;     ECX = Buffer address (if EBX = 0)
16150                                  <1>       ;           (ECX will not be used if EBX > 0)
16151                                  <1>       ;           (Note: Buffer size is 512 bytes.)
16152                                  <1>       ;     DL = Buffer size (0 = 256 byte)
16153                                  <1>       ;          (For one envrionment variable)
16154                                  <1>       ;
16155                                  <1>       ; OUTPUT ->
16156                                  <1>       ;     (EAX = 0)
16157                                  <1>       ;     CF = 0 -> Successful (EAX = String length)
16158                                  <1>       ;     CF = 1 -> Failed (EAX = 0)
16159                                  <1>       ;
16160                                  <1>       ; Note: Environment variable name, for example,
16161                                  <1>       ;     'PATH=' must be included at the beginning
16162                                  <1>       ;     of the environment string. If the variable
16163                                  <1>       ;     name is as 'PATH' but it is not as 'PATH='
16164                                  <1>       ;     the variable string (row) will be returned.
16165                                  <1>       ;     If variable name is as 'PATH=' but there is
16166                                  <1>       ;     not a following text after the variable name,
16167                                  <1>       ;     the environment variable will be reset/deleted.
16168                                  <1>
16169 00012A90 89E5                    <1>       mov    ebp, esp
16170 00012A92 81EC00020000            <1>       sub    esp, 512
16171 00012A98 89E7                    <1>       mov    edi, esp
16172                                  <1>
16173 00012A9A 31C0                    <1>       xor    eax, eax
16174 00012A9C A3[64030300]            <1>       mov    [u.r0], eax
16175                                  <1>
16176 00012AA1 21DB                    <1>       and    ebx, ebx
16177 00012AA3 7524                    <1>       jnz    short sysenv_0
16178                                  <1>
16179                                  <1>       ; EBX = 0 -> get (all) environment variables
16180 00012AA5 89EC                    <1>       mov    esp, ebp
16181 00012AA7 BE00300900              <1>       mov    esi, Env_Page  ; Environment page
16182 00012AAC 89CF                    <1>       mov    edi, ecx ; buffer addr (in user's mem space)
16183 00012AAE B900020000              <1>       mov    ecx, 512
16184 00012AB3 E83AF0FFFF              <1>       call   transfer_to_user_buffer
16185 00012AB8 0F821EAEFFFF            <1>       jc     error
16186 00012ABE 890D[64030300]          <1>       mov    [u.r0], ecx
16187 00012AC4 E933AEFFFF              <1>       jmp    sysret
16188                                  <1>
16189                                  <1> sysenv_0:
16190 00012AC9 89DE                    <1>       mov    esi, ebx ; * ; user's buffer address
16191 00012ACB 0FB6CA                  <1>       movzx  ecx, dl
16192 00012ACE 80E901                  <1>       sub    cl, 1 ; 0 -> 255, 1 -> 0
16193 00012AD1 6683D101                <1>       adc    cx, 1 ; 255 -> 256, 0 -> 1
16194 00012AD5 E862F0FFFF              <1>       call   transfer_from_user_buffer
16195 00012ADA 723F                    <1>       jc     short sysenv_err
16196 00012ADC 89FE                    <1>       mov    esi, edi
16197 00012ADE 8A06                    <1>       mov    al, [esi]
16198 00012AE0 3C20                    <1>       cmp    al, 20h
16199 00012AE2 7637                    <1>       jna    short sysenv_err
16200 00012AE4 3C3D                    <1>       cmp    al, '='
16201 00012AE6 7433                    <1>       je     short sysenv_err
16202 00012AE8 56                      <1>       push   esi
16203                                  <1> sysenv_1:
16204 00012AE9 46                      <1>       inc    esi
16205 00012AEA 803E3D                  <1>       cmp    byte [esi], '='
16206 00012AED 7433                    <1>       je     short sysenv_3
16207 00012AEF 803E20                  <1>       cmp    byte [esi], 20h
16208 00012AF2 73F5                    <1>       jnb    short sysenv_1
16209 00012AF4 C60600                  <1>       mov    byte [esi], 0
16210 00012AF7 5E                      <1>       pop    esi
16211                                  <1>       ; EDI = Output buffer
16212                                  <1>       ; CX = Buffer length
```

```
16213 00012AF8 30C0               <1>       xor    al, al
16214                             <1>       ; AL = 0 -> use ASCIIZ word in [ESI]
16215                             <1>       ; ESI = Environment variable name address
16216 00012AFA E8CC7BFFFF         <1>       call   get_environment_string
16217 00012AFF 721A               <1>       jc     short sysenv_err
16218 00012B01 89DF               <1>       mov    edi, ebx ; * ; user's buffer address
16219 00012B03 89C1               <1>       mov    ecx, eax ; String length
16220 00012B05 89E6               <1>       mov    esi, esp
16221                             <1>       ; ESI = system buffer address
16222                             <1>       ; EDI = User's buffer address
16223                             <1>       ; ECX = transfer (byte) count
16224 00012B07 E8E6EFFFFF         <1>       call   transfer_to_user_buffer
16225 00012B0C 720D               <1>       jc     short sysenv_err
16226 00012B0E 890D[64030300]     <1>       mov    [u.r0], ecx ; transfer (byte) count
16227                             <1> sysenv_2:
16228 00012B14 89EC               <1>       mov    esp, ebp
16229 00012B16 E9E1ADFFFF         <1>       jmp    sysret
16230                             <1> sysenv_err:
16231 00012B1B 89EC               <1>       mov    esp, ebp
16232 00012B1D E9BAADFFFF         <1>       jmp    error
16233                             <1> sysenv_3:
16234 00012B22 46                 <1>       inc    esi
16235 00012B23 803E20             <1>       cmp    byte [esi], 20h
16236 00012B26 73FA               <1>       jnb    short sysenv_3
16237 00012B28 C60600             <1>       mov    byte [esi], 0
16238 00012B2B 5E                 <1>       pop    esi
16239 00012B2C E85D7CFFFF         <1>       call   set_environment_string
16240 00012B31 72E8               <1>       jc     short sysenv_err
16241 00012B33 8915[64030300]     <1>       mov    [u.r0], edx
16242 00012B39 EBD9               <1>       jmp    short sysenv_2
16243                             <1>
16244                             <1>
16245                             <1> ; 22/01/2021
16246                             <1> ; temporary - 24/01/2016
16247                             <1>
16248                             <1> iget:
16249                             <1>       ;retn
16250                             <1> isintr:
16251                             <1>       ;retn
16252                             <1> iopen:
16253                             <1>       ;retn
16254                             <1> iclose:
16255                             <1>       ;retn
16256                             <1> sndc:
16257                             <1>       ;retn
16258                             <1> access:
16259                             <1>       ;retn
16260                             <1> sleep:
16261 00012B3B C3                 <1>       retn
 3085                                 %include 'trdosk7.s' ; 24/01/2016
    1                             <1> ; ****************************************************************************
    2                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - DISK READ&WRITE : trdosk7.s
    3                             <1> ; ----------------------------------------------------------------------------
    4                             <1> ; Last Update: 25/02/2016
    5                             <1> ; ----------------------------------------------------------------------------
    6                             <1> ; Beginning: 24/01/2016
    7                             <1> ; ----------------------------------------------------------------------------
    8                             <1> ; Assembler: NASM version 2.11 (trdos386.s)
    9                             <1> ; ----------------------------------------------------------------------------
   10                             <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
   11                             <1> ; DISK_IO.ASM (20/07/2011)
   12                             <1> ; ****************************************************************************
   13                             <1> ; DISK_IO.ASM (c) 2009-2011 Erdogan TAN [ 04/07/2009 ] Last Update: 20/07/2011
   14                             <1>
   15                             <1> disk_write:
   16                             <1>       ; 25/02/2016
   17                             <1>       ; 24/02/2016
   18                             <1>       ; 23/02/2016
   19 00012B3C 807E0500           <1>       cmp    byte [esi+LD_LBAYes], 0
   20 00012B40 777B               <1>         ja      short lba_write
   21                             <1>
   22                             <1> chs_write:
   23                             <1>       ; 25/02/2016
   24                             <1>       ; 23/02/2016
   25 00012B42 C605[75920100]03   <1>       mov    byte [disk_rw_op], 3 ; CHS write
   26 00012B49 EB0D               <1>       jmp    short chs_rw
   27                             <1>
   28                             <1> disk_read:
   29                             <1>       ; 25/02/2016
   30                             <1>       ; 24/02/2016
   31                             <1>       ; 23/02/2016
   32                             <1>       ; 17/02/2016
   33                             <1>       ; 14/02/2016
   34                             <1>       ; 31/01/2016 (TRDOS 386 =  TRDOS v2.0)
   35                             <1>       ; 17/10/2010
   36                             <1>       ; 18/04/2010
   37                             <1>       ;
   38                             <1>       ; INPUT -> EAX = Logical Block Address
   39                             <1>       ;         ESI = Logical Dos Disk Table Offset (DRV)
   40                             <1>       ;         ECX = Sector Count
   41                             <1>       ;         EBX = Destination Buffer
   42                             <1>       ; OUTPUT ->
   43                             <1>       ;         cf = 0 or cf = 1
   44                             <1>       ; (Modified registers: EAX, EBX, ECX, EDX)
   45                             <1>
   46 00012B4B 807E0500           <1>       cmp    byte [esi+LD_LBAYes], 0
   47 00012B4F 7775               <1>         ja      short lba_read
   48                             <1>
   49                             <1> chs_read:
   50                             <1>       ; 25/02/2016
   51                             <1>       ; 24/02/2016
   52                             <1>       ; 23/02/2016
   53                             <1>       ; 31/01/2016 (TRDOS 386 =  TRDOS v2.0)
   54                             <1>       ; 20/07/2011
   55                             <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
00012B51 C605[75920100]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:
00012B58 C605[76920100]04   <1>       mov   byte [retry_count], 4
                            <1>
                            <1> chs_read_retry:
                            <1>       ;mov   [sector_count], ecx ; 23/02/2016
                            <1>
00012B5F 50                 <1>       push  eax                ; Linear sector #
00012B60 51                 <1>       push  ecx                ; # of FAT/FILE/DIR sectors
                            <1>
00012B61 0FB74E1E           <1>       movzx ecx, word [esi+LD_BPB+SecPerTrack]
                            <1>       ;movzx ecx, byte [disk_rw_spt] ; 23/02/2016
00012B65 29D2               <1>       sub   edx, edx
00012B67 F7F1               <1>       div   ecx
                            <1>       ; eax = track, dx (dl ) = sector (on track)
                            <1>       ;sub   cl, dl ; 24/02/2016 (spt - sec)
                            <1>       ;push  ecx ; *
00012B69 6689D1             <1>       mov   cx, dx             ; Sector (zero based)
00012B6C 6641               <1>       inc   cx                 ; To make it 1 based
00012B6E 6651               <1>       push  cx
00012B70 668B4E20           <1>       mov   cx, [esi+LD_BPB+Heads]
00012B74 6629D2             <1>       sub   dx, dx
00012B77 F7F1               <1>       div   ecx                ; Convert track to head & cyl
                            <1>       ; eax (ax) = cylinder, dx (dl) = head (max. FFh)
00012B79 88D6               <1>       mov   dh, dl
00012B7B 6659               <1>       pop   cx                 ; AX=Cyl, DH=Head, CX=Sector
00012B7D 8A5602             <1>       mov   dl, [esi+LD_PhyDrvNo]
                            <1>
00012B80 88C5               <1>       mov   ch, al             ; NOTE: max. 1023 cylinders !
00012B82 C0CC02             <1>       ror   ah, 2              ; Rotate 2 bits right
00012B85 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 !
00012B87 B001               <1>       mov   al, 1 ; 25/02/2016
00012B89 8A25[75920100]     <1>       mov   ah, [disk_rw_op]  ; 02h = chs read, 03h = chs write
                            <1>       ;
00012B8F E89526FFFF         <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)
00012B94 8825[77920100]     <1>       mov   [disk_rw_err], ah
                            <1>
00012B9A 59                 <1>       pop   ecx
00012B9B 58                 <1>       pop   eax
00012B9C 7314               <1>       jnc   short chs_read_ok
                            <1>
00012B9E 803D[77920100]09   <1>       cmp   byte [disk_rw_err], 09h ; DMA crossed 64K segment boundary
00012BA5 7408               <1>       je    short chs_read_error_retn
                            <1>
00012BA7 FE0D[76920100]     <1>       dec   byte [retry_count]
00012BAD 75B0               <1>       jnz   short chs_read_retry
                            <1>
                            <1> chs_read_error_retn:
00012BAF F9                 <1>       stc
                            <1>       ;retn
00012BB0 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
00012BB2 40                 <1>       inc   eax ; next sector
00012BB3 81C300020000       <1>       add   ebx, 512
00012BB9 E29D               <1>       loop  chs_read_next_sector
00012BBB EB5E               <1>       jmp   short update_drv_error_byte
                            <1>
                            <1> lba_write:
                            <1>       ; 23/02/2016
00012BBD C605[75920100]1C   <1>       mov   byte [disk_rw_op], 1Ch ; LBA write
```

```
00012BC4 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
00012BC6 C605[75920100]1B    <1>       mov    byte [disk_rw_op], 1Bh ; LBA read
                             <1>
                             <1> lba_rw:
                             <1>       ; 17/02/2016
00012BCD 57                  <1>       push   edi
                             <1>
00012BCE 890D[78920100]      <1>       mov    [sector_count], ecx ; total sector (read) count
                             <1>
00012BD4 8A5602              <1>       mov    dl, [esi+LD_PhyDrvNo]
                             <1>       ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)
                             <1>
                             <1> lba_read_next:
00012BD7 81F900010000        <1>       cmp    ecx, 256
00012BDD 7605                <1>       jna    short lba_read_rsc
00012BDF B900010000          <1>       mov    ecx, 256 ; 17/02/2016
                             <1> lba_read_rsc:
00012BE4 290D[78920100]      <1>       sub    [sector_count], ecx ; remain sectors
                             <1>
00012BEA 89CF                <1>       mov    edi, ecx
00012BEC 89C1                <1>       mov    ecx, eax ; sector number/address
                             <1>
00012BEE C605[76920100]04    <1>       mov    byte [retry_count], 4
                             <1> lba_read_retry:
00012BF5 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
00012BF7 8A25[75920100]      <1>       mov    ah, [disk_rw_op] ; 1Bh = LBA read, 1Ch = LBA write
00012BFD E82726FFFF          <1>       call   int13h
                             <1>       ; al = ? (changed)
                             <1>       ; ah = error code
00012C02 8825[77920100]      <1>       mov    [disk_rw_err], ah
00012C08 7334                <1>       jnc    short lba_read_ok
00012C0A 80FC80              <1>       cmp    ah, 80h ; time out?
00012C0D 740A                <1>         je      short lba_read_stc_retn
00012C0F FE0D[76920100]      <1>       dec    byte [retry_count]
00012C15 7FDE                <1>       jg     short lba_read_retry
00012C17 743A                <1>       jz     short lba_read_reset
                             <1>       ; sf =  1
                             <1>
                             <1> lba_read_stc_retn:
00012C19 F9                  <1>       stc
                             <1> lba_read_retn:
00012C1A 5F                  <1>       pop    edi
                             <1>
                             <1> update_drv_error_byte:
00012C1B 9C                  <1>       pushf
00012C1C 53                  <1>       push   ebx
00012C1D 6651                <1>       push   cx
                             <1>       ;or    ecx, ecx
                             <1>       ;jz    short udrv_errb0
00012C1F 8A0D[77920100]      <1>       mov    cl, [disk_rw_err]
                             <1> udrv_errb0:
00012C25 0FB65E02            <1>       movzx  ebx, byte [esi+LD_PhyDrvNo]
00012C29 80FB02              <1>       cmp    bl, 2
00012C2C 7203                <1>         jb      short udrv_errb1
00012C2E 80EB7E              <1>       sub    bl, 7Eh
                             <1>       ;cmp   bl, 5
                             <1>       ;ja    short udrv_errb2
                             <1> udrv_errb1:
00012C31 81C3[516E0000]      <1>         add     ebx, drv.error ; 13/02/2016
00012C37 880B                <1>       mov    [ebx], cl ; error code
                             <1> udrv_errb2:
00012C39 6659                <1>       pop    cx
00012C3B 5B                  <1>       pop    ebx
00012C3C 9D                  <1>       popf
00012C3D C3                  <1>       retn
                             <1>
                             <1> lba_read_ok:
00012C3E 89C8                <1>       mov    eax, ecx ; sector number
00012C40 01F8                <1>       add    eax, edi ; sector number (next)
00012C42 C1E709              <1>       shl    edi, 9 ; sector count * 512
00012C45 01FB                <1>       add    ebx, edi ; next buffer offset
                             <1>
00012C47 8B0D[78920100]      <1>       mov    ecx, [sector_count] ; remaining sectors
00012C4D 09C9                <1>       or     ecx, ecx
00012C4F 7586                <1>       jnz    short lba_read_next
```

```
 266 00012C51 EBC7                <1>       jmp    short lba_read_retn
 267                              <1>
 268                              <1> lba_read_reset:
 269 00012C53 B40D                <1>       mov    ah, 0Dh ; Alternate reset
 270 00012C55 E8CF25FFFF          <1>         call int13h
 271                              <1>       ; al = ? (changed)
 272                              <1>       ; ah = error code
 273 00012C5A 7399                <1>       jnc    short lba_read_retry
 274 00012C5C EBBC                <1>       jmp    short lba_read_retn
3086                                  %include 'trdosk8.s' ; 24/01/2016
   1                              <1> ; ****************************************************************************
   2                              <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.2) - MAIN PROGRAM : trdosk8.s
   3                              <1> ; ----------------------------------------------------------------------------
   4                              <1> ; Last Update: 12/02/2021
   5                              <1> ; ----------------------------------------------------------------------------
   6                              <1> ; Beginning: 24/01/2016
   7                              <1> ; ----------------------------------------------------------------------------
   8                              <1> ; Assembler: NASM version 2.11 (trdos386.s)
   9                              <1> ; ----------------------------------------------------------------------------
  10                              <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
  11                              <1> ; u0.s (20/11/2015), u4.s (14/10/2015)
  12                              <1> ; ****************************************************************************
  13                              <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
  14                              <1> ; TRDOS2.ASM (09/11/2011)
  15                              <1> ; ----------------------------------------------------------------------------
  16                              <1> ; DIR.ASM (c) 2004-2011 Erdogan TAN  [07/01/2004] Last Update: 09/10/2011
  17                              <1>
  18                              <1> set_run_sequence:
  19                              <1>       ; 23/12/2016
  20                              <1>       ; 10/06/2016
  21                              <1>       ; 22/05/2016
  22                              <1>       ; 20/05/2016
  23                              <1>       ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
  24                              <1>       ; TRDOS 386 feature only !
  25                              <1>       ;
  26                              <1>       ; INPUT ->
  27                              <1>       ;     AL = process number (next process)
  28                              <1>       ;
  29                              <1>       ;     this process must be added to run sequence
  30                              <1>       ;
  31                              <1>       ;     [u.pri] = priority of present process
  32                              <1>       ;
  33                              <1>       ;     DL = priority (queue)
  34                              <1>       ;          0 = background (low) ; run on background
  35                              <1>       ;          1 = regular (normal) ; run as regular
  36                              <1>       ;          2 = event (high)     ; run for event
  37                              <1>       ;
  38                              <1>       ;     1) If the requested process is already running:
  39                              <1>       ;        a) If present priority is high ([u.pri]=2)
  40                              <1>       ;           and requested priority is also high,
  41                              <1>       ;           there is nothing to do! Because it has been
  42                              <1>       ;           done already (before this attempt).
  43                              <1>       ;        b) If present priority is high ([u.pri]=2)
  44                              <1>       ;           and requested priority is not high, there is
  45                              <1>       ;           nothing to do! Because, it's current
  46                              <1>       ;           run queue is unspecified, here. (It may be in
  47                              <1>       ;           a waiting list or in a run queue; if the new
  48                              <1>       ;           priority would be used to add it to relavant
  49                              <1>         ;            run queue, this would be wrong, unnecessary
  50                              <1>       ;           and destabilizing duplication!)
  51                              <1>       ;        c) If present priority is not high ([u.pri]<2)
  52                              <1>         ;            and requested priority is high (event),
  53                              <1>       ;           process will be added to present priority's
  54                              <1>       ;           run queue and then, priority will be changed
  55                              <1>       ;           to high ([u.pri]=2).
  56                              <1>       ;        d) If present priority is not high ([u.pri]<2)
  57                              <1>       ;           and requested priority is not high, [u.pri]
  58                              <1>       ;           value will be changed. There is nothing to do
  59                              <1>       ;           in addition. (The new priority value will be
  60                              <1>       ;           used by 'tswap/tswitch' procedure at 'sysret'
  61                              <1>       ;           or 'sysrele' stage.)
  62                              <1>       ;
  63                              <1>       ;     2) If the requested process is not running:
  64                              <1>       ;        a) If requested priority of the requested
  65                              <1>       ;           (next) process is high (event) and priority
  66                              <1>       ;           of present process is not high, the requested
  67                              <1>       ;           process will be added to ('runq_event') high
  68                              <1>       ;           priority run queue and then present (running)
  69                              <1>       ;           process will be stopped (swapped/switched out)
  70                              <1>       ;           immediately if it is in user mode, or it's
  71                              <1>       ;           [u.quant] value will be reset to 0 and (then)
  72                              <1>       ;           it will be stopped at 'sysret' stage.
  73                              <1>       ;        b) If requested priority of the requested
  74                              <1>       ;           (next) process is high (event) and priority
  75                              <1>       ;           of present process is also high, the requested
  76                              <1>       ;           process will be added to ('runq_event') high
  77                              <1>       ;           priority run queue and present (running)
  78                              <1>       ;           process will be allowed to run until it's
  79                              <1>       ;           time quantum will be elapsed ([u.quant]=0).
  80                              <1>       ;        c) If requested priority of the requested
  81                              <1>       ;           (next) process is not high ('run for event'),
  82                              <1>       ;           there is nothing to do. Because, it's current
  83                              <1>       ;           run queue is unspecified, here. (It may be in
  84                              <1>       ;           a waiting list or in a run queue; if the new
  85                              <1>       ;           priority would be used to add it to relavant
  86                              <1>         ;            run queue, this would be wrong, unnecessary
  87                              <1>       ;           and destabilizing duplication!)
  88                              <1>       ;
  89                              <1>       ; OUTPUT ->
  90                              <1>       ;     none
  91                              <1>       ;
  92                              <1>       ;     [u.pri] = priority of present process
  93                              <1>       ;
  94                              <1>       ;     cf = 1, if the request could not be fulfilled.
  95                              <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:
00012C5E 3A05[B3030300] <1>         cmp     al, [u.uno]     ; same process ?
00012C64 750C           <1>       jne     short srunseq_2 ; no
                        <1>
00012C66 8A25[A9030300] <1>       mov     ah, [u.pri]  ; present/current priority
00012C6C 80FC02         <1>       cmp     ah, 2        ; 'run for event' priority level
00012C6F 7221           <1>       jb      short srunseq_6 ; no
                        <1>
                        <1> srunseq_1:
                        <1>       ; there is nothing to do!
00012C71 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
00012C72 80FA02         <1>       cmp     dl, 2        ; event queue
00012C75 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
00012C77 3A15[A9030300] <1>       cmp     dl, [u.pri]  ; priority of present process
00012C7D 7606           <1>       jna     short srunseq_3 ; is high, also
                        <1>       ;
                        <1>       ; present process will be swapped/switched out
00012C7F FE05[51960100] <1>       inc     byte [p_change] ; 1
                        <1>
                        <1> srunseq_3:
                        <1>       ; add process to 'runq_event' queue for new event
00012C85 BB[52030300]   <1>       mov     ebx, runq_event ; high priority run queue
                        <1>
                        <1> srunseq_4:
                        <1>       ; al = process number
                        <1>       ; ebx = run queue
00012C8A E887EDFFFF     <1>       call    putlu
00012C8F C3             <1>       retn
                        <1>
                        <1> srunseq_5:
00012C90 F5             <1>       cmc
00012C91 C3             <1>       retn
                        <1>
                        <1> srunseq_6:
                        <1>       ; present priority of the process is not high
                        <1>
00012C92 8815[A9030300] <1>       mov     [u.pri], dl ; new priority
                        <1>                           ; (will be used by 'tswap')
                        <1>
00012C98 80FA02         <1>       cmp     dl, 2        ; high priority ?
00012C9B 72F3           <1>       jb      short srunseq_5 ; no, there is nothing to do
                        <1>                               ; in addition
                        <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>
00012C9D BB[54030300]   <1>       mov     ebx, runq_normal ; 'run as regular' queue
                        <1>
00012CA2 20E4           <1>       and     ah, ah  ;  previous value of [u.pri]
00012CA4 75E4           <1>       jnz     short srunseq_4
                        <1>
00012CA6 43             <1>       inc     ebx
00012CA7 43             <1>       inc     ebx
                        <1>       ; ebx = runq_background ; 'run on backgroud' queue
                        <1>
00012CA8 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>
00012CAA 803D[A8030300]00 <1>     cmp     byte [u.quant], 0
00012CB1 772C           <1>       ja      short clk_1
                        <1>       ;
00012CB3 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.
00012CBA 7623           <1>       jna     short clk_1 ; yes, do not swap out
                        <1>       ;
00012CBC 803D[5B030300]FF <1>     cmp      byte [sysflg], 0FFh ; user or system space ?
00012CC3 7520           <1>       jne     short clk_2      ; system space (sysflg <> 0FFh)
                        <1>       ;
```

```
00012CC5 66833D[AA030300]00  <1>       cmp    word [u.intr], 0
00012CCD 7616                <1>       jna    short clk_2
                             <1>       ;
                             <1>       ; 23/05/2016
00012CCF 803D[52960100]00    <1>       cmp    byte [multi_tasking], 0
00012CD6 760D                <1>       jna    short clk_2
                             <1>       ;
00012CD8 FE05[51960100]      <1>       inc    byte [p_change] ; it is time to change running process
00012CDE C3                  <1>       retn
                             <1> clk_1:
00012CDF FE0D[A8030300]      <1>       dec    byte [u.quant]
                             <1> clk_2:
00012CE5 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
00012CE6 80FC07              <1>       cmp    ah, 7
00012CE9 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
00012CEB FB                  <1>       sti
                             <1>
                             <1>       ;sti   ; enable interrupts
00012CEC 80642408FE          <1>       and    byte [esp+8], 11111110b   ; clear carry bit of eflags register
                             <1>
00012CF1 80FC01              <1>       cmp    ah, 1
00012CF4 7205                <1>       jb     short int34h_0
00012CF6 7705                <1>       ja     short int34h_1
                             <1>
00012CF8 EE                  <1>       out    dx, al
                             <1>       ;iretd
00012CF9 EB01                <1>       jmp short int34h_iret
                             <1>
                             <1> int34h_0:
00012CFB 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
00012CFC CF                  <1>       iretd
                             <1>
                             <1> int34h_1:
00012CFD F6C401              <1>       test   ah, 1
00012D00 7516                <1>       jnz    short int34h_3 ; out
                             <1>
                             <1>       ; in
00012D02 80FC02              <1>       cmp    ah, 2
00012D05 7707                <1>       ja     short int34h_2
                             <1>
00012D07 6689D8              <1>       mov    ax, bx
00012D0A 66ED                <1>       in     ax, dx
                             <1>       ;iretd
00012D0C EBEE                <1>       jmp    short int34h_iret
                             <1>
                             <1> int34h_2:
00012D0E 80FC04              <1>       cmp    ah, 4
00012D11 772C                <1>       ja     short int34h_7      ; 12/10/2017
                             <1>       ; ah = 4
00012D13 89D8                <1>       mov    eax, ebx
```

```
00012D15 ED                <1>       in     eax, dx
                           <1>       ;iretd
00012D16 EBE4              <1>       jmp    short int34h_iret
                           <1>
                           <1> int34h_3:
00012D18 80FC03            <1>       cmp    ah, 3
00012D1B 7707              <1>       ja     short int34h_4
                           <1>
00012D1D 6689D8            <1>       mov    ax, bx
00012D20 66EF              <1>       out    dx, ax
                           <1>       ;iretd
00012D22 EBD8              <1>       jmp    short int34h_iret
                           <1>
                           <1> int34h_4:
00012D24 80FC05            <1>       cmp    ah, 5
00012D27 770B              <1>       ja     short int34h_6      ; 12/10/2017
                           <1>       ; ah = 5
00012D29 89D8              <1>       mov    eax, ebx
00012D2B EF                <1>       out    dx, eax
                           <1>       ;iretd
00012D2C EBCE              <1>       jmp    short int34h_iret
                           <1>
                           <1> int34h_5:
00012D2E 804C240801        <1>       or     byte [esp+8], 1     ; set carry bit of eflags register
00012D33 CF                <1>       iretd
                           <1>
                           <1>       ; 12/10/2017
                           <1> int34h_6:
00012D34 6689D8            <1>       mov    ax, bx
00012D37 EE                <1>       out    dx, al
00012D38 EB00              <1>       jmp    short $+2
00012D3A 86E0              <1>       xchg   ah, al
00012D3C EE                <1>       out    dx, al
                           <1>       ;xchg  al, ah
                           <1>       ;iretd
00012D3D EB06              <1>       jmp    short int34h_8
                           <1> int34h_7:
00012D3F EC                <1>       in     al, dx
00012D40 EB00              <1>       jmp    short $+2
00012D42 88C4              <1>       mov    ah, al
00012D44 EC                <1>       in     al, dx
                           <1> int34h_8:
00012D45 86C4              <1>       xchg   al, ah
00012D47 CF                <1>       iretd
                           <1>
                           <1>
                           <1> INT4Ah:
                           <1>       ; 24/01/2016
                           <1>       ; this procedure will be called by 'RTC_INT' (in 'timer.s')
00012D48 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>       ;
00012D49 890424            <1>       mov    [esp], eax ; overwrite call return address
                           <1>       ;push  eax
00012D4C 66B80900          <1>       mov    ax, 9
00012D50 EB07              <1>       jmp    short comm_int
                           <1> com1_int:
                           <1>       ; 07/11/2015
                           <1>       ; 24/10/2015
00012D52 890424            <1>       mov    [esp], eax ; overwrite call return address
                           <1>       ; 23/10/2015
                           <1>       ;push  eax
00012D55 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
00012D59 53                <1>       push   ebx
00012D5A 56                <1>       push   esi
00012D5B 57                <1>       push   edi
00012D5C 1E                <1>       push   ds
00012D5D 06                <1>       push   es
                           <1>       ; 18/11/2015
00012D5E 0F20DB            <1>       mov    ebx, cr3
00012D61 53                <1>       push   ebx ; ****
                           <1>       ;
00012D62 51                <1>       push   ecx ; ***
00012D63 52                <1>       push   edx ; **
                           <1>       ;
00012D64 BB10000000        <1>       mov    ebx, KDATA
00012D69 8EDB              <1>       mov    ds, bx
00012D6B 8EC3              <1>       mov    es, bx
                           <1>       ;
00012D6D 8B0D[C0890100]    <1>       mov    ecx, [k_page_dir]
00012D73 0F22D9            <1>       mov    cr3, ecx
                           <1>       ; 20/11/2015
```

```
                            <1>       ; Interrupt identification register
00012D76 66BAFA02           <1>       mov    dx, 2FAh ; COM2
                            <1>       ;
00012D7A 3C08               <1>       cmp    al, 8
00012D7C 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>       ;
00012D7E FEC6               <1>       inc    dh ; 3FAh ; COM1 Interrupt id. register
                            <1> com_i0:
                            <1>       ;push  eax ; *
                            <1>       ; 07/11/2015
00012D80 A2[2A8A0100]       <1>       mov    byte [ccomport], al
                            <1>       ; 09/11/2015
00012D85 0FB7D8             <1>       movzx  ebx, ax ; 8 or 9
                            <1>       ; 17/11/2015
                            <1>       ; reset request for response status
00012D88 88A3[208A0100]     <1>       mov    [ebx+req_resp-8], ah ; 0
                            <1>       ;
                            <1>       ; 20/11/2015
00012D8E EC                 <1>       in     al, dx       ; read interrupt id. register
00012D8F EB00               <1>       JMP    $+2          ; I/O DELAY
00012D91 2404               <1>       and    al, 4        ; received data available?
00012D93 7470               <1>       jz     short com_eoi; (transmit. holding reg. empty)
                            <1>       ;
                            <1>       ; 20/11/2015
00012D95 80EA02             <1>       sub    dl, 3FAh-3F8h; data register (3F8h, 2F8h)
00012D98 EC                 <1>       in     al, dx       ; read character
                            <1>       ;JMP   $+2          ; I/O DELAY
                            <1>       ; 08/11/2015
                            <1>       ; 07/11/2015
00012D99 89DE               <1>       mov    esi, ebx
00012D9B 89DF               <1>       mov    edi, ebx
00012D9D 81C6[248A0100]     <1>       add    esi, rchar - 8 ; points to last received char
00012DA3 81C7[268A0100]     <1>       add    edi, schar - 8 ; points to last sent char
00012DA9 8806               <1>       mov    [esi], al ; received char (current char)
                            <1>       ; query
00012DAB 20C0               <1>       and    al, al
00012DAD 7527               <1>       jnz    short com_i2
                            <1>       ; response
                            <1>       ; 17/11/2015
                            <1>       ; set request for response status
00012DAF FE83[208A0100]     <1>         inc     byte [ebx+req_resp-8] ; 1
                            <1>       ;
00012DB5 6683C205           <1>       add    dx, 3FDh-3F8h; (3FDh, 2FDh)
00012DB9 EC                 <1>       in     al, dx       ; read line status register
00012DBA EB00               <1>       JMP    $+2          ; I/O DELAY
00012DBC 2420               <1>       and    al, 20h            ; transmitter holding reg. empty?
00012DBE 7445               <1>       jz     short com_eoi      ; no
00012DC0 B0FF               <1>       mov    al, 0FFh     ; response
00012DC2 6683EA05           <1>       sub    dx, 3FDh-3F8h      ; data port (3F8h, 2F8h)
00012DC6 EE                 <1>       out    dx, al       ; send on serial port
                            <1>       ; 17/11/2015
00012DC7 803F00             <1>       cmp    byte [edi], 0   ; query ? (schar)
00012DCA 7502               <1>       jne    short com_i1    ; no
00012DCC 8807               <1>       mov    [edi], al    ; 0FFh (responded)
                            <1> com_i1:
                            <1>       ; 17/11/2015
                            <1>       ; reset request for response status (again)
00012DCE FE8B[208A0100]     <1>         dec     byte [ebx+req_resp-8] ; 0
00012DD4 EB2F               <1>       jmp    short com_eoi
                            <1> com_i2:
                            <1>       ; 08/11/2015
00012DD6 3CFF               <1>       cmp    al, 0FFh     ; (response ?)
00012DD8 7417               <1>       je     short com_i3 ; (check for response signal)
                            <1>       ; 07/11/2015
00012DDA 3C04               <1>       cmp    al, 04h      ; EOT
00012DDC 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)
00012DDE 861D[EE890100]     <1>       xchg   bl, [ptty]  ; tty number (8 or 9)
00012DE4 E8F747FFFF         <1>       call   ctrlbrk
00012DE9 861D[EE890100]     <1>       xchg   [ptty], bl ; (restore ptty value and BL value)
                            <1>       ;mov   al, 04h ; EOT
                            <1>       ; 08/11/2015
00012DEF 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
00012DF1 803F00             <1>         cmp     byte [edi], 0 ; query ? (schar)
00012DF4 7704               <1>       ja     short com_i4 ; no
                            <1>       ; reset query status (schar)
00012DF6 8807               <1>       mov    [edi], al ; 0FFh
00012DF8 FEC0               <1>       inc    al ; 0
                            <1> com_i4:
                            <1>       ; 27/07/2014
                            <1>       ; 09/07/2014
00012DFA D0E3               <1>       shl    bl, 1
00012DFC 81C3[F0890100]     <1>       add    ebx, ttychr
                            <1>       ; 23/07/2014 (always overwrite)
                            <1>       ;;cmp  word [ebx], 0
                            <1>       ;;ja   short com_eoi
                            <1>       ;
00012E02 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 ; *
00012E05 A0[2A8A0100]   <1>         mov    al, byte [ccomport] ; current COM port
                        <1>          ; al = tty number (8 or 9)
00012E0A E85E010000     <1>           call wakeup
                        <1> com_iret:
                        <1>         ; 23/10/2015
00012E0F 5A             <1>         pop    edx ; **
00012E10 59             <1>         pop    ecx ; ***
                        <1>         ; 18/11/2015
                        <1>         ;pop   eax ; ****
                        <1>         ;mov   cr3, eax
                        <1>         ;jmp   iiret
00012E11 E995DFFEFF     <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>         ;
00012E16 BB[268A0100]   <1>         mov    ebx, com1p           ; COM1 parameters
00012E1B 66BAF803       <1>         mov    dx, 3F8h             ; COM1
                        <1>          ; 29/10/2015
00012E1F 66B90103       <1>         mov    cx, 301h  ; divisor = 1 (115200 baud)
00012E23 E86F000000     <1>         call   sp_i3  ; call A4
00012E28 A880           <1>         test   al, 80h
00012E2A 7410           <1>         jz     short sp_i0 ; OK..
                        <1>                ; Error !
                        <1>         ;mov   dx, 3F8h
00012E2C 80EA05         <1>         sub    dl, 5 ; 3FDh -> 3F8h
00012E2F 66B90E03       <1>         mov    cx, 30Eh  ; divisor = 12 (9600 baud)
00012E33 E85F000000     <1>         call   sp_i3  ; call A4
00012E38 A880           <1>         test   al, 80h
00012E3A 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>         ;
00012E3C C603E3         <1>         mov    byte [ebx], 0E3h ; 11100011b
00012E3F E8DC000000     <1>         call   sp_i5 ; 29/06/2015
                        <1> sp_i1:
00012E44 43             <1>         inc    ebx
```

```
00012E45 66BAF802          <1>       mov    dx, 2F8h           ; COM2
                           <1>        ; 29/10/2015
00012E49 66B90103          <1>       mov    cx, 301h  ; divisor = 1 (115200 baud)
00012E4D E845000000        <1>       call   sp_i3  ; call A4
00012E52 A880              <1>       test   al, 80h
00012E54 7410              <1>       jz     short sp_i2 ; OK..
                           <1>              ; Error !
                           <1>       ;mov   dx, 2F8h
00012E56 80EA05            <1>       sub    dl, 5 ; 2FDh -> 2F8h
00012E59 66B90E03          <1>       mov    cx, 30Eh  ; divisor = 12 (9600 baud)
00012E5D E835000000        <1>       call   sp_i3  ; call A4
00012E62 A880              <1>       test   al, 80h
00012E64 7530              <1>       jnz    short sp_i7
                           <1> sp_i2:
00012E66 C603E3            <1>       mov    byte [ebx], 0E3h ; 11100011b
                           <1> sp_i6:
                           <1>       ;; COM2 - enabling IRQ 3
                           <1>       ; 07/11/2015
                           <1>       ; 26/10/2015
00012E69 9C                <1>       pushf
00012E6A FA                <1>       cli
                           <1>       ;
00012E6B 66BAFC02          <1>       mov    dx, 2FCh           ; modem control register
00012E6F EC                <1>       in     al, dx                   ; read register
00012E70 EB00              <1>       JMP    $+2                ; I/O DELAY
00012E72 0C08              <1>       or     al, 8              ; enable bit 3 (OUT2)
00012E74 EE                <1>       out    dx, al             ; write back to register
00012E75 EB00              <1>       JMP    $+2                ; I/O DELAY
00012E77 66BAF902          <1>       mov    dx, 2F9h           ; interrupt enable register
00012E7B EC                <1>       in     al, dx             ; read register
00012E7C EB00              <1>       JMP    $+2                ; I/O DELAY
                           <1>       ;or    al, 1              ; receiver data interrupt enable and
00012E7E 0C03              <1>       or     al, 3              ; transmitter empty interrupt enable
00012E80 EE                <1>       out    dx, al                   ; write back to register
00012E81 EB00              <1>       JMP    $+2                ; I/O DELAY
00012E83 E421              <1>       in     al, 21h            ; read interrupt mask register
00012E85 EB00              <1>       JMP    $+2                ; I/O DELAY
00012E87 24F7              <1>       and    al, 0F7h           ; enable IRQ 3 (COM2)
00012E89 E621              <1>       out    21h, al            ; write back to register
                           <1>       ;
                           <1>       ; 23/10/2015
00012E8B B8[492D0100]      <1>       mov    eax, com2_int
00012E90 A3[682F0100]      <1>       mov    [com2_irq3], eax
                           <1>       ; 26/10/2015
00012E95 9D                <1>       popf
                           <1> sp_i7:
00012E96 C3                <1>       retn
                           <1>
                           <1> sp_i3:
                           <1> ;A4:        ;----- INITIALIZE THE COMMUNICATIONS PORT
                           <1>       ; 28/10/2015
00012E97 FEC2              <1>       inc    dl     ; 3F9h (2F9h); 3F9h, COM1 Interrupt enable register
00012E99 B000              <1>       mov    al, 0
00012E9B EE                <1>       out    dx, al             ; disable serial port interrupt
00012E9C EB00              <1>       JMP    $+2                ; I/O DELAY
00012E9E 80C202            <1>       add    dl, 2  ; 3FBh (2FBh); COM1 Line control register (3FBh)
00012EA1 B080              <1>       mov    al, 80h
00012EA3 EE                <1>       out    dx, al             ; SET DLAB=1 ; divisor latch access bit
                           <1>       ;----- SET BAUD RATE DIVISOR
                           <1>       ; 26/10/2015
00012EA4 80EA03            <1>       sub    dl, 3   ; 3F8h (2F8h)      ; register for least significant byte
                           <1>                                   ; of the divisor value
00012EA7 88C8              <1>       mov    al, cl ; 1
00012EA9 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)
00012EAA EB00              <1>       JMP    $+2                ; I/O DELAY
00012EAC 28C0              <1>       sub    al, al
00012EAE FEC2              <1>       inc    dl      ; 3F9h (2F9h)      ; register for most significant byte
                           <1>                                   ; of the divisor value
00012EB0 EE                <1>       out    dx, al ; 0
00012EB1 EB00              <1>       JMP    $+2                ; I/O DELAY
                           <1>       ;
00012EB3 88E8              <1>       mov    al, ch ; 3         ; 8 data bits, 1 stop bit, no parity
                           <1>       ;and   al, 1Fh ; Bits 0,1,2,3,4
00012EB5 80C202            <1>       add    dl, 2  ; 3FBh (2FBh); Line control register
00012EB8 EE                <1>       out    dx, al
00012EB9 EB00              <1>       JMP    $+2                ; I/O DELAY
                           <1>       ; 29/10/2015
00012EBB FECA              <1>       dec    dl     ; 3FAh (2FAh); FIFO Control register (16550/16750)
00012EBD 30C0              <1>       xor    al, al             ; 0
00012EBF EE                <1>       out    dx, al             ; Disable FIFOs (reset to 8250 mode)
00012EC0 EB00              <1>       JMP    $+2
                           <1> sp_i4:
                           <1> ;A18: ;----- COMM PORT STATUS ROUTINE
                           <1>       ; 29/06/2015 (line status after modem status)
00012EC2 80C204            <1>       add    dl, 4  ; 3FEh (2FEh); Modem status register
                           <1> sp_i4s:
00012EC5 EC                <1>       in     al, dx             ; GET MODEM CONTROL STATUS
00012EC6 EB00              <1>       JMP    $+2                ; I/O DELAY
00012EC8 88C4              <1>       mov    ah, al             ; PUT IN (AH) FOR RETURN
00012ECA FECA              <1>       dec    dl     ; 3FDh (2FDh); POINT TO LINE STATUS REGISTER
                           <1>                                   ; dx = 3FDh for COM1, 2FDh for COM2
00012ECC EC                <1>       in     al, dx             ; GET LINE CONTROL STATUS
                           <1>       ; AL = Line status, AH = Modem status
00012ECD C3                <1>       retn
                           <1>
                           <1> sp_status:
                           <1>       ; 29/06/2015
                           <1>       ; 27/06/2015 (Retro UNIX 386 v1)
                           <1>       ; Get serial port status
00012ECE 66BAFE03          <1>       mov    dx, 3FEh           ; Modem status register (COM1)
00012ED2 28C6              <1>       sub    dh, al             ; dh = 2 for COM2 (al = 1)
```

```
                        <1>                                     ; dx = 2FEh for COM2
00012ED4 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>     ;
00012ED6 66BAF803       <1>     mov    dx, 3F8h
00012EDA BB[268A0100]   <1>     mov    ebx, com1p ; COM1 control byte offset
00012EDF 3C01           <1>     cmp    al, 1
00012EE1 776B           <1>     ja     short sp_invp_err
00012EE3 7203           <1>     jb     short sp_setp1 ;  COM1 (AL = 0)
00012EE5 FECE           <1>     dec    dh ; 2F8h
00012EE7 43             <1>     inc    ebx ; COM2 control byte offset
                        <1> sp_setp1:
                        <1>     ; 29/10/2015
00012EE8 8823           <1>     mov    [ebx], ah
00012EEA 0FB6CC         <1>     movzx  ecx, ah
00012EED C0E905         <1>     shr    cl, 5 ; -> baud rate index
00012EF0 80E41F         <1>     and    ah, 1Fh ; communication parameters except baud rate
00012EF3 8A81[5D2F0100] <1>     mov    al, [ecx+b_div_tbl]
00012EF9 6689C1         <1>     mov    cx, ax
00012EFC E896FFFFFF     <1>     call   sp_i3
00012F01 6689C1         <1>     mov    cx, ax ; CL = Line status, CH = Modem status
00012F04 A880           <1>     test   al, 80h
00012F06 740F           <1>     jz     short sp_setp2
00012F08 C603E3         <1>       mov     byte [ebx], 0E3h ; Reset to initial value (11100011b)
                        <1> stp_dnr_err:
00012F0B C705[C8030300]0F00- <1>     mov    dword [u.error], ERR_DEV_NOT_RDY ; 'device not ready !'
00012F13 0000           <1>
                        <1>     ; CL = Line status, CH = Modem status
00012F15 F9             <1>     stc
00012F16 C3             <1>     retn
                        <1> sp_setp2:
00012F17 80FE02         <1>     cmp    dh, 2 ; COM2 (2F?h)
00012F1A 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
                        <1>     pushf
                        <1>     cli
00012F22 66BAFC03       <1>     mov    dx, 3FCh           ; modem control register
00012F26 EC             <1>     in     al, dx                  ; read register
00012F27 EB00           <1>     JMP    $+2                ; I/O DELAY
00012F29 0C08           <1>     or     al, 8              ; enable bit 3 (OUT2)
00012F2B EE             <1>     out    dx, al             ; write back to register
00012F2C EB00           <1>     JMP    $+2                ; I/O DELAY
00012F2E 66BAF903       <1>     mov    dx, 3F9h           ; interrupt enable register
00012F32 EC             <1>     in     al, dx             ; read register
00012F33 EB00           <1>     JMP    $+2                ; I/O DELAY
                        <1>     ;or    al, 1              ; receiver data interrupt enable and
00012F35 0C03           <1>     or     al, 3              ; transmitter empty interrupt enable
00012F37 EE             <1>     out    dx, al                  ; write back to register
00012F38 EB00           <1>     JMP    $+2                ; I/O DELAY
00012F3A E421           <1>     in     al, 21h            ; read interrupt mask register
00012F3C EB00           <1>     JMP    $+2                ; I/O DELAY
00012F3E 24EF           <1>     and    al, 0EFh           ; enable IRQ 4 (COM1)
00012F40 E621           <1>     out    21h, al            ; write back to register
                        <1>     ;
                        <1>     ; 23/10/2015
00012F42 B8[522D0100]   <1>     mov    eax, com1_int
00012F47 A3[642F0100]   <1>     mov    [com1_irq4], eax
                        <1>     ; 26/10/2015
```

```
00012F4C 9D                    <1>       popf
00012F4D C3                    <1>       retn
                               <1>
                               <1> sp_invp_err:
00012F4E C705[C8030300]1700-   <1>       mov    dword [u.error], ERR_INV_PARAMETER ; 'invalid parameter !'
00012F56 0000                  <1>
00012F58 31C9                  <1>       xor    ecx, ecx
00012F5A 49                    <1>       dec    ecx ; 0FFFFh
00012F5B F9                    <1>       stc
00012F5C C3                    <1>       retn
                               <1>
                               <1> ; 29/10/2015
                               <1> b_div_tbl: ; Baud rate divisor table (115200/divisor)
00012F5D 010C0603080401        <1>       db 1, 12, 6, 3, 8, 4, 1
                               <1>
                               <1>
                               <1> ; 23/10/2015
                               <1> com1_irq4:
00012F64 [6C2F0100]            <1>       dd dummy_retn
                               <1> com2_irq3:
00012F68 [6C2F0100]            <1>       dd dummy_retn
                               <1>
                               <1> dummy_retn:
00012F6C C3                    <1>       retn
                               <1>
                               <1> wakeup:
                               <1>       ; 24/01/2016
00012F6D C3                    <1>       retn
                               <1>
                               <1> set_working_path_x:
                               <1>             ; 17/10/2016 (TRDOS 386 - FFF & FNF)
00012F6E 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>
00012F72 8825[74960100]        <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>
00012F78 66A3[78960100]        <1>             mov    [SWP_Mode], ax
00012F7E A0[868A0100]          <1>             mov    al, [Current_Drv]
00012F83 30E4                  <1>             xor    ah, ah
00012F85 66A3[7A960100]        <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>
00012F8B 55                    <1>             push   ebp
00012F8C 89E5                  <1>             mov    ebp, esp
                               <1>
00012F8E B980000000            <1>             mov    ecx, 128 ; maximum path length = 128 bytes
00012F93 29CC                  <1>             sub    esp, ecx ; reserve 128 bytes (buffer) on stack
```

```
00012F95 89E7              <1>             mov    edi, esp ; destination address (kernel space)
                           <1>             ; esi = source address (virtual, in user's memory space)
00012F97 E8A0EBFFFF        <1>             call   transfer_from_user_buffer
00012F9C 720A              <1>             jc     short loc_swp_xor_retn
                           <1>
00012F9E 89E6              <1>             mov    esi, esp ; temporary buffer (the path) on stack
                           <1> loc_swp_fchar:
00012FA0 8A06              <1>             mov    al, [esi]
00012FA2 3C20              <1>             cmp    al, 20h
00012FA4 7711              <1>             ja     short loc_swp_parse_path_name
00012FA6 740C              <1>             je     short loc_swp_fchar_next
                           <1>
                           <1> loc_swp_xor_retn:
00012FA8 31C0              <1>             xor    eax, eax
00012FAA F9                <1>             stc
                           <1> loc_swp_retn:
00012FAB 89EC              <1>             mov    esp, ebp
00012FAD 5D                <1>             pop    ebp
                           <1>
                           <1>             ;mov   esi, FindFile_Drv
00012FAE BE[68930100]      <1>             mov    esi, FindFile_Name ; 12/10/2016
00012FB3 C3                <1>             retn
                           <1>
                           <1> loc_swp_fchar_next:
00012FB4 46                <1>             inc    esi
00012FB5 EBE9              <1>             jmp    short loc_swp_fchar
                           <1>
                           <1> loc_swp_parse_path_name:
00012FB7 BF[26930100]      <1>             mov    edi, FindFile_Drv
00012FBC E89186FFFF        <1>             call   parse_path_name
00012FC1 72E8              <1>             jc     short loc_swp_retn
                           <1>
                           <1> loc_swp_checkfile_name:
00012FC3 803D[78960100]00  <1>             cmp    byte [SWP_Mode], 0
00012FCA 761E              <1>             jna    short loc_swp_drv
                           <1>
                           <1>             ; 10/10/2016 (valid file name checking)
00012FCC BE[68930100]      <1>             mov    esi, FindFile_Name
00012FD1 803E20            <1>             cmp    byte [esi], 20h
00012FD4 76D2              <1>             jna    short loc_swp_xor_retn
                           <1>
                           <1>             ; 16/10/2016
00012FD6 C605[77960100]00  <1>             mov    byte [SWP_inv_fname], 0 ; reset
                           <1>             ; esi = file name address (ASCIIZ)
00012FDD E85D68FFFF        <1>             call   check_filename
00012FE2 7306              <1>             jnc    short loc_swp_drv
                           <1>
00012FE4 FE05[77960100]    <1>             inc    byte [SWP_inv_fname] ; set
                           <1> loc_swp_drv:
00012FEA 8A35[868A0100]    <1>             mov    dh, [Current_Drv]
                           <1>               ;mov      [RUN_CDRV], dh
                           <1>
00012FF0 8A15[26930100]    <1>             mov    dl, [FindFile_Drv]
                           <1>               ;cmp      dl, dh
00012FF6 3A15[868A0100]    <1>             cmp    dl, [Current_Drv]
00012FFC 740D              <1>             je     short loc_swp_change_directory
                           <1>
00012FFE FE05[7B960100]    <1>             inc    byte [SWP_DRV_chg]
00013004 E8D550FFFF        <1>             call   change_current_drive
00013009 72A0              <1>             jc     short loc_swp_retn ; eax = error code
                           <1>             ; eax = 0
                           <1>
                           <1> loc_swp_change_directory:
0001300B 803D[27930100]21  <1>             cmp    byte [FindFile_Directory], 21h
00013012 F5                <1>             cmc
00013013 7396              <1>             jnc    short loc_swp_retn
                           <1>
00013015 FE05[7B960100]    <1>             inc    byte [SWP_DRV_chg]
0001301B FE05[51400100]    <1>             inc    byte [Restore_CDIR]
00013021 BE[27930100]      <1>             mov    esi, FindFile_Directory
00013026 8A25[79960100]    <1>             mov    ah, [SWP_Mode+1]
0001302C E80B80FFFF        <1>             call   change_current_directory
00013031 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
00013037 E8257FFFFF        <1>             call   change_prompt_dir_str
0001303C 29C0              <1>             sub    eax, eax ; 0
0001303E 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>
00013043 31C0              <1>             xor    eax, eax
00013045 48                <1>             dec    eax
                           <1>
00013046 668B15[7A960100]  <1>             mov    dx, [SWP_DRV]
0001304D 08F6              <1>             or     dh, dh
0001304F 742E              <1>             jz     short loc_rwp_return
```

```
1039                                    <1>
1040 00013051 3A15[868A0100]            <1>             cmp    dl, [Current_Drv]
1041 00013057 7407                      <1>             je     short loc_rwp_restore_cdir
1042                                    <1> loc_rwp_restore_cdrv:
1043 00013059 E88050FFFF                <1>             call   change_current_drive
1044 0001305E EB10                      <1>             jmp    short loc_rwp_restore_ok
1045                                    <1> loc_rwp_restore_cdir:
1046 00013060 31DB                      <1>             xor    ebx, ebx
1047 00013062 88D7                      <1>             mov    bh, dl
1048 00013064 BE00010900                <1>             mov    esi, Logical_DOSDisks
1049 00013069 01DE                      <1>             add    esi, ebx
1050                                    <1>
1051 0001306B E82551FFFF                <1>             call   restore_current_directory
1052                                    <1>
1053                                    <1> loc_rwp_restore_ok:
1054 00013070 668B15[7A960100]          <1>             mov    dx, [SWP_DRV]
1055 00013077 31C0                      <1>             xor    eax, eax
1056 00013079 66A3[7B960100]            <1>             mov    [SWP_DRV_chg], ax
1057                                    <1> loc_rwp_return:
1058 0001307F C3                        <1>             retn
1059                                    <1>
1060                                    <1> get_file_name:
1061                                    <1>             ; 15/10/2016 - TRDOS 386 (TRDOS v2.0)
1062                                    <1>             ; Convert file name
1063                                    <1>             ;      from directory entry format
1064                                    <1>                ;  to (8.3) dot file name format
1065                                    <1>             ;
1066                                    <1>             ; TRDOS v1.0 (DIR.ASM, "get_file_name")
1067                                    <1>                ; 2005 - 09/10/2011
1068                                    <1>             ; INPUT:
1069                                    <1>             ;      DS:SI -> Directory Entry Format File Name
1070                                    <1>             ;        ES:DI -> DOS Dot File Name Address
1071                                    <1>             ; OUTPUT:
1072                                    <1>             ;      DS:SI -> DOS Dot File Name Address
1073                                    <1>                ;  ES:DI -> Directory Entry Format File Name
1074                                    <1>             ;
1075                                    <1>             ; TRDOS 386 (15/10/2016)
1076                                    <1>             ; INPUT:
1077                                    <1>             ;      ESI = File name addr in dir entry format
1078                                    <1>             ;      EDI = Dot file name address (destination)
1079                                    <1>             ; OUTPUT:
1080                                    <1>             ;      File name is converted and moved
1081                                    <1>             ;      to destination (as 8.3 dot filename)
1082                                    <1>             ;
1083                                    <1>             ; Modified registers: EAX, ECX
1084                                    <1>
1085                                    <1>                ; 2005 (TRDOS 8086) - 2016 (TRDOS 386)
1086                                    <1>
1087 00013080 57                        <1>             push   edi
1088 00013081 56                        <1>             push   esi
1089 00013082 AC                        <1>             lodsb
1090 00013083 3C20                      <1>             cmp    al, 20h
1091 00013085 762A                      <1>             jna    short pass_gfn_ext
1092 00013087 56                        <1>             push   esi
1093 00013088 AA                        <1>             stosb
1094 00013089 B907000000                <1>             mov    ecx, 7
1095                                    <1> loc_gfn_next_char:
1096 0001308E AC                        <1>             lodsb
1097 0001308F 3C20                      <1>             cmp    al, 20h
1098 00013091 7603                      <1>             jna    short pass_gfn_fn
1099 00013093 AA                        <1>             stosb
1100 00013094 E2F8                      <1>             loop   loc_gfn_next_char
1101                                    <1> pass_gfn_fn:
1102 00013096 5E                        <1>             pop    esi
1103 00013097 83C607                    <1>             add    esi, 7
1104 0001309A AC                        <1>             lodsb
1105 0001309B 3C20                      <1>             cmp    al, 20h
1106 0001309D 7612                      <1>             jna    short pass_gfn_ext
1107 0001309F B42E                      <1>             mov    ah, '.'
1108 000130A1 86E0                      <1>             xchg   ah, al
1109 000130A3 66AB                      <1>             stosw
1110 000130A5 AC                        <1>             lodsb
1111 000130A6 3C20                      <1>             cmp    al, 20h
1112 000130A8 7607                      <1>             jna    short pass_gfn_ext
1113 000130AA AA                        <1>             stosb
1114 000130AB AC                        <1>             lodsb
1115 000130AC 3C20                      <1>             cmp    al, 20h
1116 000130AE 7601                      <1>             jna    short pass_gfn_ext
1117 000130B0 AA                        <1>             stosb
1118                                    <1> pass_gfn_ext:
1119 000130B1 30C0                      <1>             xor    al, al
1120 000130B3 AA                        <1>             stosb
1121 000130B4 5E                        <1>             pop    esi
1122 000130B5 5F                        <1>             pop    edi
1123 000130B6 C3                        <1>             retn
1124                                    <1>
1125                                    <1> set_hardware_int_vector:
1126                                    <1>             ; 18/03/2017
1127                                    <1>             ; 03/03/2017
1128                                    <1>             ; 28/02/2017 - TRDOS 386 (TRDOS v2.0)
1129                                    <1>             ;
1130                                    <1>             ; SET/RESET HARDWARE INTERRUPT GATE
1131                                    <1>             ;
1132                                    <1>             ; Changes interrupt gate descriptor table
1133                                    <1>             ; (without changing default interrupt list)
1134                                    <1>             ;
1135                                    <1>             ; INPUT:
1136                                    <1>             ;      AL = IRQ number (0 to 15)
1137                                    <1>             ;      AH > 0 -> set
1138                                    <1>             ;      AH = 0 -> reset
1139                                    <1>             ;
1140                                    <1>             ; Modified registers: eax, ebx, edx, edi
1141                                    <1>             ;
1142                                    <1>
1143 000130B7 C0E002                    <1>             shl    al, 2 ; IRQ number * 4
```

```
1144 000130BA 0FB6D8              <1>             movzx  ebx, al
1145                              <1>
1146 000130BD 08E4                <1>             or     ah, ah
1147 000130BF 7508                <1>             jnz    short shintv_1 ; set (for user call service)
1148                              <1>
1149                              <1>             ; 18/03/2017
1150 000130C1 81C3[504A0100]      <1>             add    ebx, IRQ_list ; reset to default interrupt list
1151 000130C7 EB06                <1>             jmp    short shintv_2
1152                              <1> shintv_1:
1153 000130C9 81C3[F0300100]      <1>             add    ebx, IRQ_u_list
1154                              <1> shintv_2:
1155 000130CF 8B13                <1>             mov    edx, [ebx] ; IRQ handler address
1156                              <1>
1157                              <1>             ; 03/03/2017
1158 000130D1 D0E0                <1>             shl    al, 1 ; IRQ number * 8
1159                              <1>             ; 18/03/2017
1160 000130D3 0FB6F8              <1>             movzx  edi, al
1161 000130D6 81C7[D8870100]      <1>             add    edi, idt + (8*32) ; IRQ 0 offset = idt + 256
1162                              <1>
1163 000130DC 89D0                <1>             mov    eax, edx ; IRQ handler address
1164 000130DE BB00000800          <1>             mov    ebx, 80000h
1165                              <1>
1166                              <1>             ;mov   edx, eax
1167 000130E3 66BA008E            <1>             mov    dx, 8E00h
1168 000130E7 6689C3              <1>             mov    bx, ax
1169 000130EA 89D8                <1>             mov    eax, ebx ; /* selector = 0x0008 = cs */
1170                              <1>                                     ; /* interrupt gate - dpl=0, present */
1171 000130EC AB                  <1>             stosd  ; selector & offset bits 0-15
1172 000130ED 8917                <1>             mov    [edi], edx ; attributes & offset bits 16-23
1173                              <1>
1174 000130EF C3                  <1>             retn
1175                              <1> IRQ_u_list:
1176                              <1>             ; 28/02/2017
1177 000130F0 [56090000]          <1>             dd     timer_int
1178 000130F4 [CE100000]          <1>             dd     kb_int
1179 000130F8 [380B0000]          <1>             dd     irq2
1180 000130FC [30310100]          <1>             dd     IRQ_service3
1181 00013100 [3A310100]          <1>             dd     IRQ_service4
1182 00013104 [44310100]          <1>             dd     IRQ_service5
1183 00013108 [D3510000]          <1>             dd     fdc_int
1184 0001310C [4E310100]          <1>             dd     IRQ_service7
1185 00013110 [C10A0000]          <1>             dd     rtc_int
1186 00013114 [58310100]          <1>             dd     IRQ_service9
1187 00013118 [62310100]          <1>             dd     IRQ_service10
1188 0001311C [6C310100]          <1>             dd     IRQ_service11
1189 00013120 [76310100]          <1>             dd     IRQ_service12
1190 00013124 [80310100]          <1>             dd     IRQ_service13
1191 00013128 [865B0000]          <1>             dd     hdc1_int
1192 0001312C [AD5B0000]          <1>             dd     hdc2_int
1193                              <1>
1194                              <1>             ; 03/03/2017
1195                              <1>             ; 27/02/2017
1196                              <1> IRQ_service3:
1197 00013130 36C605[3E9C0100]03  <1>             mov    byte [ss:IRQnum], 3
1198 00013138 EB4E                <1>             jmp    short IRQ_service
1199                              <1> IRQ_service4:
1200 0001313A 36C605[3E9C0100]04  <1>             mov    byte [ss:IRQnum], 4
1201 00013142 EB44                <1>             jmp    short IRQ_service
1202                              <1> IRQ_service5:
1203 00013144 36C605[3E9C0100]05  <1>             mov    byte [ss:IRQnum], 5
1204 0001314C EB3A                <1>             jmp    short IRQ_service
1205                              <1> IRQ_service7:
1206 0001314E 36C605[3E9C0100]07  <1>             mov    byte [ss:IRQnum], 7
1207 00013156 EB30                <1>             jmp    short IRQ_service
1208                              <1> IRQ_service9:
1209 00013158 36C605[3E9C0100]09  <1>             mov    byte [ss:IRQnum], 9
1210 00013160 EB26                <1>             jmp    short IRQ_service
1211                              <1> IRQ_service10:
1212 00013162 36C605[3E9C0100]0A  <1>             mov    byte [ss:IRQnum], 10
1213 0001316A EB1C                <1>             jmp    short IRQ_service
1214                              <1> IRQ_service11:
1215 0001316C 36C605[3E9C0100]0B  <1>             mov    byte [ss:IRQnum], 11
1216 00013174 EB12                <1>             jmp    short IRQ_service
1217                              <1> IRQ_service12:
1218 00013176 36C605[3E9C0100]0C  <1>             mov    byte [ss:IRQnum], 12
1219 0001317E EB08                <1>             jmp    short IRQ_service
1220                              <1> IRQ_service13:
1221 00013180 36C605[3E9C0100]0D  <1>             mov    byte [ss:IRQnum], 13
1222                              <1>             ;jmp   short IRQ_service
1223                              <1> IRQ_service:
1224                              <1>             ; 13/06/2017
1225                              <1>             ; 11/06/2017
1226                              <1>             ; 10/06/2017
1227                              <1>             ; 01/03/2017, 04/03/2017
1228                              <1>             ; 27/02/2017, 28/02/2017
1229 00013188 1E                  <1>             push   ds
1230 00013189 06                  <1>             push   es
1231 0001318A 0FA0                <1>             push   fs
1232 0001318C 0FA8                <1>             push   gs
1233                              <1>
1234 0001318E 60                  <1>             pushad ; eax,ecx,edx,ebx,esp,ebp,esi,edi
1235 0001318F 66B91000            <1>             mov     cx, KDATA
1236 00013193 8ED9                <1>             mov     ds, cx
1237 00013195 8EC1                <1>             mov     es, cx
1238 00013197 8EE1                <1>             mov     fs, cx
1239 00013199 8EE9                <1>             mov     gs, cx
1240                              <1>
1241 0001319B 0F20D8              <1>             mov    eax, cr3
1242 0001319E A3[3A9C0100]        <1>             mov    [IRQ_cr3], eax
1243                              <1>
1244 000131A3 A1[C0890100]        <1>             mov    eax, [k_page_dir]
1245 000131A8 0F22D8              <1>             mov    cr3, eax
1246                              <1>
1247 000131AB A0[3E9C0100]        <1>             mov    al, [IRQnum]
1248                              <1>
```

```
1249                                    <1>             ;mov   cl, [sysflg]
1250                                    <1>             ;mov   [u.r_mode], cl  ; system (0) or user mode (FFh)
1251                                    <1> IRQsrv_0:
1252 000131B0 0FB6D8                   <1>             movzx  ebx, al
1253 000131B3 8A9B[86490100]            <1>             mov    bl, [ebx+IRQenum] ; IRQ (available) index number + 1
1254                                    <1>             ; 01/03/2017
1255 000131B9 FECB                      <1>             dec    bl  ; IRQ index number, 0 to 8
1256 000131BB 0F8807010000              <1>             js     IRQsrv_5 ; not available to use here!?
1257                                    <1>             ;
1258 000131C1 80BB[049C0100]80          <1>             cmp    byte [ebx+IRQ.method], 80h ; using by a dev or kernel?
1259 000131C8 7205                      <1>             jb     short IRQsrv_1 ; no
1260                                    <1>
1261                                    <1>             ; If the IRQ service is already owned by TRDOS 386 kernel
1262                                    <1>             ;      or a Device driver
1263                                    <1>             ; we need to call 'dev_IRQ_service'
1264                                    <1>
1265                                    <1>             ; IRQ number in AL
1266 000131CA E866020000                <1>             call   dev_IRQ_service      ; IRQ service for device drivers
1267                                    <1>             ; IRQ number in AL
1268                                    <1> IRQsrv_1:
1269                                    <1>             ; check user callback service status
1270                                    <1>             ; AL = IRQ number
1271                                    <1>             ; EBX = IRQ (Available) Index number
1272                                    <1>
1273 000131CF A2[D7030300]              <1>             mov    [u.irqwait], al ; set waiting IRQ flag
1274                                    <1>
1275 000131D4 8A83[F29B0100]            <1>             mov    al, [ebx+IRQ.owner]
1276 000131DA 20C0                      <1>             and    al, al
1277 000131DC 0F84E6000000              <1>             jz     IRQsrv_5 ; it is not owned by a user/proc
1278                                    <1>
1279                                    <1>             ; 03/03/2017
1280 000131E2 89DA                      <1>             mov    edx, ebx
1281 000131E4 C0E202                    <1>             shl    dl, 2
1282 000131E7 8B92[169C0100]            <1>             mov    edx, [edx+IRQ.addr] ; S.R.B. or Callback service addr
1283                                    <1>
1284 000131ED 8AA3[049C0100]            <1>             mov    ah, [ebx+IRQ.method]
1285 000131F3 F6C401                    <1>             test   ah, 1
1286 000131F6 7534                      <1>             jnz    short IRQsrv_4 ; Callback service method
1287                                    <1>
1288                                    <1>             ; Signal Response Byte method
1289                                    <1>             ;mov   edx, [edx+IRQ.addr] ; Signal Response Byte address
1290                                    <1>             ;                          ; (Physical address, non-swappable)
1291 000131F8 80E402                    <1>             and    ah, 2 ; bit 1, (S.R.B.) counter (auto increment) method
1292 000131FB 8AA3[0D9C0100]            <1>             mov    ah, [ebx+IRQ.srb] ; Signal Response Byte value
1293 00013201 7408                      <1>             jz     short IRQsrv_2 ; fixed S.R.B. value
1294                                    <1>             ; counter method (auto increment)
1295 00013203 FEC4                      <1>             inc    ah
1296 00013205 88A3[0D9C0100]            <1>             mov    [ebx+IRQ.srb], ah ; next (count) number
1297                                    <1> IRQsrv_2:
1298 0001320B 8822                      <1>             mov    [edx], ah ; put S.R.B. val to the user's S.R.B. addr
1299 0001320D C605[D7030300]00          <1>             mov    byte [u.irqwait], 0 ; clear waiting IRQ flag
1300                                    <1>
1301 00013214 3A05[B3030300]            <1>             cmp    al, [u.uno]
1302 0001321A 0F84A8000000              <1>             je     IRQsrv_5 ; the owner is current user/process
1303                                    <1> IRQsrv_3:
1304                                    <1>             ; the owner is not current user/process
1305                                    <1>             ; AL = process number
1306 00013220 B202                      <1>             mov    dl, 2 ; priority, 2 = event (high)
1307 00013222 E837FAFFFF                <1>             call   set_run_sequence
1308                                    <1>
1309                                    <1>             ; [u.irqwait] = waiting IRQ number for callback service
1310                                    <1>
1311 00013227 E99C000000                <1>             jmp    IRQsrv_5
1312                                    <1> IRQsrv_4:
1313 0001322C 3A05[B3030300]            <1>             cmp    al, [u.uno]  ; is the owner is current user/process?
1314 00013232 75EC                      <1>             jne    short IRQsrv_3 ; no !
1315                                    <1>
1316                                    <1>             ; Check if an IRQ callback service already in progress
1317 00013234 803D[D8030300]00          <1>             cmp    byte [u.r_lock], 0
1318 0001323B 0F8787000000              <1>             ja     IRQsrv_5 ; nothing to do !
1319                                    <1>                                     ; (we need to complete prev callback)
1320 00013241 803D[D4030300]00          <1>             cmp    byte [u.t_lock], 0
1321 00013248 777E                      <1>             ja     short IRQsrv_5 ; nothing to do !
1322                                    <1>                                     ; (we need to complete timer callback)
1323                                    <1>
1324                                    <1>             ; 04/03/2017
1325 0001324A C605[D7030300]00          <1>             mov    byte [u.irqwait], 0 ; reset/clear waiting IRQ flag
1326                                    <1>
1327 00013251 FE05[D8030300]            <1>             inc    byte [u.r_lock] ; 'IRQ callback service in progress' flag
1328                                    <1>
1329 00013257 8A0D[5B030300]            <1>             mov    cl, [sysflg]   ; (system call) mode flag (kernel/user)
1330 0001325D 880D[D9030300]            <1>             mov    [u.r_mode], cl ; system mode (0) or user mode (FFh)
1331                                    <1>
1332                                    <1>             ;
1333 00013263 8B2D[5C890100]            <1>             mov    ebp, [tss.esp0] ; kernel stack address (for ring 0)
1334 00013269 83ED14                    <1>             sub    ebp, 20             ; eip, cs, eflags, esp, ss
1335 0001326C 892D[5C030300]            <1>             mov    [u.sp], ebp
1336 00013272 8925[60030300]            <1>             mov    [u.usp], esp
1337                                    <1>
1338                                    <1>             ;or    word [ebp+8], 200h ; 22/01/2017, force enabling interrupts
1339                                    <1>
1340 00013278 8B44241C                  <1>             mov    eax, [esp+28] ; pushed eax
1341 0001327C A3[64030300]              <1>             mov    [u.r0], eax
1342                                    <1>
1343 00013281 E820E7FFFF                <1>             call   wswap ; save user's registers & status
1344                                    <1>
1345                                    <1>             ; software int is in ring 0 but IRQ handler must return to ring 3
1346                                    <1>             ; so, ring 3 return address and stack registers
1347                                    <1>             ; (eip, cs, eflags, esp, ss)
1348                                    <1>             ; must be copied to IRQ handler return
1349                                    <1>             ; eip will be replaced by callback service routine address
1350                                    <1>
1351 00013286 C605[5B030300]FF          <1>             mov    byte [sysflg], 0FFh ; user mode
1352                                    <1>
1353                                    <1>             ; system mode (system call)
```

```
1354                                 <1>             ;mov   ebp, [u.sp] ; EIP (u), CS (UCODE), EFLAGS (u),
1355                                 <1>                                ; ESP (u), SS (UDATA)
1356                                 <1>
1357 0001328D 8B4510               <1>             mov    eax, [ebp+16]; SS (UDATA)
1358 00013290 89E6                 <1>             mov    esi, esp
1359 00013292 50                   <1>             push   eax
1360 00013293 50                   <1>             push   eax
1361 00013294 89E7                 <1>             mov    edi, esp
1362 00013296 893D[60030300]       <1>             mov    [u.usp], edi
1363 0001329C B908000000           <1>             mov    ecx, ((ESPACE/4) - 4) ; except DS, ES, FS, GS
1364 000132A1 F3A5                 <1>             rep    movsd
1365 000132A3 B104                 <1>             mov    cl, 4
1366 000132A5 F3AB                 <1>             rep    stosd
1367 000132A7 893D[5C030300]       <1>             mov    [u.sp], edi
1368 000132AD 89EE                 <1>             mov    esi, ebp
1369 000132AF B105                 <1>             mov    cl, 5 ; EIP (u), CS (UCODE), EFLAGS (u), ESP (u), SS (UDATA)
1370 000132B1 F3A5                 <1>             rep    movsd
1371                                 <1>             ;
1372                                 <1>
1373 000132B3 8B0D[B8030300]       <1>             mov    ecx, [u.pgdir]
1374 000132B9 890D[3A9C0100]       <1>             mov    [IRQ_cr3], ecx
1375                                 <1>
1376                                 <1> set_IRQ_callback_addr:
1377                                 <1>             ;
1378                                 <1>             ; This routine sets return address
1379                                 <1>             ; to start of user's interrupt
1380                                 <1>             ; service (callback) address
1381                                 <1>             ;
1382                                 <1>             ; INPUT:
1383                                 <1>             ;      EDX = callback routine/service address
1384                                 <1>             ;            (virtual, not physical address!)
1385                                 <1>             ;      [u.sp] = kernel stack, points to
1386                                 <1>             ;             user's EIP,CS,EFLAGS,ESP,SS
1387                                 <1>             ;             registers.
1388                                 <1>             ; OUTPUT:
1389                                 <1>             ;      EIP (user) = callback (service) address
1390                                 <1>             ;      CS (user) = UCODE
1391                                 <1>             ;      EFLAGS (user) = flags before callback
1392                                 <1>             ;       ESP (user) = ESP-4 (user, before callback)
1393                                 <1>             ;      [ESP](user) = EIP (user) before callback
1394                                 <1>             ;
1395                                 <1>             ; Note: If CPU was in user mode while entering
1396                                 <1>             ;      the timer interrupt service routine,
1397                                 <1>             ;      'IRET' will get return to callback routine
1398                                 <1>             ;      immediately. If CPU was in system/kernel mode
1399                                 <1>             ;      'iret' will get return to system call and
1400                                 <1>             ;      then, callback routine will be return address
1401                                 <1>             ;      from system call. (User's callback/service code
1402                                 <1>             ;      will be able to return to normal return address
1403                                 <1>             ;      via a 'sysrele' system call at the end.)
1404                                 <1>             ;
1405                                 <1>             ; Note: User's IRQ callback service code must be ended
1406                                 <1>             ;      with a 'sysrele' system call !
1407                                 <1>             ;
1408                                 <1>             ;      For example:
1409                                 <1>             ;
1410                                 <1>             ;      audio_IRQ_callback:
1411                                 <1>             ;          ...
1412                                 <1>             ;          <load DMA buffer with audio data>
1413                                 <1>             ;          ...
1414                                 <1>             ;          mov eax, 39 ; 'sysrele'
1415                                 <1>             ;          int 40h ; TRDOS 386 system call (interrupt)
1416                                 <1>             ;
1417                                 <1>
1418                                 <1>             ;mov   edx, [edx+IRQ.addr] ; Callback service address
1419                                 <1>             ;                          ; (Virtual address)
1420                                 <1>
1421 000132BF 8B2D[5C030300]       <1>             mov    ebp, [u.sp]; kernel's stack, points to EIP (user)
1422 000132C5 895500               <1>             mov    [ebp], edx
1423                                 <1> IRQsrv_5:
1424                                 <1>             ; EOI & return
1425                                 <1>             ; 01/08/2020
1426                                 <1>             ; 11/06/2017
1427                                 <1>             ; 10/06/2017
1428                                 <1>             ;mov   al, [IRQnum]
1429 000132C8 B020                 <1>             mov    al, 20h ; 01/08/2020
1430 000132CA FA                   <1>             cli
1431                                 <1>             ;cmp   al, 7
1432 000132CB 803D[3E9C0100]07     <1>             cmp    byte [IRQnum], 7 ; 01/08/2020
1433 000132D2 7602                 <1>             jna    short IRQsrv_6
1434                                 <1>             ;
1435                                 <1>             ;;mov  al, EOI      ; end of interrupt
1436                                 <1>             ;mov   al, 20h ; 01/08/2020
1437                                 <1>             ;cli         ; disable interrupts till stack cleared
1438                                 <1>             ;out   INTB00, al ; For controll2 #2
1439 000132D4 E6A0                 <1>             out    0A0h, al
1440                                 <1> IRQsrv_6:
1441                                 <1>             ;mov   byte [IRQnum], 0 ; reset
1442                                 <1>             ;;mov  al, EOI      ; end of interrupt
1443                                 <1>             ;mov   al, 20h ; 01/08/2020
1444                                 <1>             ;cli         ; disable interrupts till stack cleared
1445                                 <1>             ;out   INTA00, al ; end of interrupt to 8259 - 1
1446 000132D6 E620                 <1>             out    20h, al
1447                                 <1> IRQsrv_7:
1448                                 <1>             ;; 13/06/2017
1449                                 <1>             ;or    word [ebp+8], 200h ; force enabling interrupts
1450                                 <1>             ;
1451 000132D8 8B0D[3A9C0100]       <1>             mov    ecx, [IRQ_cr3]      ; previous content of cr3 register
1452 000132DE 0F22D9               <1>             mov    cr3, ecx     ; restore cr3 register content
1453                                 <1>             ;
1454 000132E1 61                   <1>             popad ; edi,esi,ebp,(icrement esp by 4),ebx,edx,ecx,eax
1455                                 <1>             ;
1456 000132E2 0FA9                 <1>             pop    gs
1457 000132E4 0FA1                 <1>             pop    fs
1458 000132E6 07                   <1>             pop    es
```

```
1459 000132E7 1F                <1>             pop    ds
1460                            <1>             ;
1461 000132E8 CF                <1>             iretd  ; return from interrupt
1462                            <1>
1463                            <1> get_device_number:
1464                            <1>             ; 08/10/2016
1465                            <1>             ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
1466                            <1>             ;
1467                            <1>             ; This procedure compares name of requested
1468                            <1>             ; device with kernel device names and
1469                            <1>             ; installable device names. If names match,
1470                            <1>             ; the relevant device index (entry) number
1471                            <1>             ; will be returned the caller (sysopen)
1472                            <1>             ; for the requested device.
1473                            <1>             ;
1474                            <1>             ; NOTE: Installable device drivers must
1475                            <1>             ; be loaded before using 'sysopen'
1476                            <1>             ; (opendev) system call.
1477                            <1>             ;
1478                            <1>             ; INPUT:
1479                            <1>             ;    ESI = device name address (ASCIIZ)
1480                            <1>             ;         (in kernel's memory space)
1481                            <1>             ;    max name length = 8 without '/dev/')
1482                            <1>             ;    Device name will be capitalized
1483                            <1>             ;    and if there is, '/dev/' will be
1484                            <1>             ;    removed from name before comparising)
1485                            <1>             ;
1486                            <1>             ; OUTPUT:
1487                            <1>             ;    cf = 0 ->
1488                            <1>             ;      EAX (AL) = device entry/index number
1489                            <1>             ;    cf = 1 -> device not found (installed)
1490                            <1>             ;              or invalid device name
1491                            <1>             ;              (AL=0)
1492                            <1>             ;    device_name = device name address (asciiz)
1493                            <1>             ;
1494                            <1>             ; Modified registers: EAX, EBX, ESI, EDI
1495                            <1>
1496 000132E9 BF[7D960100]      <1>             mov    edi, device_name
1497 000132EE E805010000        <1>             call   lodsb_capitalize
1498 000132F3 88C4              <1>             mov    ah, al
1499 000132F5 3C2F              <1>             cmp    al, '/'
1500 000132F7 750E              <1>             jne    short gdn_1
1501 000132F9 BF[7D960100]      <1>             mov    edi, device_name
1502 000132FE E8F5000000        <1>             call   lodsb_capitalize
1503                            <1> gdn_0:
1504 00013303 20C0              <1>             and    al, al ; 0 ?
1505 00013305 7420              <1>             jz     short gdn_err ; null name after '/'
1506                            <1> gdn_1:
1507 00013307 3C44              <1>             cmp    al, 'D'
1508 00013309 7517              <1>             jne    short gdn_2
1509 0001330B E8E8000000        <1>             call   lodsb_capitalize
1510 00013310 3C45              <1>             cmp    al, 'E'
1511 00013312 750E              <1>             jne    short gdn_2
1512 00013314 E8DF000000        <1>             call   lodsb_capitalize
1513 00013319 3C56              <1>             cmp    al, 'V'
1514 0001331B 7505              <1>             jne    short gdn_2
1515 0001331D AC                <1>             lodsb
1516 0001331E 3C2F              <1>             cmp    al, '/'
1517 00013320 740D              <1>             je     short gdn_4
1518                            <1> gdn_2:
1519 00013322 80FC2F            <1>             cmp    ah, '/'
1520 00013325 750F              <1>             jne    short gdn_5
1521                            <1> gdn_err:
1522                            <1>             ; invalid device name or device not found
1523 00013327 31C0              <1>             xor    eax, eax ; 0
1524 00013329 F9                <1>             stc
1525 0001332A C3                <1>             retn
1526                            <1> gdn_3:
1527 0001332B 3C2F              <1>             cmp    al, '/'
1528 0001332D 7507              <1>             jne    short gdn_5
1529                            <1> gdn_4:
1530 0001332F BF[7D960100]      <1>             mov    edi, device_name
1531 00013334 EB04              <1>             jmp    short gdn_6
1532                            <1> gdn_5:
1533 00013336 3C00              <1>             cmp    al, 0
1534 00013338 7419              <1>             je     short gdn_7
1535                            <1> gdn_6:
1536 0001333A E8B9000000        <1>             call lodsb_capitalize
1537 0001333F 81FF[85960100]    <1>             cmp    edi, device_name + 8
1538 00013345 72E4              <1>             jb     short gdn_3
1539 00013347 3C00              <1>             cmp    al, 0
1540 00013349 75DC              <1>             jne    short gdn_err
1541 0001334B 81FF[7E960100]    <1>             cmp    edi, device_name + 1
1542 00013351 76D4              <1>             jna    short gdn_err ; null name after '/'
1543                            <1> gdn_7:
1544 00013353 AA                <1>             stosb
1545                            <1>             ; zero padding ("NAME",0,0,0,0)
1546 00013354 81FF[85960100]    <1>             cmp    edi, device_name + 8
1547 0001335A 72F7              <1>             jb     short gdn_7
1548                            <1> gdn_8:
1549                            <1>             ; search for kernel device names
1550 0001335C BE[7D960100]      <1>             mov    esi, device_name
1551 00013361 BF[6C470100]      <1>             mov    edi, KDEV_NAME
1552 00013366 31C0              <1>             xor    eax, eax
1553                            <1> gdn_9:
1554 00013368 A7                <1>             cmpsd
1555 00013369 7505              <1>             jne    short gdn_10
1556 0001336B A7                <1>             cmpsd
1557 0001336C 7503              <1>             jne    short gdn_11
1558 0001336E EB2B              <1>             jmp    short gdn_17 ; match
1559                            <1> gdn_10:
1560 00013370 A7                <1>             cmpsd  ; add esi, 4 & add edi, 4
1561                            <1> gdn_11:
1562 00013371 BE[7D960100]      <1>             mov    esi, device_name
1563 00013376 FEC0              <1>             inc    al
```

```
1564 00013378 3C16                <1>             cmp    al, NumOfKernelDevNames
1565 0001337A 72EC                <1>             jb     short gdn_9
1566                              <1> gdn_12:
1567                              <1>             ; search for installable device names
1568                              <1>             ; esi = offset device_name
1569 0001337C BF[A8960100]        <1>             mov    edi, IDEV_NAME
1570 00013381 28C0                <1>             sub    al, al ; 0
1571                              <1> gdn_13:
1572 00013383 A7                  <1>             cmpsd
1573 00013384 7505                <1>             jne    short gdn_14
1574 00013386 A7                  <1>             cmpsd
1575 00013387 7503                <1>             jne    short gdn_15
1576 00013389 EB3F                <1>             jmp    short gdn_19 ; match
1577                              <1> gdn_14:
1578 0001338B A7                  <1>             cmpsd  ; add esi, 4 & add edi, 4
1579                              <1> gdn_15:
1580 0001338C BE[7D960100]        <1>             mov    esi, device_name
1581 00013391 FEC0                <1>             inc    al
1582 00013393 3C08                <1>             cmp    al, NumOfInstallableDevices
1583 00013395 72EC                <1>             jb     short gdn_13
1584                              <1>
1585                              <1> gdn_16:     ; error: invalid device name (not found) !
1586 00013397 30C0                <1>             xor    al, al
1587 00013399 F9                  <1>             stc
1588 0001339A C3                  <1>             retn
1589                              <1>
1590                              <1> gdn_17:           ; name match (with one of kernel device names)
1591                              <1>             ;
1592                              <1>             ; convert KDEV_NAME index to
1593                              <1>             ; KDEV_NUMBER index
1594                              <1>             ; (different names are used for same devices)
1595                              <1>             ; (example: "COM1" & "TTY8" = device number 18)
1596 0001339B 89C3                <1>             mov    ebx, eax ; < 256
1597 0001339D 8A83[1C480100]      <1>             mov    al, [KDEV_NUMBER+ebx]
1598                              <1>
1599                              <1>             ; check if empty dev entry in the list
1600 000133A3 80B8[2C980100]00    <1>             cmp    byte [DEV_OPENMODE+eax], 0
1601 000133AA 771B                <1>             ja     short gdn_18 ; it must be already set
1602                              <1>
1603                              <1>             ; (re)set device name and access flags
1604                              <1>             ; (remain open work will be easy after that)
1605                              <1>             ; (NOTE: here, data will be copied to bss section)
1606 000133AC 88C3                <1>             mov    bl, al
1607 000133AE 83EF08              <1>             sub    edi, 8 ; kernel device name address (data)
1608 000133B1 66C1E302            <1>             shl    bx, 2
1609 000133B5 89BB[4A980100]      <1>             mov    [DEV_NAME_PTR+ebx], edi ; (all) device names
1610 000133BB 8A98[72490100]      <1>             mov    bl, [KDEV_ACCESS+eax] ; kernel dev list (data)
1611 000133C1 8898[78970100]      <1>             mov    [DEV_ACCESS+eax], bl ; (all) device list (bss)
1612                              <1> gdn_18:
1613 000133C7 FEC0                <1>             inc    al ; 1 to NumOfKernelDevNames (<=7Fh)
1614                              <1>             ; eax = device index/entry number
1615 000133C9 C3                  <1>             retn
1616                              <1>
1617                              <1> gdn_19:           ; name match (with one of installable device names)
1618                              <1>             ;
1619                              <1>             ; al = 0 to NumOfInstallableDevices - 1 (<=7Fh)
1620                              <1>
1621 000133CA 89C3                <1>             mov    ebx, eax
1622 000133CC 80C316              <1>             add    bl, NumOfKernelDevices    ; < NUMOFDEVICES
1623                              <1>
1624                              <1>             ; check if empty dev entry in the list
1625 000133CF 80BB[2C980100]00    <1>             cmp    byte [DEV_OPENMODE+ebx], 0
1626 000133D6 771D                <1>             ja     short gdn_20 ; it must be already set
1627                              <1>
1628                              <1>             ; (re)set device name and access flags
1629                              <1>             ; (remain open work will be easy after that)
1630 000133D8 83EF08              <1>             sub    edi, 8 ; installable device name address
1631 000133DB 66C1E302            <1>             shl    bx, 2 ;*4
1632 000133DF 89BB[4A980100]      <1>             mov    [DEV_NAME_PTR+ebx], edi ; (all) device names
1633 000133E5 66C1EB02            <1>             shr    bx, 2
1634 000133E9 8A80[F0960100]      <1>             mov    al, [IDEV_FLAGS+eax] ; installable dev list
1635 000133EF 8883[78970100]      <1>             mov    [DEV_ACCESS+ebx], al ; (all) device list
1636                              <1> gdn_20:
1637 000133F5 88D8                <1>             mov    al, bl
1638                              <1>             ; eax = device index/entry number ; < NUMOFDEVICES
1639 000133F7 C3                  <1>             retn
1640                              <1>
1641                              <1> lodsb_capitalize:
1642                              <1>       ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
1643                              <1>       ; INPUT -> [esi] = character
1644                              <1>       ;          edi = destination
1645                              <1>       ; OUTPUT -> AL contains capitalized character
1646                              <1>       ;         esi = esi+1
1647                              <1>       ;         edi = edi+1
1648                              <1>       ;
1649 000133F8 AC                  <1>       lodsb
1650 000133F9 3C61                <1>       cmp al, 61h
1651 000133FB 7206                <1>       jb  short lodsb_cap_retn
1652 000133FD 3C7A                <1>       cmp al, 7Ah
1653 000133FF 7702                <1>       ja  short lodsb_cap_retn
1654 00013401 24DF                <1>       and al, 0DFh
1655                              <1> lodsb_cap_retn:
1656 00013403 AA                  <1>       stosb
1657 00013404 C3                  <1>       retn
1658                              <1>
1659                              <1> device_open:
1660                              <1>       ; 08/10/2016 - TRDOS 386 (TRDOS v2.0)
1661                              <1>       ; Complete device opening work for sysopen (device)
1662                              <1>       ;
1663                              <1>       ; INPUT ->
1664                              <1>       ;     EAX = Device Number (AL)
1665                              <1>       ;       CL = Open mode (1 = read, 2 = write)
1666                              <1>       ;      CH = Device access byte (bit 0 = 0)
1667                              <1>       ; OUTPUT ->
1668                              <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>
00013405 89C3                   <1>       mov   ebx, eax
00013407 66C1E302               <1>       shl   bx, 2 ; *4
                                <1>
0001340B F6C580                 <1>       test  ch, 80h ; bit 7, installable device driver flag
0001340E 7406                   <1>       jz    short d_open_2 ; Kernel device
                                <1>       ; installable device
                                <1> d_open_1:
00013410 FFA3[F4960100]         <1>         jmp dword [ebx+IDEV_OADDR-4]
                                <1> d_open_2:
00013416 FFA3[2E480100]         <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>
0001341C 89C3                   <1>       mov   ebx, eax
0001341E 66C1E302               <1>       shl   bx, 2 ; *4
                                <1>
00013422 F6C580                 <1>       test  ch, 80h ; bit 7, installable device driver flag
00013425 7406                   <1>       jz    short d_close_2 ; Kernel device
                                <1>       ; installable device
                                <1> d_close_1:
00013427 FFA3[14970100]         <1>         jmp dword [ebx+IDEV_CADDR-4]
                                <1> d_close_2:
0001342D FFA3[7E480100]         <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)
00013433 C3                     <1>       retn
                                <1>
                                <1> wnull:
                                <1>       ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; write null (write to null device)
00013434 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>       ;
00013435 53                     <1>       push  ebx
00013436 0FB6D8                 <1>       movzx ebx, al
00013439 C0E302                 <1>       shl   bl, 2 ; * 4
0001343C 8B9B[B29B0100]         <1>       mov   ebx, [ebx+DEV_INT_HNDLR]
00013442 21DB                   <1>       and   ebx, ebx
00013444 7404                   <1>         jz  short dIRQ_s_retn
00013446 50                     <1>       push  eax
                                <1>
00013447 FFD3                   <1>       call  ebx
                                <1>
00013449 58                     <1>       pop   eax
                                <1> dIRQ_s_retn:
0001344A 5B                     <1>       pop   ebx
0001344B 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
0001344C 0FB6F0                 <1>       movzx esi, al
0001344F 66C1E602               <1>       shl   si, 2 ; * 4
00013453 899E[B29B0100]         <1>       mov   [esi+DEV_INT_HNDLR], ebx
                                <1>       ;pop   esi
00013459 C3                     <1>       retn
                                <1>
                                <1>
                                <1> sysaudio: ; AUDIO FUNCTIONS
                                <1>       ; 12/02/2021 (TRDOS 386 v2.0.3)
                                <1>       ; 28/07/2020
```

```
1774                    <1>     ; 27/07/2020
1775                    <1>     ; 10/10/2017
1776                    <1>     ; 22/06/2017
1777                    <1>     ; 28/05/2017, 04/06/2017, 05/06/2017, 10/06/2017
1778                    <1>     ; 01/05/2017, 12/05/2017, 15/05/2017, 20/05/2017
1779                    <1>     ; 21/04/2017, 22/04/2017, 23/04/2017, 24/04/2017
1780                    <1>     ; 10/04/2017, 13/04/2017, 14/04/2017, 16/04/2017
1781                    <1>     ; 03/04/2017 (VIA VT8237R)
1782                    <1>     ; 01/04/2016 (trdosk6.s -> tdosk8.s)
1783                    <1>     ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
1784                    <1>     ;
1785                    <1>     ; Inputs:
1786                    <1>     ;
1787                    <1>     ;     BH = 0 -> Beep (PC Speaker)
1788                    <1>     ;          BL = Duration Counter (1 for 1/64 second)
1789                    <1>     ;          CX = Frequency Divisor (1193180/Frequency)
1790                    <1>     ;             (1331 for 886 Hz)
1791                    <1>     ;
1792                    <1>     ;     01/04/2017
1793                    <1>     ;
1794                    <1>     ;     BH = 1 -> DETECT (& ENABLE) AUDIO DEVICE
1795                    <1>     ;          BL = 0 : PC SPEAKER
1796                    <1>     ;               1 : SOUND BLASTER 16
1797                    <1>     ;               2 : INTEL AC'97
1798                    <1>     ;               3 : VIA VT8237R (VT8233)
1799                    <1>     ;               4 : INTEL HDA
1800                    <1>     ;             5-FEh : unknown/invalid
1801                    <1>     ;               ; 04/06/2017
1802                    <1>     ;              FFh : Get current audio device id
1803                    <1>     ;
1804                    <1>     ;     BH = 2 -> ALLOCATE AUDIO BUFFER (for user)
1805                    <1>     ;            ECX = Audio Buffer Size (must be equal to
1806                    <1>     ;                  the half of DMA buffer size)
1807                    <1>     ;            EDX = Virtual Address of the buffer
1808                    <1>     ;                  (This is not DMA buffer!)
1809                    <1>     ;
1810                    <1>     ;     BH = 3 -> INITIALIZE AUDIO DEVICE
1811                    <1>     ;          BL = 0,2 -> for Signal Response Byte
1812                    <1>     ;            CL = Signal Response Byte Value (fixed)
1813                    <1>     ;                        if BL = 0
1814                    <1>     ;                 auto increment of S.R.B. value
1815                    <1>     ;                        if BL = 2
1816                    <1>     ;              EDX = Signal Response (Return) Byte Address
1817                    <1>     ;
1818                    <1>     ;          BL = 1 for CallBack Method
1819                    <1>     ;            EDX = CallBack Service Address (Virtual)
1820                    <1>     ;
1821                    <1>     ;          BL > 2 -> invalid function
1822                    <1>     ;
1823                    <1>     ;         (Audio buffer must be allocated before
1824                    <1>     ;          initialization.)
1825                    <1>     ;
1826                    <1>     ;     BH = 4 -> START TO PLAY
1827                    <1>     ;          BL = Mode
1828                    <1>     ;              Bit 0 = mono/stereo (1 = stereo)
1829                    <1>     ;              Bit 1 = 8 bit / 16 bit (1 = 16 bit)
1830                    <1>     ;          CX = Sampling Rate (Hz)
1831                    <1>     ;
1832                    <1>     ;     BH = 5 -> PAUSE
1833                    <1>     ;          BL = Any
1834                    <1>     ;
1835                    <1>     ;     BH = 6 -> CONTINUE TO PLAY
1836                    <1>     ;          BL = Any
1837                    <1>     ;
1838                    <1>     ;     BH = 7 -> STOP
1839                    <1>     ;          BL = Any
1840                    <1>     ;
1841                    <1>     ;     BH = 8 -> RESET
1842                    <1>     ;          BL = Any
1843                    <1>     ;
1844                    <1>     ;     BH = 9 -> CANCEL (CALLBACK or S.R.B. SERVICE)
1845                    <1>     ;          BL = Any
1846                    <1>     ;
1847                    <1>     ;     BH = 10 -> DEALLOCATE AUDIO BUFFER (for user)
1848                    <1>     ;          BL = Any
1849                    <1>     ;
1850                    <1>     ;     BH = 11 -> SET VOLUME LEVEL
1851                    <1>     ;          BL: (Bit 0 to 6)
1852                    <1>     ;              0 = Master (Playback, Lineout) volume
1853                    <1>     ;          CL = Left Channel Volume
1854                    <1>     ;          CH = Right Channel Volume
1855                    <1>     ;
1856                    <1>     ;          Note: If BL >= 80h (Bit 7 of BL is set),
1857                    <1>     ;          volume level will be set for next playing
1858                    <1>     ;          (actual volume level will not be changed
1859                    <1>     ;          immediately)
1860                    <1>     ;
1861                    <1>     ;     BH = 12 -> DISABLE AUDIO DEVICE
1862                    <1>     ;          (reset audio device and unlink dma buffer)
1863                    <1>     ;          BL = Any
1864                    <1>     ;
1865                    <1>     ;     12/05/2017
1866                    <1>     ;     BH = 13 -> MAP DMA BUFFER TO USER
1867                    <1>     ;         (for direct access to system's dma buffer)
1868                    <1>     ;
1869                    <1>     ;          ECX = map size in bytes
1870                    <1>     ;              (will be rounded up to page borders)
1871                    <1>     ;          EDX = Virtual Address of the buffer
1872                    <1>     ;              (Will be rounded up to page borders)
1873                    <1>     ;
1874                    <1>     ;     05/06/2017
1875                    <1>     ;     04/06/2017
1876                    <1>     ;     BH = 14 -> GET AUDIO DEVICE INFO
1877                    <1>     ;          BL: 0 = Audio Controller Info
1878                    <1>     ;            > 0 = Invalid for now!
```

```
1879                    <1>     ;
1880                    <1>     ;       22/06/2017
1881                    <1>     ;       BH = 15 -> GET CURRENT SOUND DATA (for graphics)
1882                    <1>     ;            BL: 0 -> PCM OUT data
1883                    <1>     ;              > 0 -> Invalid for now!
1884                    <1>     ;            ECX = 0 -> Get DMA Buffer Pointer
1885                    <1>     ;               EDX = Not Used
1886                    <1>     ;            ECX > 0 -> Byte count for buffer (EDX)
1887                    <1>     ;                EDX = Buffer Address (Virtual)
1888                    <1>     ;
1889                    <1>     ;       10/10/2017
1890                    <1>     ;       BH = 16 -> UPDATE DMA BUFFER DATA
1891                    <1>     ;                  (by using the Audio Buffer content)
1892                    <1>     ;            BL = 0 : Update dma half buffer in sequence
1893                    <1>     ;                     (automatic destination)
1894                    <1>     ;                   1 : Update 1st half of the dma buffer
1895                    <1>     ;                   2 : Update 2nd half of the dma buffer
1896                    <1>     ;                   3-FEh: Invalid!
1897                    <1>     ;                   FFh = Get current flag value
1898                    <1>     ;                        (Half buffer number -1)
1899                    <1>     ;
1900                    <1>     ;
1901                    <1>     ; Outputs:
1902                    <1>     ;
1903                    <1>     ;       For BH = 0 -> Beep
1904                    <1>     ;           None
1905                    <1>     ;
1906                    <1>     ;       01/04/2017
1907                    <1>     ;
1908                    <1>     ;       For BH = 1 -> DETECT (& ENABLE) AUDIO DEVICE
1909                    <1>     ;           AH = 0 : PC SPEAKER
1910                    <1>     ;                1 : SOUND BLASTER 16
1911                    <1>     ;                2 : INTEL AC'97
1912                    <1>     ;                3 : VIA VT8237R (VT8233)
1913                    <1>     ;                4 : INTEL HDA
1914                    <1>     ;              5-FFh : unknown/invalid
1915                    <1>     ;           AL = mode status
1916                    <1>     ;               bit 0 = mono /stereo (1 = stereo)
1917                    <1>     ;               bit 1 = 8 bit / 16 bit ( 1 = 16 bit)
1918                    <1>     ;           04/06/2017
1919                    <1>     ;           EBX = PCI DEVICE/VENDOR ID (if >0)
1920                    <1>     ;               (BX = VENDOR ID)
1921                    <1>     ;           (if CF = 1 -> Error code in EAX)
1922                    <1>     ;
1923                    <1>     ;       For BH = 2 -> ALLOCATE AUDIO BUFFER (for user)
1924                    <1>     ;           EAX = Physical Address of the buffer
1925                    <1>     ;           (if CF = 1 -> Error code in EAX)
1926                    <1>     ;
1927                    <1>     ;       For BH = 3 -> INITIALIZE AUDIO DEVICE
1928                    <1>     ;           (if CF = 1 -> Error code in EAX)
1929                    <1>     ;
1930                    <1>     ;       For BH = 4 -> START TO PLAY
1931                    <1>     ;           none (if CF = 1 -> Error code in EAX)
1932                    <1>     ;
1933                    <1>     ;       For BH = 5 -> PAUSE
1934                    <1>     ;           none (if CF = 1 -> Error code in EAX)
1935                    <1>     ;
1936                    <1>     ;       For BH = 6 -> CONTINUE TO PLAY
1937                    <1>     ;           none (if CF = 1 -> Error code in EAX)
1938                    <1>     ;
1939                    <1>     ;       For BH = 7 -> STOP
1940                    <1>     ;           none (if CF = 1 -> Error code in EAX)
1941                    <1>     ;
1942                    <1>     ;       For BH = 8 -> RESET
1943                    <1>     ;           none (if CF = 1 -> Error code in EAX)
1944                    <1>     ;
1945                    <1>     ;       For BH = 9 -> CANCEL (CALLBACK or S.R.B. SERVICE)
1946                    <1>     ;           none (if CF = 1 -> Error code in EAX)
1947                    <1>     ;
1948                    <1>     ;       For BH = 10 -> DEALLOCATE AUDIO BUFFER (for user)
1949                    <1>     ;           none (if CF = 1 -> Error code in EAX)
1950                    <1>     ;
1951                    <1>     ;       For BH = 11 -> SET VOLUME LEVEL
1952                    <1>     ;           none (if CF = 1 -> Error code in EAX)
1953                    <1>     ;
1954                    <1>     ;       For BH = 12 -> DISABLE AUDIO DEVICE
1955                    <1>     ;           none (if CF = 1 -> Error code in EAX)
1956                    <1>     ;
1957                    <1>     ;       12/05/2017
1958                    <1>     ;       For BH = 13 -> MAP DMA BUFFER TO USER
1959                    <1>     ;           EAX = Physical Address of the buffer
1960                    <1>     ;           (if CF = 1 -> Error code in EAX)
1961                    <1>     ;
1962                    <1>     ;       04/06/2017
1963                    <1>     ;       For BH = 14 -> GET AUDIO DEVICE INFO
1964                    <1>     ;       (for BL = 0) ; 05/06/2017
1965                    <1>     ;           EAX = IRQ Number in AL
1966                    <1>     ;                Audio Device Number in AH
1967                    <1>     ;           EBX = DEV/VENDOR ID
1968                    <1>     ;               (DDDDDDDDDDDDDDDDVVVVVVVVVVVVVVVV)
1969                    <1>     ;           ECX = BUS/DEV/FN
1970                    <1>     ;               (00000000BBBBBBBBDDDDDFFF00000000)
1971                    <1>     ;           EDX = NABMBAR/NAMBAR (for AC97)
1972                    <1>     ;                (Low word, DX = NAMBAR address)
1973                    <1>     ;           EDX = Base IO Addr (DX) for SB16 & VT8233
1974                    <1>     ;           (if CF = 1 -> Error code in EAX)
1975                    <1>     ;                         (ERR_DEV_NOT_RDY = 15)
1976                    <1>     ;
1977                    <1>     ;       22/06/2017
1978                    <1>     ;       For BH = 15 -> GET CURRENT SOUND DATA
1979                    <1>     ;                      (for graphics)
1980                    <1>     ;       (for BL = 0)
1981                    <1>     ;        If ECX input is 0
1982                    <1>     ;           EAX = DMA Buffer Current Position (Offset)
1983                    <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>
0001345A 80FF11                 <1>       cmp    bh, AUDIO1L/4
0001345D 0F8399A4FFFF           <1>       jnb    sysret
                                <1>
00013463 C0E702                 <1>       shl    bh, 2 ; *4
00013466 0FB6F7                 <1>       movzx  esi, bh
                                <1>
                                <1>       ; 22/04/2017
00013469 31C0                   <1>       xor    eax, eax
0001346B A3[64030300]           <1>       mov    [u.r0], eax  ; 0
                                <1>
00013470 FF96[7B340100]         <1>       call   dword [esi+AUDIO1]
                                <1>       ;jc    error
00013476 E981A4FFFF             <1>       jmp    sysret
                                <1>
0001347B [E1230000]             <1> AUDIO1:      dd     _beep ; 12/02/2021
                                <1>       ;dd    beep ; FUNCTION = 0 (bl = Duration Counter
                                <1>                   ;                    cx = Frequency Divisor
0001347F [BF340100]             <1>       dd     soundc_detect
00013483 [5B350100]             <1>       dd     sound_alloc
00013487 [19360100]             <1>       dd     soundc_init
0001348B [D1370100]             <1>       dd     sound_play
0001348F [6D380100]             <1>       dd     sound_pause
00013493 [97380100]             <1>       dd     sound_continue
00013497 [C1380100]             <1>       dd     sound_stop
0001349B [EA380100]             <1>       dd     soundc_reset
0001349F [1B390100]             <1>       dd     soundc_cancel
000134A3 [41390100]             <1>       dd     sound_dalloc
000134A7 [6C390100]             <1>       dd     sound_volume
000134AB [BE390100]             <1>       dd     soundc_disable
000134AF [303A0100]             <1>       dd     sound_dma_map
000134B3 [9F3A0100]             <1>       dd     soundc_info
000134B7 [FE3A0100]             <1>       dd     sound_data
000134BB [AB3B0100]             <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
000134BF 8A25[419C0100]         <1>       mov    ah, [audio_device]
000134C5 80FBFF                 <1>       cmp    bl, 0FFh ; get current audio device id
000134C8 7408                   <1>       je     short sysaudio0
                                <1>
000134CA 20E4                   <1>       and    ah, ah
000134CC 741E                   <1>       jz     short soundc_get_dev
                                <1>
000134CE 38DC                   <1>       cmp    ah, bl
000134D0 7567                   <1>       jne    short soundc_dev_err
                                <1>
                                <1> sysaudio0:
000134D2 A0[429C0100]           <1>       mov    al, [audio_mode]
                                <1> sysaudio1:
000134D7 A3[64030300]           <1>       mov    [u.r0], eax
000134DC 8B1D[4C9C0100]         <1>       mov    ebx, [audio_vendor] ; (DEVICE/VENDOR ID)
000134E2 8B2D[60030300]         <1>       mov    ebp, [u.usp]
000134E8 895D10                 <1>       mov    [ebp+16], ebx  ; ebx
000134EB C3                     <1>       retn
                                <1>
                                <1> soundc_get_dev:
                                <1>       ; 28/05/2017
                                <1>       ; 03/04/2017, 24/04/2017
000134EC C605[409C0100]00       <1>       mov    byte [audio_pci], 0
000134F3 80FB03                 <1>       cmp    bl, 3 ; VIA VT8233 (VT8237R) Audio Controller & AC97 Codec
                                <1>       ;jne   short soundc_get_dev_sb
                                <1>       ; 28/05/2017
000134F6 7220                   <1>       jb     short soundc_get_dev_sb
000134F8 773F                   <1>       ja     short soundc_dev_err  ; temporary (28/05/2017)
                                <1>       ;
000134FA E86E180000             <1>       call   DetectVT8233
000134FF 7238                   <1>       jc     short soundc_dev_err
                                <1>       ; eax = 0
                                <1>
                                <1>       ;mov   ebx, [audio_vendor]
                                <1>       ; ebx = DEVICE/VENDOR ID
                                <1>       ;      DDDDDDDDDDDDDDDDVVVVVVVVVVVVVVVV
                                <1>
00013501 B003                   <1>       mov    al, 3  ; VIA VT8237R (VT3233) Audio Controller
00013503 88C4                   <1>       mov    ah, al
                                <1>
                                <1> soundc_get_pci_dev_ok: ; 28/05/2017
00013505 FE05[409C0100]         <1>       inc    byte [audio_pci] ; = 1
                                <1> soundc_get_dev_ok:
                                <1>
                                <1> soundc_get_dev_sb16_ok:
0001350B A2[419C0100]           <1>       mov    [audio_device], al
```

```
2089 00013510 8825[429C0100]      <1>       mov   [audio_mode], ah ; stereo (bit0), 16 bit (bit1) capability
2090 00013516 EBBF                <1>       jmp   short sysaudio1
2091                              <1>
2092                              <1> soundc_get_dev_sb:
2093                              <1>       ; 24/04/2017
2094 00013518 80FB01              <1>       cmp   bl, 1 ; Sound Blaster 16
2095 0001351B 750E                <1>       jne   short soundc_get_dev_ich ; 28/05/2017
2096                              <1>       ;
2097 0001351D E86F1D0000          <1>       call  DetectSB
2098 00013522 7215                <1>       jc    short soundc_dev_err
2099 00013524 B801030000          <1>       mov   eax, 0301h ; Sound Blaster 16
2100 00013529 EBE0                <1>       jmp   short soundc_get_dev_sb16_ok
2101                              <1>
2102                              <1> soundc_get_dev_ich:
2103                              <1>       ; 28/05/2017
2104                              <1>       ;cmp  bl, 2 ; Intel AC'97 Audio Controller (ICH)
2105                              <1>       ;jne  short soundc_dev_err ; Temporary  (28/05/2017)
2106                              <1>       ;                          ; (Here will be modified just after
2107                              <1>       ;                          ; new sound card code will be ready!)
2108 0001352B E830180000          <1>       call  DetectICH
2109 00013530 7207                <1>       jc    short soundc_dev_err
2110                              <1>       ;
2111 00013532 B802030000          <1>       mov   eax, 0302h ; AC'97 (ICH)
2112 00013537 EBCC                <1>       jmp   short soundc_get_pci_dev_ok
2113                              <1>
2114                              <1> soundc_dev_err:
2115 00013539 B80F000000          <1>       mov   eax, ERR_DEV_NOT_RDY ; Device not ready !
2116 0001353E EB0C                <1>       jmp   short sysaudio_err
2117                              <1>
2118                              <1> sound_buff_error:
2119 00013540 B82E000000          <1>       mov   eax, ERR_BUFFER ; Buffer error !
2120 00013545 EB05                <1>       jmp   short sysaudio_err
2121                              <1>
2122                              <1> soundc_respond_err:
2123                              <1>       ; ERR_TIME_OUT ; 'time out !' error
2124 00013547 B819000000          <1>       mov   eax, ERR_DEV_NOT_RESP ; 'device not responding !' error
2125                              <1> sysaudio_err:
2126 0001354C A3[64030300]        <1>       mov   [u.r0], eax
2127 00013551 A3[C8030300]        <1>       mov   [u.error], eax
2128 00013556 E981A3FFFF          <1>       jmp   error
2129                              <1>
2130                              <1> sound_alloc:
2131                              <1>       ; FUNCTION =  2
2132                              <1>       ; ecx = audio buffer size (in bytes)
2133                              <1>       ; edx = audio buffer address (virtual)
2134                              <1>       ; 27/07/2020
2135                              <1>       ; 28/05/2017
2136                              <1>       ; 01/05/2017, 15/05/2017
2137                              <1>       ; 21/04/2017, 24/04/2017
2138 0001355B 803D[409C0100]00     <1>       cmp   byte [audio_pci], 0
2139 00013562 7708                <1>       ja    short snd_alloc_0
2140                              <1>       ; Max. 64KB DMA buffer !!!
2141 00013564 81F900800000        <1>       cmp   ecx, 32768
2142 0001356A 77D4                <1>       ja    short sound_buff_error
2143                              <1> snd_alloc_0:
2144                              <1>       ; 15/05/2017
2145 0001356C 81F900100000        <1>       cmp   ecx, 4096 ; PAGE_SIZE
2146 00013572 72CC                <1>       jb    short sound_buff_error
2147                              <1>       ;
2148 00013574 A1[549C0100]        <1>       mov   eax, [audio_buffer] ; audio buffer address (current)
2149 00013579 09C0                <1>       or    eax, eax
2150 0001357B 7445                <1>       jz    short snd_alloc_2
2151                              <1>       ; audio buffer exists !
2152 0001357D 8A1D[B3030300]      <1>       mov   bl, [u.uno]
2153 00013583 3A1D[699C0100]      <1>       cmp   bl, [audio_user]
2154 00013589 0F85FC000000        <1>       jne   sndc_owner_error ; not owner !
2155 0001358F 39D0                <1>       cmp   eax, edx ; same virtual buffer address ?
2156 00013591 7508                <1>       jne   short snd_alloc_1
2157 00013593 3B0D[5C9C0100]      <1>       cmp   ecx, [audio_buff_size]
2158 00013599 746C                <1>       je    short snd_alloc_3 ; Nothing to do !
2159                              <1>                               ; Buffer has been set already!
2160                              <1> snd_alloc_1:
2161 0001359B 51                  <1>       push  ecx
2162 0001359C 52                  <1>       push  edx
2163 0001359D 89C3                <1>       mov   ebx, eax ; audio buffer address (current)
2164 0001359F 8B0D[5C9C0100]      <1>       mov   ecx, [audio_buff_size]
2165 000135A5 E82E32FFFF          <1>       call  deallocate_user_pages
2166 000135AA 5A                  <1>       pop   edx
2167 000135AB 59                  <1>       pop   ecx
2168 000135AC 31C0                <1>       xor   eax, eax ; 0
2169 000135AE A3[549C0100]        <1>       mov   [audio_buffer], eax  ; 0
2170 000135B3 A3[589C0100]        <1>       mov   [audio_p_buffer], eax  ; 0
2171 000135B8 A3[5C9C0100]        <1>       mov   [audio_buff_size], eax
2172 000135BD A2[699C0100]        <1>       mov   [audio_user], al ; 0
2173                              <1> snd_alloc_2:
2174 000135C2 89D3                <1>       mov   ebx, edx
2175                              <1>       ; 01/05/2017
2176 000135C4 BA00F0FFFF          <1>       mov   edx, ~PAGE_OFF ; truncating page offsets
2177                              <1>                            ; for aligning to page borders
2178                              <1>       ;and  eax, edx
2179 000135C9 21D3                <1>       and   ebx, edx
2180 000135CB 21D1                <1>       and   ecx, edx
2181                              <1>       ; 15/05/2017
2182                              <1>       ; EAX = Beginning address (physical)
2183                              <1>       ; EAX = 0 -> Allocate mem block from the 1st proper aperture
2184                              <1>       ; ECX = Number of bytes to be allocated
2185 000135CD E8AB2EFFFF          <1>       call  allocate_memory_block
2186 000135D2 0F8268FFFFFF        <1>       jc    sound_buff_error
2187                              <1>       ; EAX = Physical address of the allocated memory block
2188                              <1>       ; ECX = Allocated bytes (as truncated to page border)
2189                              <1>       ; EBX = Virtual address (as truncated to page border)
2190 000135D8 50                  <1>       push  eax
2191 000135D9 53                  <1>       push  ebx
2192 000135DA 51                  <1>       push  ecx
2193 000135DB E8ED32FFFF          <1>       call  allocate_user_pages
```

```
2194 000135E0 59                  <1>       pop    ecx
2195 000135E1 5B                  <1>       pop    ebx
2196 000135E2 58                  <1>       pop    eax
2197 000135E3 722A                <1>       jc     short snd_alloc_4  ; insufficient memory, buff error
2198                              <1>       ; eax = physical address of the user's audio buffer
2199                              <1>       ; ebx = virtual address of the user's audio buffer
2200                              <1>       ; ecx = buffer size (in bytes)
2201 000135E5 A3[589C0100]        <1>       mov    [audio_p_buffer], eax
2202 000135EA 891D[549C0100]      <1>       mov    [audio_buffer], ebx
2203 000135F0 890D[5C9C0100]      <1>       mov    [audio_buff_size], ecx
2204 000135F6 8A15[B3030300]      <1>       mov    dl, [u.uno]
2205 000135FC 8815[699C0100]      <1>       mov    [audio_user], dl
2206 00013602 A3[64030300]        <1>       mov    [u.r0], eax
2207                              <1> snd_alloc_3:
2208                              <1>       ; 27/07/2020
2209 00013607 C605[689C0100]00    <1>       mov    byte [audio_flag], 0 ; clear dma half buffer flag
2210                              <1>       ;
2211 0001360E C3                  <1>       retn
2212                              <1> snd_alloc_4:
2213                              <1>       ; 15/05/2017
2214                              <1>       ; EAX = Beginning address (physical)
2215                              <1>       ; ECX = Number of bytes to be deallocated
2216 0001360F E87630FFFF          <1>       call   deallocate_memory_block
2217 00013614 E927FFFFFF          <1>       jmp    sound_buff_error  ; insufficient memory, buff error
2218                              <1>
2219                              <1> soundc_init:
2220                              <1>       ; FUNCTION = 3
2221                              <1>       ; bl = method (0= s.r.b., 1= callback, 2= auto incr s.r.b.)
2222                              <1>       ; cl = signal response byte (initial or fixed) value
2223                              <1>       ; edx = signal response byte or callback address
2224                              <1>       ; 27/07/2020
2225                              <1>       ; 28/05/2017
2226                              <1>       ; 12/05/2017, 20/05/2017
2227                              <1>       ; 22/04/2017, 23/04/2017, 24/04/2017
2228                              <1>       ; 13/04/2017, 14/04/2017, 16/04/2017, 21/04/2017
2229                              <1>       ; 03/04/2017, 10/04/2017
2230                              <1>
2231 00013619 A0[419C0100]        <1>       mov    al, [audio_device]
2232 0001361E 20C0                <1>       and    al, al
2233 00013620 7549                <1>       jnz    short sndc_init_6
2234                              <1>       ;
2235 00013622 C605[409C0100]00    <1>       mov    byte [audio_pci], 0
2236 00013629 52                  <1>       push   edx
2237 0001362A 53                  <1>       push   ebx
2238 0001362B 51                  <1>       push   ecx
2239 0001362C E8601C0000          <1>       call   DetectSB
2240 00013631 7213                <1>       jc     short sndc_init_8
2241 00013633 66B80103            <1>       mov    ax, 0301h ; Sound Blaster 16
2242 00013637 EB1E                <1>       jmp    short sndc_init_7
2243                              <1>
2244                              <1> sndc_init_11:
2245                              <1>       ; 28/05/2017
2246 00013639 E822170000          <1>       call   DetectICH ; Detect AC'97 (ICH) Audio Controller
2247 0001363E 7217                <1>       jc     short sndc_init_7
2248 00013640 66B80203            <1>       mov    ax, 0302h ; Intel AC'97 Audio Device
2249 00013644 EB0B                <1>       jmp    short sndc_init_12 ; (PCI device)
2250                              <1>
2251                              <1> sndc_init_8:
2252 00013646 E822170000          <1>       call   DetectVT8233
2253                              <1>       ;jc    short sndc_init_7
2254 0001364B 72EC                <1>       jc     sndc_init_11 ; 28/05/2017
2255                              <1>       ; eax = 0
2256 0001364D B003                <1>       mov    al, 3  ; VIA VT8237R (VT3233) Audio Controller
2257 0001364F 88C4                <1>       mov    ah, al
2258                              <1>
2259                              <1> sndc_init_12:
2260 00013651 FE05[409C0100]      <1>       inc    byte [audio_pci] ; = 1
2261                              <1> sndc_init_7:
2262 00013657 59                  <1>       pop    ecx
2263 00013658 5B                  <1>       pop    ebx
2264 00013659 5A                  <1>       pop    edx
2265 0001365A 0F82D9FEFFFF        <1>       jc     soundc_dev_err
2266                              <1>       ;
2267 00013660 A2[419C0100]        <1>       mov    [audio_device], al
2268 00013665 8825[429C0100]      <1>       mov    [audio_mode], ah ; stereo (bit0), 16 bit (bit1) capability
2269                              <1>
2270                              <1> sndc_init_6:
2271 0001366B 833D[549C0100]00    <1>       cmp    dword [audio_buffer], 0
2272 00013672 0F86C8FEFFFF        <1>       jna    sound_buff_error
2273                              <1>
2274 00013678 A0[B3030300]        <1>       mov    al, [u.uno]
2275 0001367D 8A25[699C0100]      <1>       mov    ah, [audio_user]
2276 00013683 08E4                <1>       or     ah, ah
2277 00013685 7418                <1>       jz     short sndc_init0
2278 00013687 38E0                <1>       cmp    al, ah
2279 00013689 7419                <1>       je     short sndc_init1
2280                              <1>
2281                              <1> sndc_owner_error:
2282 0001368B B80B000000          <1>       mov    eax, ERR_NOT_OWNER ; 'permission denied !' error
2283                              <1> sndc_perm_error:
2284 00013690 A3[64030300]        <1>       mov    [u.r0], eax
2285 00013695 A3[C8030300]        <1>       mov    [u.error], eax
2286 0001369A E93DA2FFFF          <1>       jmp    error
2287                              <1> sndc_init0:
2288 0001369F A2[699C0100]        <1>       mov    [audio_user], al
2289                              <1> sndc_init1:
2290 000136A4 8915[6C9C0100]      <1>       mov    [audio_cb_addr], edx
2291 000136AA 881D[6A9C0100]      <1>       mov    [audio_cb_mode], bl
2292 000136B0 880D[6B9C0100]      <1>       mov    [audio_srb], cl
2293                              <1>
2294                              <1>       ; 27/07/2020
2295                              <1>       ;mov   byte [audio_flag], 0  ; clear dma half buffer flag
2296                              <1>
2297                              <1>       ; 24/04/2017
2298 000136B6 803D[419C0100]03    <1>       cmp    byte [audio_device], 3 ; VT8233 (VT8237R)
```

```
2299 000136BD 7438                <1>       je    short sndc_init_9
2300                              <1>       ;ja   short soundc_respond_err ; temporary (28/05/2017)
2301 000136BF 803D[419C0100]01    <1>       cmp   byte [audio_device], 1 ; SB 16
2302 000136C6 7510                <1>       jne   short sndc_init_13
2303 000136C8 BB[B6540100]        <1>       mov   ebx, sb16_int_handler
2304                              <1>       ; Note: 'SbInit' is at 'Start to Play' stage
2305                              <1>       ; 20/05/2017
2306 000136CD 66C705[769C0100]08- <1>       mov   word [audio_master_volume], 0808h ; 2/8
2306 000136D5 08                  <1>
2307 000136D6 EB3F                <1>       jmp   short sndc_init_10
2308                              <1> sndc_init_13:
2309                              <1>       ; 28/05/2017
2310 000136D8 803D[419C0100]02    <1>       cmp   byte [audio_device], 2 ; AC 97 (ICH)
2311 000136DF 0F8562FEFFFF        <1>       jne   soundc_respond_err ; temporary (28/05/2017)
2312                              <1>
2313 000136E5 E8211F0000          <1>       call  ac97_codec_config
2314 000136EA 0F8257FEFFFF        <1>       jc    soundc_respond_err ; codec error !
2315                              <1>
2316 000136F0 BB[F2570100]        <1>       mov   ebx, ac97_int_handler
2317 000136F5 EB20                <1>       jmp   short sndc_init_10
2318                              <1>
2319                              <1> sndc_init_9:
2320                              <1>       ;call  reset_codec
2321                              <1>       ;; eax = 1
2322                              <1>       ;call  codec_io_w16 ; w32
2323 000136F7 E8E8170000          <1>       call  init_codec ; 28/05/2017
2324 000136FC 0F8245FEFFFF        <1>       jc    soundc_respond_err ; codec error !
2325                              <1>
2326 00013702 E80F1A0000          <1>       call  channel_reset
2327                              <1>
2328                              <1>       ; setup the Codec (actually mixer registers)
2329 00013707 E82F190000          <1>         call    codec_config  ; unmute codec, set rates.
2330 0001370C 0F8235FEFFFF        <1>       jc    soundc_respond_err ; codec error !
2331                              <1>
2332 00013712 BB[A8500100]        <1>       mov   ebx, vt8233_int_handler
2333                              <1> sndc_init_10:
2334                              <1>       ; 13/04/2017
2335 00013717 A0[439C0100]        <1>       mov   al, [audio_intr] ; IRQ number
2336 0001371C E82BFDFFFF          <1>       call  set_dev_IRQ_service
2337                              <1>
2338                              <1>       ; SETUP (audio) INTERRUPT CALLBACK SERVICE
2339 00013721 8A1D[439C0100]      <1>       mov   bl, [audio_intr] ; IRQ number
2340 00013727 8A3D[6A9C0100]      <1>       mov   bh, [audio_cb_mode]
2341 0001372D FEC7                <1>       inc   bh  ; 1 = Signal Response Byte method (fixed value)
2342                              <1>                 ; 2 = Callback service method
2343                              <1>                 ; 3 = Auto Increment S.R.B. method
2344 0001372F 8A0D[6B9C0100]      <1>       mov   cl, [audio_srb]
2345 00013735 8B15[6C9C0100]      <1>       mov   edx, [audio_cb_addr]
2346 0001373B A0[699C0100]        <1>       mov   al, [audio_user]
2347                              <1>       ; 14/04/2017
2348 00013740 E8E1040000          <1>       call  set_irq_callback_service
2349                              <1>       ; 16/04/2017
2350 00013745 A3[64030300]        <1>       mov   [u.r0], eax
2351                              <1>       ;jnc  sysret
2352 0001374A 7316                <1>       jnc   short sndc_init2 ; 21/04/2017
2353                              <1>       ;
2354 0001374C A3[C8030300]        <1>       mov   dword [u.error], eax
2355                              <1>
2356 00013751 A0[439C0100]        <1>       mov   al, [audio_intr] ; IRQ number
2357 00013756 31DB                <1>       xor   ebx, ebx ; reset IRQ handler address
2358 00013758 E8EFFCFFFF          <1>       call  set_dev_IRQ_service
2359                              <1>
2360 0001375D E97AA1FFFF          <1>       jmp   error
2361                              <1>
2362                              <1> sndc_init2:
2363                              <1>       ; 21/04/2017
2364 00013762 8B0D[5C9C0100]      <1>       mov   ecx, [audio_buff_size] ; audio buffer size
2365 00013768 D1E1                <1>       shl   ecx, 1 ; *2
2366 0001376A A1[609C0100]        <1>       mov   eax, [audio_dma_buff]
2367 0001376F 21C0                <1>       and   eax, eax
2368 00013771 7415                <1>       jz    short sndc_init3
2369                              <1>
2370 00013773 8B15[649C0100]      <1>       mov   edx, [audio_dmabuff_size] ; dma buffer size
2371 00013779 39D1                <1>       cmp   ecx, edx
2372 0001377B 744D                <1>       je    short sndc_init5
2373                              <1>
2374 0001377D 87CA                <1>       xchg  ecx, edx
2375 0001377F E8062FFFFF          <1>       call  deallocate_memory_block
2376 00013784 87D1                <1>       xchg  edx, ecx
2377 00013786 31C0                <1>       xor   eax, eax
2378                              <1> sndc_init3:
2379                              <1>       ; 12/05/2017
2380 00013788 803D[419C0100]01    <1>       cmp   byte [audio_device], 1 ; SB 16
2381 0001378F 7515                <1>       jne   short sndc_init4
2382 00013791 C705[609C0100]-     <1>       mov   dword [audio_dma_buff], sb16_dma_buffer
2382 00013797 [00000200]          <1>
2383 0001379B C705[649C0100]0000- <1>       mov   dword [audio_dmabuff_size], 65536
2383 000137A3 0100                <1>
2384                              <1>       ;xor  eax, eax
2385                              <1>       ;mov  [u.r0], eax ; 0 = no error, successful
2386 000137A5 C3                  <1>       retn
2387                              <1>
2388                              <1> sndc_init4:
2389                              <1>       ; EAX = Beginning address (physical)
2390                              <1>       ; EAX = 0 -> Allocate mem block from the 1st proper aperture
2391                              <1>       ; ECX = Number of bytes to be allocated (>0)
2392 000137A6 E8D22CFFFF          <1>       call  allocate_memory_block
2393 000137AB 0F828FFDFFFF        <1>       jc    sound_buff_error
2394                              <1>
2395                              <1>       ; set dma buffer address and size parameters
2396 000137B1 A3[609C0100]        <1>       mov   [audio_dma_buff], eax ; dma buffer address
2397 000137B6 890D[649C0100]      <1>       mov   [audio_dmabuff_size], ecx ; dma buffer size
2398                              <1> ;     ; EAX = Beginning (physical) addr of the allocated mem block
2399                              <1> ;     ; ECX = Num of allocated bytes (rounded up to page borders)
2400                              <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:
000137BC 803D[419C0100]03   <1>         cmp     byte [audio_device], 3
                            <1>         ;jne    short sndc_init5
000137C3 7506               <1>         jne     short sndc_init14 ; 28/05/2017
000137C5 E88D190000         <1>         call    set_vt8233_bdl
                            <1> sndc_init5:
                            <1>         ;sub    eax, eax ; 0
                            <1>         ;mov    [u.r0], eax ; 0 = no error, successful
000137CA C3                 <1>         retn
                            <1> sndc_init14:
000137CB E8541F0000         <1>         call    set_ac97_bdl
                            <1>         ;jmp    short sndc_init5
000137D0 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>         ; 28/07/2020
                            <1>         ; 27/07/2020
                            <1>         ; 28/05/2017
                            <1>         ; 15/05/2017, 20/05/2017
                            <1>         ; 21/04/2017, 24/04/2017
                            <1>         ; ... device check at first
000137D1 A0[419C0100]       <1>         mov     al, [audio_device]
000137D6 08C0               <1>         or      al, al ; 0 ; pc speaker or invalid
000137D8 0F8414ECFEFF       <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
000137DE 833D[549C0100]00   <1>         cmp     dword [audio_buffer], 0
000137E5 0F8655FDFFFF       <1>         jna     sound_buff_error
000137EB A0[B3030300]       <1>         mov     al, [u.uno]
000137F0 3A05[699C0100]     <1>         cmp     al, [audio_user]
000137F6 0F858FFEFFFF       <1>         jne     sndc_owner_error
                            <1>
000137FC 66890D[729C0100]   <1>         mov     [audio_freq], cx ; sample frequency (Hertz)
00013803 88D8               <1>         mov     al, bl
00013805 2401               <1>         and     al, 1 ; mono/stereo (1= stereo)
00013807 FEC0               <1>         inc     al ; channels
00013809 A2[719C0100]       <1>         mov     [audio_stmo], al ; sound channels (1 or 2)
0001380E B008               <1>         mov     al, 8
00013810 F6C302             <1>         test    bl, 2 ; bits per sample (1= 16 bit)
00013813 7402               <1>         jz      short snd_play_bps
00013815 D0E0               <1>         shl     al, 1
                            <1> snd_play_bps:
00013817 A2[709C0100]       <1>         mov     [audio_bps], al
                            <1>
                            <1>         ; Transfer ring 3 (user's) audio buffer content to dma buffer
0001381C 8B3D[609C0100]     <1>         mov     edi, [audio_dma_buff] ; dma buffer (ring 0)
00013822 09FF               <1>         or      edi, edi
00013824 0F8416FDFFFF       <1>         jz      sound_buff_error
                            <1>
                            <1>         ; 27/07/2020
                            <1>
0001382A 8B35[589C0100]     <1>         mov     esi, [audio_p_buffer] ; physical address (ring 3)
                            <1>         ;mov    ecx, [audio_buff_size] ; 15/05/2017
00013830 8B0D[649C0100]     <1>         mov     ecx, [audio_dmabuff_size] ; 27/07/2020
                            <1>         ;or     ecx, ecx
                            <1>         ;jz     sound_buff_error
                            <1>         ; 28/07/2020
00013836 D1E9               <1>         shr     ecx, 1  ; dma half buffer size
                            <1>
00013838 8035[689C0100]01   <1>         xor     byte [audio_flag], 1 ;  0 -> 1, 1 -> 0
0001383F 7502               <1>         jnz     short snd_play_0 ; [audio_flag] = 1
                            <1>                                  ; fill dma half buffer 1
                            <1>         ; [audio_flag] = 0
                            <1>
                            <1>         ; fill dma half buffer 2
00013841 01CF               <1>         add     edi, ecx
```

```
2506                                    <1>
2507                                    <1> snd_play_0:
2508                                    <1>         ;rep   movsb
2509 00013843 C1E902                   <1>         shr    ecx, 2  ; convert byte count to dword count
2510 00013846 F3A5                     <1>         rep    movsd
2511                                    <1>
2512                                    <1>         ; here, if [audio_flag] = 0, interrupt handler will update
2513                                    <1>                         ; dma half buffer 2
2514                                    <1>                         ; (user's audio buffer data will be
2515                                    <1>                         ; copied into dma half buffer 2)
2516                                    <1>         ;; 20/05/2017
2517                                    <1>         ;mov   byte [audio_flag], 1 ; next half (on next time)
2518                                    <1>
2519                                    <1>         ; 24/04/2017
2520 00013848 A0[419C0100]             <1>         mov    al, [audio_device]
2521 0001384D 3C03                     <1>         cmp    al, 3 ; VT8233 (VT8237R)
2522 0001384F 7410                     <1>         je     short snd_play_1
2523 00013851 3C01                     <1>         cmp    al, 1 ; Sound Blaster 16
2524 00013853 7512                     <1>         jne    short snd_play_2  ; 28/05/2017
2525 00013855 E8051B0000               <1>         call   SbInit_play
2526 0001385A 0F82E7FCFFFF             <1>         jc     soundc_respond_err
2527 00013860 C3                       <1>         retn
2528                                    <1>
2529                                    <1> snd_play_1:
2530 00013861 E828190000               <1>         call   vt8233_start_play
2531 00013866 C3                       <1>         retn
2532                                    <1>
2533                                    <1> snd_play_2:
2534                                    <1>         ; 28/05/2017
2535                                    <1>         ;cmp   al, 2 ; AC'97
2536                                    <1>         ;jne   short snd_play_3
2537                                    <1>
2538 00013867 E8EC1E0000               <1>         call   ac97_start_play
2539 0001386C C3                       <1>         retn
2540                                    <1>
2541                                    <1> ;snd_play_3:
2542                                    <1> ;       ;call  hda_start_play
2543                                    <1> ;       retn
2544                                    <1>
2545                                    <1> sound_pause:
2546                                    <1>         ; FUNCTION = 5
2547                                    <1>         ; Pause
2548                                    <1>         ; 28/05/2017
2549                                    <1>         ; 24/04/2017
2550                                    <1>         ; 22/04/2017
2551 0001386D E814030000               <1>         call   snd_dev_check
2552 00013872 7275                     <1>         jc     short snd_nothing ; temporary.
2553 00013874 E81A030000               <1>         call   snd_buf_check
2554 00013879 726E                     <1>         jc     short snd_nothing ; temporary.
2555 0001387B A0[419C0100]             <1>         mov    al, [audio_device]
2556 00013880 3C03                     <1>         cmp    al, 3 ; VIA VT 8237R (vt8233)
2557 00013882 7409                     <1>         je     short snd_pause_1
2558 00013884 3C01                     <1>         cmp    al, 1 ; Sound Blaster 16
2559 00013886 750A                     <1>         jne    short snd_pause_2 ; 28/05/2017
2560 00013888 E9B01C0000               <1>         jmp    sb16_pause
2561                                    <1> snd_pause_1:
2562 0001388D E9B5190000               <1>         jmp    vt8233_pause
2563                                    <1> snd_pause_2:
2564                                    <1>         ; 28/05/2017
2565                                    <1>         ;cmp   al, 2 ; AC'97
2566                                    <1>         ;jne   short snd_nothing ; temporary.
2567 00013892 E94F200000               <1>         jmp    ac97_pause
2568                                    <1>
2569                                    <1> sound_continue:
2570                                    <1>         ; FUNCTION = 6
2571                                    <1>         ; Continue to play
2572                                    <1>         ; 28/05/2017
2573                                    <1>         ; 22/04/2017
2574 00013897 E8EA020000               <1>         call   snd_dev_check
2575 0001389C 724B                     <1>         jc     short snd_nothing ; temporary.
2576 0001389E E8F0020000               <1>         call   snd_buf_check
2577 000138A3 7244                     <1>         jc     short snd_nothing ; temporary.
2578 000138A5 A0[419C0100]             <1>         mov    al, [audio_device]
2579 000138AA 3C03                     <1>         cmp    al, 3 ; VIA VT 8237R (vt8233)
2580 000138AC 7409                     <1>         je     short snd_cont_1
2581 000138AE 3C01                     <1>         cmp    al, 1 ; Sound Blaster 16
2582 000138B0 750A                     <1>         jne    short snd_cont_2 ; 28/05/2017
2583 000138B2 E9A91C0000               <1>         jmp    sb16_continue
2584                                    <1> snd_cont_1:
2585 000138B7 E93C190000               <1>         jmp    vt8233_play
2586                                    <1> snd_cont_2:
2587                                    <1>         ; 28/05/2017
2588                                    <1>         ;cmp   al, 2 ; AC'97
2589                                    <1>         ;jne   short snd_nothing ; temporary.
2590 000138BC E9ED1E0000               <1>         jmp    ac97_play
2591                                    <1>
2592                                    <1> sound_stop:
2593                                    <1>         ; FUNCTION = 7
2594                                    <1>         ; Stop playing
2595                                    <1>         ; 28/05/2017
2596                                    <1>         ; 24/05/2017
2597                                    <1>         ; 21/04/2017, 22/04/2017, 24/04/2017
2598 000138C1 E8C0020000               <1>         call   snd_dev_check
2599 000138C6 7221                     <1>         jc     short snd_nothing ; temporary.
2600                                    <1>         ;call  snd_buf_check
2601 000138C8 E8CF020000               <1>         call   snd_user_check ; 24/05/2017
2602 000138CD 721A                     <1>         jc     short snd_nothing ; temporary.
2603                                    <1>
2604 000138CF A0[419C0100]             <1>         mov    al, [audio_device]
2605 000138D4 3C03                     <1>         cmp    al, 3 ; VIA VT 8237R (vt8233)
2606 000138D6 0F8472180000             <1>         je     vt8233_stop
2607                                    <1>         ; 28/05/2017
2608                                    <1>         ;ja    short snd_nothing
2609 000138DC 3C01                     <1>         cmp    al, 1 ; Sound Blaster 16
2610 000138DE 0F849F1C0000             <1>         je     sb16_stop
```

```
2611                            <1>       ;cmp   al, 2
2612                            <1>       ;je    short ac97_stop
2613 000138E4 E9CF1F0000        <1>       jmp    ac97_stop ; temporary.
2614                            <1>       ;jmp   hda_stop
2615                            <1>
2616                            <1> snd_nothing:
2617                            <1>       ; 21/04/2017
2618 000138E9 C3                <1>       retn
2619                            <1>
2620                            <1> soundc_reset:
2621                            <1>       ; FUNCTION = 8
2622                            <1>       ; Reset Audio Controller
2623                            <1>       ; 28/05/2017
2624                            <1>       ; 22/04/2017
2625 000138EA E897020000        <1>       call   snd_dev_check
2626 000138EF 72F8              <1>       jc     snd_nothing ; temporary.
2627 000138F1 E89D020000        <1>       call   snd_buf_check
2628 000138F6 72F1              <1>       jc     snd_nothing ; temporary.
2629                            <1>
2630 000138F8 A0[419C0100]      <1>       mov    al, [audio_device]
2631 000138FD 3C03              <1>       cmp    al, 3 ; VIA VT 8237R (vt8233)
2632 000138FF 0F844E190000      <1>       je     vt8233_reset
2633 00013905 77E2              <1>       ja     short snd_nothing ; temporary.
2634                            <1>       ;ja    hda_reset
2635 00013907 3C01              <1>       cmp    al, 1 ; Sound Blaster 16
2636 00013909 0F8528200000      <1>       jne    ac97_reset
2637 0001390F E8C11C0000        <1>       call   sb16_reset
2638 00013914 0F822DFCFFFF      <1>       jc     soundc_respond_err
2639 0001391A C3                <1>       retn
2640                            <1>
2641                            <1> soundc_cancel:
2642                            <1>       ; FUNCTION = 9
2643                            <1>       ; Cancel audio callback service
2644                            <1>       ; 22/04/2017
2645 0001391B A0[699C0100]      <1>       mov    al, [audio_user]
2646 00013920 3A05[B3030300]    <1>       cmp    al, [u.uno]
2647 00013926 75C1              <1>       jne    short snd_nothing
2648                            <1>       ; RESET (audio) INTERRUPT CALLBACK SERVICE
2649 00013928 8A1D[439C0100]    <1>       mov    bl, [audio_intr] ; IRQ number
2650 0001392E A0[B3030300]      <1>       mov    al, [u.uno]
2651 00013933 28FF              <1>       sub    bh, bh ; 0 ; unlink IRQ from user service
2652 00013935 E8EC020000        <1>       call   set_irq_callback_service
2653 0001393A 0F8250FDFFFF      <1>       jc     sndc_perm_error ; 'permission denied' error
2654 00013940 C3                <1>       retn
2655                            <1>
2656                            <1> sound_dalloc:
2657                            <1>       ; FUNCTION = 10
2658                            <1>       ; Deallocate (ring 3) audio buffer
2659                            <1>       ; 22/04/2017
2660 00013941 A0[699C0100]      <1>       mov    al, [audio_user]
2661 00013946 3A05[B3030300]    <1>       cmp    al, [u.uno]
2662 0001394C 759B              <1>       jne    short snd_nothing
2663 0001394E 8B1D[549C0100]    <1>       mov    ebx, [audio_buffer]
2664                            <1>       ;or    ebx, ebx
2665                            <1>       ;jz    short snd_nothing
2666 00013954 8B0D[5C9C0100]    <1>       mov    ecx, [audio_buff_size]
2667 0001395A E8792EFFFF        <1>       call   deallocate_user_pages
2668 0001395F 31C0              <1>       xor    eax, eax
2669 00013961 A3[549C0100]      <1>       mov    [audio_buffer], eax ; 0
2670 00013966 A2[699C0100]      <1>       mov    [audio_user], al ; 0
2671 0001396B C3                <1>       retn
2672                            <1>
2673                            <1> sound_volume:
2674                            <1>       ; FUNCTION = 11
2675                            <1>       ; Set sound volume level
2676                            <1>       ; 28/05/2017
2677                            <1>       ; 20/05/2017
2678                            <1>       ; 22/04/2017, 24/04/2017
2679                            <1>       ; bl = component (0 = master/playback/lineout volume)
2680                            <1>       ; cl = left channel volume level (0 to 31)
2681                            <1>       ; ch = right channel volume level (0 to 31)
2682                            <1>
2683 0001396C 80FB80            <1>       cmp    bl, 80h
2684 0001396F 720E              <1>       jb     short snd_vol_1
2685 00013971 0F8772FFFFFF      <1>       ja     snd_nothing ; temporary.
2686                            <1>       ; Set volume level for next play (BL>= 80h)
2687 00013977 66890D[769C0100]  <1>       mov    [audio_master_volume], cx
2688 0001397E C3                <1>       retn
2689                            <1> snd_vol_1:
2690                            <1>       ; set volume level immediate (BL< 80h)
2691 0001397F 80FB00            <1>       cmp    bl, 0
2692 00013982 0F8761FFFFFF      <1>       ja     snd_nothing ; temporary.
2693                            <1>
2694 00013988 E8F9010000        <1>       call   snd_dev_check
2695 0001398D 0F8256FFFFFF      <1>       jc     snd_nothing ; temporary.
2696 00013993 E8FB010000        <1>       call   snd_buf_check
2697 00013998 0F824BFFFFFF      <1>       jc     snd_nothing ; temporary.
2698                            <1>
2699 0001399E A0[419C0100]      <1>       mov    al, [audio_device]
2700 000139A3 3C03              <1>       cmp    al, 3 ; VIA VT 8237R (vt8233)
2701 000139A5 0F84C1180000      <1>       je     vt8233_volume
2702                            <1>       ; 28/05/2017
2703 000139AB 0F8738FFFFFF      <1>       ja     snd_nothing ; temporary.
2704                            <1>       ;ja    hda_volume
2705                            <1>       ; Sound Blaster 16
2706 000139B1 3C01              <1>       cmp    al, 1 ; SB 16
2707 000139B3 0F844F1B0000      <1>       je     sb16_volume
2708 000139B9 E90C1E0000        <1>       jmp    ac97_volume
2709                            <1>
2710                            <1> soundc_disable:
2711                            <1>       ; FUNCTION = 12
2712                            <1>       ; Disable audio device (and unlink DMA memory)
2713                            <1>       ; 28/05/2017
2714                            <1>       ; 24/05/2017
2715                            <1>       ; 22/04/2017
```

```
2716 000139BE E8C3010000          <1>       call   snd_dev_check
2717 000139C3 0F8270FBFFFF        <1>       jc     soundc_dev_err ; temporary.
2718                              <1>       ;call  snd_buf_check
2719                              <1>       ;jc    sndc_owner_error ; temporary.
2720                              <1>
2721 000139C9 A0[419C0100]        <1>       mov    al, [audio_device]
2722 000139CE 3C03                <1>       cmp    al, 3 ; VIA VT 8237R (vt8233)
2723 000139D0 7418                <1>       je     short snd_disable_1
2724 000139D2 0F8711FFFFFF        <1>       ja     snd_nothing ; temporary.
2725 000139D8 3C01                <1>       cmp    al, 1 ; Sound Blaster 16
2726 000139DA 7507                <1>       jne    short snd_disable_0
2727 000139DC E8A21B0000          <1>       call   sb16_stop
2728 000139E1 EB0C                <1>       jmp    short snd_disable_2
2729                              <1> snd_disable_0:
2730 000139E3 E8D01E0000          <1>       call   ac97_stop
2731 000139E8 EB05                <1>       jmp    short snd_disable_2
2732                              <1> snd_disable_1:
2733 000139EA E85F170000          <1>       call   vt8233_stop
2734                              <1> snd_disable_2:
2735 000139EF A0[439C0100]        <1>       mov    al, [audio_intr]
2736 000139F4 29DB                <1>       sub    ebx, ebx ; 0 = reset
2737 000139F6 E851FAFFFF          <1>       call   set_dev_IRQ_service
2738                              <1>
2739                              <1>       ;mov   al, [audio_intr]
2740 000139FB 28E4                <1>       sub    ah, ah ; 0 = reset
2741 000139FD E8B5F6FFFF          <1>       call   set_hardware_int_vector
2742                              <1>
2743 00013A02 31C0                <1>       xor    eax, eax
2744 00013A04 A2[419C0100]        <1>       mov    byte [audio_device], al
2745 00013A09 A2[439C0100]        <1>       mov    byte [audio_intr], al
2746 00013A0E 8705[609C0100]      <1>       xchg   eax, [audio_dma_buff]
2747                              <1>       ; 24/05/2017
2748                              <1>       ;or    eax, eax
2749                              <1>       ;jz    short snd_disable_3
2750                              <1>       ;cmp   eax, sb16_dma_buffer ; default DMA buffer
2751                              <1>       ;je    short snd_disable_3
2752 00013A14 803D[409C0100]00    <1>       cmp    byte [audio_pci], 0 ; AC97 audio controller ?
2753 00013A1B 7612                <1>       jna    short snd_disable_3
2754 00013A1D C605[409C0100]00    <1>       mov    byte [audio_pci], 0
2755                              <1>       ;sub   ecx, ecx
2756                              <1>       ;xchg  ecx, [audio_dmabuff_size]
2757 00013A24 8B0D[649C0100]      <1>       mov    ecx, [audio_dmabuff_size]
2758 00013A2A E85B2CFFFF          <1>       call   deallocate_memory_block
2759                              <1> snd_disable_3:
2760 00013A2F C3                  <1>       retn
2761                              <1>
2762                              <1> sound_dma_map:
2763                              <1>       ; FUNCTION = 13
2764                              <1>       ; Map audio dma buff addr to user's buffer addr
2765                              <1>       ; 12/05/2017
2766 00013A30 21C9                <1>       and    ecx, ecx
2767 00013A32 0F8408FBFFFF        <1>       jz     sound_buff_error
2768 00013A38 803D[419C0100]01    <1>       cmp    byte [audio_device], 1
2769 00013A3F 7229                <1>       jb     short snd_dma_map_1
2770                              <1> snd_dma_map_0:
2771 00013A41 A1[609C0100]        <1>       mov    eax, [audio_dma_buff]
2772 00013A46 21C0                <1>       and    eax, eax
2773 00013A48 7420                <1>       jz     short snd_dma_map_1
2774                              <1>       ;
2775 00013A4A 8A1D[699C0100]      <1>       mov    bl, [audio_user]
2776 00013A50 08DB                <1>       or     bl, bl
2777 00013A52 7416                <1>       jz     short snd_dma_map_1
2778 00013A54 3A1D[B3030300]      <1>       cmp    bl, [u.uno]
2779 00013A5A 0F852BFCFFFF        <1>       jne    sndc_owner_error
2780                              <1>       ;
2781 00013A60 8B1D[649C0100]      <1>       mov    ebx, [audio_dmabuff_size]
2782 00013A66 21DB                <1>       and    ebx, ebx
2783 00013A68 750A                <1>       jnz    short snd_dma_map_2
2784                              <1> snd_dma_map_1:
2785 00013A6A B8[00000200]        <1>       mov    eax, sb16_dma_buffer
2786 00013A6F BB00000100          <1>       mov    ebx, 65536
2787                              <1> snd_dma_map_2:
2788 00013A74 81C1FF0F0000        <1>       add    ecx, PAGE_SIZE-1 ; 4095
2789 00013A7A 6681E100F0          <1>       and    cx, ~PAGE_OFF ; not 4095
2790 00013A7F 39D9                <1>       cmp    ecx, ebx
2791 00013A81 0F87B9FAFFFF        <1>       ja     sound_buff_error
2792 00013A87 50                  <1>       push   eax
2793 00013A88 89D3                <1>       mov    ebx, edx
2794 00013A8A C1E90C              <1>       shr    ecx, 12 ; byte count to page count
2795                              <1>       ; eax = physical address of (audio) dma buffer
2796                              <1>       ; ebx = virtual address of (audio) dma buffer (user's pgdir)
2797                              <1>       ; ecx = page count (>0)
2798 00013A8D E8682CFFFF          <1>       call   direct_memory_access
2799 00013A92 58                  <1>       pop    eax
2800 00013A93 0F82A7FAFFFF        <1>       jc     sound_buff_error
2801 00013A99 A3[64030300]        <1>       mov    [u.r0], eax
2802 00013A9E C3                  <1>       retn
2803                              <1>
2804                              <1> soundc_info:
2805                              <1>       ; FUNCTION = 14
2806                              <1>       ; Get Audio Controller Info
2807                              <1>       ; 10/06/2017
2808                              <1>       ; 05/06/2017
2809 00013A9F 20DB                <1>       and    bl, bl ; 0
2810 00013AA1 740A                <1>       jz     short sndc_info_0
2811                              <1>       ; invalid parameter !
2812 00013AA3 B817000000          <1>       mov    eax, ERR_INV_PARAMETER ; 23
2813                              <1> ;sndc_inf_error:
2814                              <1> ;     mov    [u.r0], eax
2815                              <1> ;     mov    [u.error], eax
2816                              <1> ;     jmp    error
2817 00013AA8 E99FFAFFFF          <1>       jmp    sysaudio_err
2818                              <1>
2819                              <1> sndc_info_0:
2820 00013AAD E8D4000000          <1>       call   snd_dev_check
```

```
00013AB2 0F8281FAFFFF        <1>       jc     soundc_dev_err
                             <1>
00013AB8 8B1D[4C9C0100]      <1>       mov    ebx, [audio_vendor]
00013ABE 8B0D[489C0100]      <1>       mov    ecx, [audio_dev_id]
                             <1>       ;mov   al,  [audio_device]
00013AC4 3C02                <1>       cmp    al, 2 ; AC'97 (ICH)
00013AC6 7513                <1>       jne    short sndc_info_1
                             <1>       ; Intel AC97 (ICH) Audio Controller (=2)
00013AC8 668B15[7A9C0100]    <1>       mov    dx, [NABMBAR]
00013ACF C1E210              <1>       shl    edx, 16
00013AD2 668B15[789C0100]    <1>       mov    dx, [NAMBAR]
00013AD9 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) !!!
00013ADB 0FB715[469C0100]    <1>       movzx  edx, word [audio_io_base]
                             <1> sndc_info_2:
00013AE2 88C4                <1>       mov    ah, al ;  [audio_device]
00013AE4 A0[439C0100]        <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
00013AE9 A3[64030300]        <1>       mov    [u.r0], eax
00013AEE 8B2D[60030300]      <1>       mov    ebp, [u.usp]
00013AF4 895D10              <1>       mov    [ebp+16], ebx  ; ebx
00013AF7 895514              <1>       mov    [ebp+20], edx  ; edx
00013AFA 894D18              <1>       mov    [ebp+24], ecx  ; ecx
                             <1>
00013AFD C3                  <1>       retn
                             <1>
                             <1> sound_data:
                             <1>       ; FUNCTION = 15
                             <1>       ; Get Current Sound data for graphics
                             <1>       ; 22/06/2017
                             <1>       ;
00013AFE E883000000          <1>       call   snd_dev_check
00013B03 0F8230FAFFFF        <1>       jc     soundc_dev_err ; Device not ready !
                             <1>
00013B09 80FB00              <1>       cmp    bl, 0
00013B0C 760A                <1>       jna    short sound_data_0
                             <1>
                             <1>       ; Only PCM OUT buffer data is valid for now!
00013B0E B817000000          <1>       mov    eax, ERR_INV_PARAMETER  ; 23
00013B13 E934FAFFFF          <1>       jmp    sysaudio_err
                             <1>
                             <1> sound_data_0:
00013B18 A1[609C0100]        <1>       mov    eax, [audio_dma_buff]
00013B1D 09C0                <1>       or     eax, eax
00013B1F 0F841BFAFFFF        <1>       jz     sound_buff_error
                             <1>
00013B25 803D[419C0100]04    <1>       cmp    byte [audio_device], 4 ; Intel HDA
00013B2C 744F                <1>       je     short sound_data_4 ; temporary ! (22/06/2017)
                             <1>
00013B2E 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>
00013B30 0F84791F0000        <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>
00013B36 3B0D[649C0100]      <1>       cmp    ecx, [audio_dmabuff_size]
00013B3C 0F87FEF9FFFF        <1>       ja     sound_buff_error
                             <1>
00013B42 89D0                <1>       mov    eax, edx
00013B44 25FF0F0000          <1>       and    eax, PAGE_OFF ; 4095 (0FFFh)
00013B49 81F900100000        <1>       cmp    ecx, 4096
00013B4F 7605                <1>       jna    short sound_data_2
00013B51 B900100000          <1>       mov    ecx, 4096 ; max. 1 page
                             <1> sound_data_2:
00013B56 01C8                <1>       add    eax, ecx
00013B58 3D00100000          <1>       cmp    eax, 4096
00013B5D 7606                <1>       jna    short sound_data_3
00013B5F 6625FF0F            <1>       and    ax, PAGE_OFF ; 4095 (0FFFh)
00013B63 29C1                <1>       sub    ecx, eax
                             <1>       ; here, ECX has been adjusted to fit
                             <1>       ;       in page border..  (<= 4096, >0)
                             <1> sound_data_3:
00013B65 51                  <1>       push   ecx
00013B66 52                  <1>       push   edx
00013B67 89D3                <1>       mov    ebx, edx
```

```
2926 00013B69 E87A27FFFF          <1>       call   get_physical_addr
2927 00013B6E 5A                  <1>       pop    edx
2928 00013B6F 59                  <1>       pop    ecx
2929 00013B70 0F82CAF9FFFF        <1>       jc     sound_buff_error
2930                              <1>
2931                              <1>       ; eax = physical address of user's buffer
2932 00013B76 89C3                <1>       mov    ebx, eax
2933                              <1>       ; ecx = byte (transfer) count
2934                              <1>       ;call  get_current_sound_data
2935                              <1>       ;retn
2936 00013B78 E98F1E0000          <1>       jmp    get_current_sound_data
2937                              <1>
2938                              <1> sound_data_4:
2939                              <1>       ; Intel HDA code is not ready yet !
2940                              <1>       ; 22/06/2017
2941 00013B7D 31C0                <1>       xor    eax, eax
2942 00013B7F 48                  <1>       dec    eax
2943 00013B80 A3[64030300]        <1>       mov    [u.r0], eax ; 0FFFFFFFFh
2944 00013B85 C3                  <1>       retn
2945                              <1>
2946                              <1> snd_dev_check:
2947                              <1>       ; 10/06/2017
2948                              <1>       ; 05/06/2017
2949                              <1>       ; 24/05/2017
2950                              <1>       ; 22/04/2017
2951                              <1>       ; 21/04/2017
2952                              <1>       ; ... device check at first
2953 00013B86 A0[419C0100]        <1>       mov    al, [audio_device]
2954 00013B8B 3C01                <1>       cmp    al, 1 ; SB 16
2955 00013B8D 7203                <1>       jb     short snd_dev_chk_retn ; error !
2956                              <1>       ;cmp   al, 4 ; Intel HDA
2957                              <1>       ;ja    short snd_dbchk_stc ; invalid !
2958                              <1>       ; 10/06/2017
2959 00013B8F 3C05                <1>       cmp    al, 5
2960 00013B91 F5                  <1>       cmc
2961                              <1> snd_dev_chk_retn:
2962 00013B92 C3                  <1>       retn
2963                              <1>
2964                              <1> snd_buf_check:
2965                              <1>       ; 10/06/2017
2966                              <1>       ; 22/04/2017
2967                              <1>       ; 21/04/2017
2968                              <1>       ; ... buffer & (buffer) owner check at second
2969 00013B93 833D[549C0100]00    <1>       cmp    dword [audio_buffer], 0
2970 00013B9A 760D                <1>       jna    short snd_dbchk_stc
2971                              <1> snd_user_check:
2972 00013B9C A0[B3030300]        <1>       mov    al, [u.uno]
2973 00013BA1 3A05[699C0100]      <1>       cmp    al, [audio_user]
2974                              <1>       ;jne   short snd_dbchk_stc
2975                              <1>       ;retn
2976 00013BA7 74E9                <1>       je     short snd_dev_chk_retn
2977                              <1>
2978                              <1> snd_dbchk_stc:
2979 00013BA9 F9                  <1>       stc
2980 00013BAA C3                  <1>       retn
2981                              <1>
2982                              <1> sound_update:
2983                              <1>       ; FUNCTION = 16
2984                              <1>       ; bl =
2985                              <1>       ;    0 = automatic (sequental) update (with flag switch!)
2986                              <1>       ;    1 = update dma half buffer 1 (without flag switch!)
2987                              <1>       ;    2 = update dma half buffer 2 (without flag switch!)
2988                              <1>       ;  FFh = get current flag value
2989                              <1>       ;      0 = dma half buffer 1 (will be played next)
2990                              <1>       ;      1 = dma half buffer 2 (will be played next)
2991                              <1>
2992                              <1>       ; 10/10/2017
2993                              <1>
2994                              <1>       ; ... device check at first
2995 00013BAB A0[419C0100]        <1>       mov    al, [audio_device]
2996 00013BB0 08C0                <1>       or     al, al ; 0 ; pc speaker or invalid
2997 00013BB2 0F8481F9FFFF        <1>       jz     soundc_dev_err
2998                              <1>
2999                              <1>       ; ... buffer & (buffer) owner check at second
3000 00013BB8 833D[549C0100]00    <1>       cmp    dword [audio_buffer], 0
3001 00013BBF 0F867BF9FFFF        <1>       jna    sound_buff_error
3002 00013BC5 A0[B3030300]        <1>       mov    al, [u.uno]
3003 00013BCA 3A05[699C0100]      <1>       cmp    al, [audio_user]
3004 00013BD0 0F85B5FAFFFF        <1>       jne    sndc_owner_error
3005                              <1>
3006                              <1>       ; Transfer ring 3 (user's) audio buffer content to dma buffer
3007 00013BD6 8B3D[609C0100]      <1>       mov    edi, [audio_dma_buff] ; dma buffer (ring 0)
3008 00013BDC 09FF                <1>       or     edi, edi
3009 00013BDE 0F845CF9FFFF        <1>       jz     sound_buff_error
3010 00013BE4 8B35[589C0100]      <1>       mov    esi, [audio_p_buffer] ; physical address (ring 3)
3011 00013BEA 8B0D[5C9C0100]      <1>       mov    ecx, [audio_buff_size]
3012                              <1>
3013                              <1>       ;movzx eax, byte [audio_flag]
3014 00013BF0 A0[689C0100]        <1>       mov    al, [audio_flag]
3015                              <1>
3016 00013BF5 FEC3                <1>       inc    bl
3017 00013BF7 7427                <1>       jz     short snd_update_3 ; bl = 0FFh
3018 00013BF9 FECB                <1>       dec    bl
3019 00013BFB 7411                <1>       jz     short snd_update_0 ; bl = 0
3020                              <1>
3021 00013BFD 80FB02              <1>       cmp    bl, 2
3022 00013C00 7417                <1>       je     short snd_update_1 ; dma half buffer 2
3023 00013C02 7217                <1>       jb     short snd_update_2 ; dma half buffer 1
3024                              <1>
3025                              <1>       ; invalid parameter !
3026 00013C04 B817000000          <1>       mov    eax, ERR_INV_PARAMETER ; 23
3027                              <1> ;     mov    [u.r0], eax
3028                              <1> ;     mov    [u.error], eax
3029                              <1> ;     jmp    error
3030 00013C09 E93EF9FFFF          <1>       jmp    sysaudio_err
```

```
3031                                    <1>
3032                                    <1> snd_update_0:
3033 00013C0E 8035[689C0100]01          <1>         xor     byte [audio_flag], 1 ; update flag !!!
3034 00013C15 3C01                      <1>         cmp     al, 1
3035 00013C17 7202                      <1>         jb      short snd_update_2 ; dma half buffer 1
3036                                    <1> snd_update_1:
3037                                    <1>         ; dma half buffer 2
3038 00013C19 01CF                      <1>         add     edi, ecx
3039                                    <1> snd_update_2:
3040                                    <1>         ;rep    movsb
3041 00013C1B C1E902                    <1>         shr     ecx, 2
3042 00013C1E F3A5                      <1>         rep     movsd
3043                                    <1> snd_update_3:
3044 00013C20 A3[64030300]              <1>         mov     [u.r0], eax
3045                                    <1>
3046 00013C25 C3                        <1>         retn
3047                                    <1>
3048                                    <1> set_irq_callback_service:
3049                                    <1>         ; 03/08/2020
3050                                    <1>         ; 10/06/2017
3051                                    <1>         ; 12/05/2017
3052                                    <1>         ; 24/04/2017
3053                                    <1>         ; 22/04/2017
3054                                    <1>         ; caller: 'syscalbac' or 'sysaudio' or ...
3055                                    <1>         ; 13/04/2017, 14/04/2017, 17/04/2017
3056                                    <1>         ; 24/02/2017, 26/02/2017, 28/02/2017
3057                                    <1>         ; 21/02/2017 - TRDOS 386 (TRDOS v2.0)
3058                                    <1>         ;
3059                                    <1>         ; Link or unlink IRQ callback service to/from user (ring 3)
3060                                    <1>         ;
3061                                    <1>         ; INPUT ->
3062                                    <1>         ;       If AL = 0, the caller is 'syscalbac';
3063                                    <1>         ;          otherwise, the caller is 'sysaudio' or ...
3064                                    <1>         ;          (AL = user number)
3065                                    <1>         ;
3066                                    <1>         ;       BL = IRQ number (Hardware interrupt request number)
3067                                    <1>         ;            (0 t0 15 but IRQ 0,1,2,6,8,14,15 are prohibited)
3068                                    <1>         ;            IRQ numbers 3,4,5,7,9,10,11,12,13 are valid
3069                                    <1>         ;            (numbers >15 are invalid)
3070                                    <1>         ;
3071                                    <1>         ;       BH = 0 = Unlink IRQ (in BL) from user (ring 3) service
3072                                    <1>         ;            1 = Link IRQ by using Signal Response Byte method
3073                                    <1>         ;            2 = Link IRQ by using Callback service method
3074                                    <1>         ;            3 = Link IRQ by using Auto Increment S.R.B. method
3075                                    <1>         ;           >3 = invalid
3076                                    <1>         ;              (syscallback version will return to user)
3077                                    <1>         ;
3078                                    <1>         ;       CL = Signal Return/Response Byte value
3079                                    <1>         ;
3080                                    <1>         ;       If BH = 3, kernel will put a counter value ; 03/08/2020
3081                                    <1>         ;                 (into the S.R.B. addr)
3082                                    <1>         ;                between 0 to 255. (start value = CL+1)
3083                                    <1>         ;
3084                                    <1>         ;       NOTE: counter value, for example: even and odd numbers
3085                                    <1>         ;             may be used for -audio- DMA buffer switch
3086                                    <1>         ;             within double buffer method, etc.
3087                                    <1>         ;
3088                                    <1>         ;       EDX = Signal return (Response) byte address
3089                                    <1>         ;                             - or -
3090                                    <1>         ;             Interrupt/Callback service/routine address
3091                                    <1>         ;
3092                                    <1>         ;             (virtual address in user's memory space)
3093                                    <1>         ;
3094                                    <1>         ; OUTPUT ->
3095                                    <1>         ;       CF = 0 & EAX = 0 -> Successful setting
3096                                    <1>         ;       CF = 1 & EAX > 0 -> IRQ is prohibited or locked
3097                                    <1>         ;                     by another process
3098                                    <1>         ;               eax = ERR_PERM_DENIED -> prohibited or locked
3099                                    <1>         ;               eax = ERR_INV_PARAMETER ->
3100                                    <1>         ;                     invalid parameter/option or bad address
3101                                    <1>         ;
3102                                    <1>         ; TRDOS 386 - IRQ CALLBACK structures (parameters):
3103                                    <1>         ;
3104                                    <1>         ;       [u.irqlock] = 1 word, IRQ flags (0-15) that indicates
3105                                    <1>         ;                     which IRQs are locked by (that) user.
3106                                    <1>         ;                      Lock and unlock (by user) will change
3107                                    <1>         ;                     these flags or 'terminate process' (sysexit)
3108                                    <1>         ;                     will clear these flags and unlock those IRQs.
3109                                    <1>         ;
3110                                    <1>         ;                     Bit 0 is for IRQ 0 and Bit 15 is for IRQ 15
3111                                    <1>         ;
3112                                    <1>         ;       IRQ(x).owner      : 1 byte, user, [u.uno], 0 = free (unlocked)
3113                                    <1>         ;
3114                                    <1>         ;       IRQ(x).method : 1 byte for callback method & status
3115                                    <1>         ;                       0 = Signal Response Byte method
3116                                    <1>         ;                       1 = Callback service method
3117                                    <1>         ;                       >1 = invalid for current 'syscalback'.
3118                                    <1>         ;                     or(+) 80h = IRQ is in use by system (ring 0)
3119                                    <1>         ;                                 function (audio etc.) or
3120                                    <1>         ;                              a device driver.
3121                                    <1>         ;                     (system function will ignore the lock/owner)
3122                                    <1>         ;
3123                                    <1>         ;       IRQ(x).srb: 1 byte, Signal Return/Response byte value
3124                                    <1>         ;                       (a fixed value by user or a counter value
3125                                    <1>         ;                      from 0 to 255, which is increased by every
3126                                    <1>         ;                      interrupt just before putting it into
3127                                    <1>         ;                      the Signal Response byte address
3128                                    <1>         ;                      (This is not used in callback serv method)
3129                                    <1>         ;
3130                                    <1>         ;       IRQ(x).addr       : 1 dword
3131                                    <1>         ;                       Signal Response Byte address (physical)
3132                                    <1>         ;                                 -or-
3133                                    <1>         ;                       Callback service address (virtual)
3134                                    <1>         ;
3135                                    <1>         ;       IRQ(x).dev: 1 byte
```

```
3136                                    <1>     ;                   0 = Default device or kernel function
3137                                    <1>     ;                               -or-
3138                                    <1>     ;                   1-255 = Assigned device driver number
3139                                    <1>     ;
3140                                    <1>     ;       (x) = 3,4,5,7,9,10,11,12,13
3141                                    <1>     ;
3142                                    <1>
3143 00013C26 80FB0F                   <1>     cmp   bl, 15
3144 00013C29 7729                     <1>     ja    short scbs_2
3145                                    <1>
3146 00013C2B 80FF03                   <1>     cmp   bh, 3
3147 00013C2E 7724                     <1>     ja    short scbs_2  ; invalid parameter
3148                                    <1>
3149 00013C30 0FB6FB                   <1>     movzx  edi, bl ; save IRQ number
3150                                    <1>
3151                                    <1>           ;  IRQ 0,1,2,6,8,14,15 are prohibited
3152                                    <1>     ;IRQenum: ; 'trdosk9.s'
3153                                    <1>     ;     db  0,0,0,1,2,3,0,4,0,5,6,7,8,9,0,0
3154                                    <1>
3155 00013C33 0FB6B7[86490100]         <1>     movzx  esi, byte [edi+IRQenum] ; IRQ availability
3156                                    <1>                                    ; enumeration/index
3157                                    <1>     ;dec   esi
3158 00013C3A 664E                     <1>     dec   si
3159 00013C3C 780F                     <1>     js    short scbs_1 ;  0 -> 0FFFFh
3160                                    <1>
3161                                    <1>     ; ESI = IRQ callback parameters index number (0 to 8)
3162                                    <1>
3163 00013C3E 08FF                     <1>     or    bh, bh
3164 00013C40 7419                     <1>     jz    short scbs_4 ; unlink the IRQ (in BL)
3165                                    <1>
3166 00013C42 FECF                     <1>     dec   bh
3167                                    <1>     ; bh = method (0 = signal response byte, 1 = callback)
3168                                    <1>     ;             (2 = auto increment of signal response byte)
3169                                    <1>
3170 00013C44 80BE[F29B0100]00         <1>           cmp   byte [esi+IRQ.owner], 0 ; locked ?
3171 00013C4B 7637                     <1>           jna   short scbs_6 ; no... OK...
3172                                    <1>
3173                                    <1> scbs_1:
3174                                    <1>           ; permission denied (prohibited IRQ)
3175 00013C4D B80B000000               <1>           mov   eax, ERR_PERM_DENIED
3176 00013C52 F9                       <1>           stc
3177 00013C53 C3                       <1>           retn
3178                                    <1> scbs_2:
3179 00013C54 F9                       <1>           stc
3180                                    <1> scbs_3:
3181 00013C55 B817000000               <1>           mov   eax, ERR_INV_PARAMETER
3182 00013C5A C3                       <1>           retn
3183                                    <1>
3184                                    <1> scbs_4: ; unlink the requested IRQ (if it belongs to current user)
3185                                    <1>           ; 10/06/2017
3186                                    <1>           ; 22/04/2017
3187                                    <1>           ; 14/04/2017
3188                                    <1>           ; If AL = 0 -> The caller is 'syscalbac'
3189 00013C5B 8AA6[F29B0100]           <1>           mov   ah, [esi+IRQ.owner]
3190 00013C61 3A25[B3030300]           <1>           cmp   ah, [u.uno]
3191 00013C67 75E4                     <1>           jne   short scbs_1
3192                                    <1>
3193 00013C69 FE0D[D6030300]           <1>           dec   byte [u.irqc] ; decrease IRQ count (in use)
3194                                    <1>
3195                                    <1>           ;sub   ah, ah
3196                                    <1>           ;mov   [esi+IRQ.owner], ah ; 0 ; free !!!
3197                                    <1>           ;and   byte [esi+IRQ.method], 80h
3198                                    <1>           ;mov   [esi+IRQ.srb], ah ; 0
3199                                    <1>           ;mov   [esi+IRQ.dev], ah ; 0
3200                                    <1>           ;mov   dword [esi+IRQ.addr], 0
3201                                    <1>           ;mov   dword [u.r0], 0
3202                                    <1>
3203                                    <1>           ;mov   byte [esi+IRQ.owner], 0
3204                                    <1>
3205                                    <1>           ; 22/04/2017
3206 00013C6F 29C0                     <1>           sub   eax, eax
3207 00013C71 8886[F29B0100]           <1>           mov   [esi+IRQ.owner], al ; 0
3208                                    <1>           ; 10/06/2017
3209 00013C77 8686[049C0100]           <1>           xchg  al, [esi+IRQ.method]
3210 00013C7D 2480                     <1>           and   al, 80h
3211 00013C7F 745E                     <1>           jz    short scbs_12
3212                                    <1>           ; Audio device must be disabled -later- ! ([IRQ.medhod] = 80h)
3213                                    <1>
3214                                    <1> ;         cmp   byte [esi+IRQ.method], 80h ; device drv or kernel extension ?
3215                                    <1> ;         jb    short scbs_12 ; bh = 0 reset to default IRQ handler
3216                                    <1> ;
3217                                    <1> ;         and   al, al
3218                                    <1> ;         jz    short  scbs_5  ; the caller is 'syscalbac'
3219                                    <1> ;         ; The caller is 'sysaudio' or ...
3220 00013C81 30C0                     <1>           xor   al, al
3221                                    <1> ;         mov   [esi+IRQ.method], al ; 0 ; reset kernel extension flag
3222                                    <1> ;scbs_5:
3223                                    <1> ;         sub   ah, ah
3224                                    <1>           ;mov   [u.r0], eax ; 0
3225 00013C83 C3                       <1>           retn
3226                                    <1>
3227                                    <1> scbs_6:
3228                                    <1>           ; 14/04/2017
3229 00013C84 20C0                     <1>           and   al, al
3230 00013C86 7405                     <1>           jz    short scbs_7  ; the caller is 'syscalbac'
3231                                    <1>           ; AL = user number ([u.uno] or [audio.user] or ...)
3232                                    <1>           ; The caller is 'sysaudio' or ...
3233                                    <1>           ;
3234                                    <1>           ; bh = method (0 = signal response byte, 1 = callback)
3235                                    <1>           ;             (2 = auto increment of signal response byte)
3236                                    <1>
3237 00013C88 80CF80                   <1>           or    bh, 80h             ; Kernel extension flag !
3238 00013C8B EB0A                     <1>           jmp   short scbs_8
3239                                    <1> scbs_7:
3240 00013C8D 8A86[049C0100]           <1>           mov   al, [esi+IRQ.method] ; >= 80h = kernel is using this IRQ
```

```
3241 00013C93 2480              <1>       and   al, 80h ; use only bit 7 (kernel function flag)
3242 00013C95 08C7              <1>       or    bh, al        ; method
3243                            <1>                           ; 0 = signal response byte, 1 = callback
3244                            <1>                           ; 2 = auto increment of s.r.b.
3245                            <1> scbs_8:
3246 00013C97 A0[B3030300]      <1>       mov   al, [u.uno] ; user (process) number (1 to 16)
3247 00013C9C 8886[F29B0100]    <1>       mov   [esi+IRQ.owner], al ; lock the IRQ for user
3248 00013CA2 88BE[049C0100]    <1>       mov   [esi+IRQ.method], bh
3249                            <1>
3250                            <1> ;     test  bh, 1
3251                            <1> ;     jnz   short scbs_9  ; Callback method, CX will not be used
3252                            <1> ;
3253                            <1> ;     test  bh, 2         ; use auto increment (counter) method
3254                            <1> ;     jz    short scbs_10 ; (count can be used for buffer switch)
3255                            <1> ;scbs_9:
3256                            <1> ;     xor   ecx, ecx ; 0
3257                            <1> scbs_10:
3258                            <1>       ;mov  [esi+IRQ.method], bh
3259 00013CA8 888E[0D9C0100]    <1>       mov   [esi+IRQ.srb], cl
3260 00013CAE C686[FB9B0100]00  <1>       mov   byte [esi+IRQ.dev], 0 ; device number is always 0
3261                            <1>                           ; for this system call
3262                            <1>       ;test bh, 1
3263 00013CB5 80E701            <1>       and   bh, 1 ; 17/04/2017
3264 00013CB8 7513              <1>       jnz   short scbs_11 ; callback method, use virtual address
3265                            <1>
3266 00013CBA 53                <1>       push  ebx ; IRQ number (in BL)
3267 00013CBB 89D3              <1>       mov   ebx, edx
3268                            <1>       ; ebx = virtual address
3269                            <1>       ; [u.pgdir] = page directory's physical address
3270 00013CBD FE05[929B0100]    <1>       inc     byte [no_page_swap] ; 1
3271                            <1>                   ; Do not add this page to swap queue
3272                            <1>                   ; and remove it from swap queue if it is
3273                            <1>                   ; on the queue.
3274 00013CC3 E82026FFFF        <1>       call  get_physical_addr
3275 00013CC8 5B                <1>       pop   ebx
3276 00013CC9 728A              <1>       jc    scbs_3 ; invalid address !
3277                            <1>       ; eax = physical address of the virtual address in user's space
3278 00013CCB 89C2              <1>       mov   edx, eax
3279                            <1> scbs_11:
3280 00013CCD 66C1E602          <1>       shl   si, 2        ; byte (index) to dword (offset)
3281 00013CD1 8996[169C0100]    <1>       mov   [esi+IRQ.addr], edx
3282                            <1>
3283 00013CD7 FE05[D6030300]    <1>       inc   byte [u.irqc]; increase IRQ (in use) count
3284                            <1>
3285 00013CDD FEC7              <1>       inc    bh ; 17/04/2017
3286                            <1>       ; bh > 0 -> set to requested IRQ handler (IRQ_u_list)
3287                            <1> scbs_12:
3288 00013CDF 88D8              <1>       mov   al, bl ; IRQ number
3289 00013CE1 88FC              <1>       mov   ah, bh ; 0 = reset, >0 = set
3290 00013CE3 E8CFF3FFFF        <1>       call  set_hardware_int_vector
3291                            <1>
3292 00013CE8 31C0              <1>       xor   eax, eax
3293                            <1>       ;mov  [u.r0], eax ; 0
3294                            <1>
3295 00013CEA C3                <1>       retn  ; return with success (cf=0, eax=0)
3296                            <1>
3297                            <1>
3298                            <1> sysdma: ; DMA FUNCTIONS
3299                            <1>       ; 02/09/2017
3300                            <1>       ; 28/08/2017
3301                            <1>       ; 20/08/2017 - TRDOS 386 (TRDOS v2.0)
3302                            <1>       ;
3303                            <1>       ; Inputs:
3304                            <1>       ;     BH = 0 -> Allocate DMA buffer
3305                            <1>       ;        BL = 0 -> Use the system's default DMA
3306                            <1>       ;                  (SB16) Buffer
3307                            <1>       ;             Buffer Size (max.) = 65536 bytes
3308                            <1>       ;        BL > 0 -> Allocate (a new) DMA buffer
3309                            <1>       ;        ECX = DMA Buffer Size in bytes (<=128KB)
3310                            <1>       ;        EDX = Virtual Address of DMA buffer
3311                            <1>       ;
3312                            <1>       ;     BH = 1 -> Initialize (Start) DMA service
3313                            <1>       ;          BL, bit 0 to 3 = Channel Number (0 to 7)
3314                            <1>       ;          BL, bit 7 = Auto Initialized Mode
3315                            <1>       ;                  (If bit 7 is set)
3316                            <1>       ;            bit 6 = Record (read) mode (0= playback)
3317                            <1>       ;          ECX = byte count (0 = use dma buffer size)
3318                            <1>       ;          EDX = physical buffer address
3319                            <1>       ;               (0 = use dma buffer -start- address)
3320                            <1>       ;
3321                            <1>       ;     BH = 2 -> Get Current DMA Buffer Offset
3322                            <1>       ;          BL = DMA channel number
3323                            <1>       ;
3324                            <1>       ;     BH = 3 -> Get Current DMA count down value
3325                            <1>       ;          BL = DMA channel number (0 tO 7)
3326                            <1>       ;
3327                            <1>       ;     BH = 4 -> Get Current DMA channel (in progress)
3328                            <1>       ;
3329                            <1>       ;     BH = 5 -> Get System's Default DMA Buffer Address
3330                            <1>       ;
3331                            <1>       ;     BH = 6 -> Get Current DMA Buffer Address
3332                            <1>       ;
3333                            <1>       ;     BH = 7 -> Stop DMA service
3334                            <1>       ;
3335                            <1>       ;
3336                            <1>       ; Outputs:
3337                            <1>       ;
3338                            <1>       ;     For BH = 0 ; Allocate DMA buffer
3339                            <1>       ;         EAX = Physical address of DMA buffer
3340                            <1>       ;         ECX = Allocated buffer size in bytes
3341                            <1>       ;             - page count * 4096 -
3342                            <1>       ;             (may be bigger than requested)
3343                            <1>       ;         If BL input > 0,
3344                            <1>       ;            'sysalloc:' system call will be used with
3345                            <1>       ;            EBX (for 'sysalloc') = EDX (for 'sysdma')
```

```
3346                                 <1>       ;            ECX is same, byte count (buffer size)
3347                                 <1>       ;            EDX = 1024*1024*16 ; 16 MB upper limit
3348                                 <1>       ;         If BL input = 0,
3349                                 <1>       ;            Default DMA buffer (SB16 buffer) will be
3350                                 <1>       ;            checked and if it is free, it's address
3351                                 <1>       ;            will be returned in EAX and it's size
3352                                 <1>       ;            will be returned in ECX (as 65536)
3353                                 <1>       ;
3354                                 <1>       ;         If CF = 1, error code is in EAX
3355                                 <1>       ;            EAX = -1 ; DMA buffer allocation error!
3356                                 <1>       ;            EAX = 11 ; 'Permission Denied' error !
3357                                 <1>       ;
3358                                 <1>       ;            Note: 'sysalloc' error return method
3359                                 <1>       ;                  will be applied if BL input > 0 !
3360                                 <1>       ;
3361                                 <1>       ;       For BH = 1 ; Initialize (Start) DMA
3362                                 <1>       ;           EAX = 0 (Successful)
3363                                 <1>       ;           If CF = 1, error code is in EAX
3364                                 <1>       ;
3365                                 <1>       ;       For BH = 2 ; Get Current DMA Buffer Offset
3366                                 <1>       ;           EAX = DMA Buffer Offset (in bytes)
3367                                 <1>       ;           ;
3368                                 <1>       ;            AX = DMA buffer offset
3369                                 <1>       ;           EAX bits 16 to 23 = Page register value
3370                                 <1>       ;
3371                                 <1>       ;       For BH = 3 ; Get Current DMA count down value
3372                                 <1>       ;           EAX = Count down value (remain bytes)
3373                                 <1>       ;
3374                                 <1>       ;       For BH = 4 ; Get Current DMA channel (in progress)
3375                                 <1>       ;           EAX = DMA channel number (0 to 7)
3376                                 <1>       ;             AH = 0 if the owner is the caller process
3377                                 <1>       ;             AH > 0 if the dma channel is in use by
3378                                 <1>       ;                    another user/process
3379                                 <1>       ;               EAX = -1 (0FFFFFFFFh)
3380                                 <1>       ;                    if DMA service is not in use
3381                                 <1>       ;                    (stopped or not initialized/started)
3382                                 <1>       ;
3383                                 <1>       ;       For BH = 5 ; Get System's Default DMA Buff Addr
3384                                 <1>       ;           EAX = Default DMA Buffer Address (Physical)
3385                                 <1>       ;             = offset 'sb16_dma_buffer:'
3386                                 <1>       ;           ECX = Buffer size
3387                                 <1>       ;             = 65536
3388                                 <1>       ;
3389                                 <1>       ;       For BH = 6 ; Get Current DMA Buffer Address
3390                                 <1>       ;           EAX = Current DMA buffer address (Physical)
3391                                 <1>       ;           ECX = Current DMA buffer size (setting value)
3392                                 <1>       ;           Note: These values are for current dma channel
3393                                 <1>       ;                settings for the user/process
3394                                 <1>       ;           ** For now (for current TRDOS 386 version)
3395                                 <1>       ;             only one user/process can use only one
3396                                 <1>       ;             dma channel & one dma buffer at same time
3397                                 <1>       ;             (no multi tasking on DMA service) !!! **
3398                                 <1>       ;           (Once, current DMA user must stop it's own DMA
3399                                 <1>       ;            DMA service, than another user/program
3400                                 <1>       ;            can use DMA service with same dma channel
3401                                 <1>       ;            or with another DMA channel.)
3402                                 <1>       ;
3403                                 <1>       ;       For BH = 7 ; Stop DMA service (for current user
3404                                 <1>       ;           and current DMA channel)
3405                                 <1>       ;           EAX = 0 ; successful
3406                                 <1>       ;           CF = 1 & EAX > 0 (= -1) -> Error
3407                                 <1>
3408 00013CEB 80FF07               <1>       cmp   bh, 7
3409 00013CEE 7612                 <1>       jna   short sysdma_0
3410                                 <1>
3411                                 <1> sysdma_err:
3412 00013CF0 31C0                 <1>       xor   eax, eax
3413 00013CF2 48                   <1>       dec   eax ; -1
3414                                 <1> sysdma_perm_err:
3415 00013CF3 A3[64030300]         <1>       mov   [u.r0], eax
3416 00013CF8 A3[C8030300]         <1>       mov   [u.error], eax ; DMA service error !
3417 00013CFD E9DA9BFFFF           <1>       jmp   error
3418                                 <1>
3419                                 <1> sysdma_0:
3420 00013D02 08FF                 <1>       or    bh, bh
3421 00013D04 0F85BA000000         <1>       jnz   sysdma_1
3422                                 <1>
3423 00013D0A 20DB                 <1>       and   bl, bl
3424 00013D0C 7416                 <1>       jz    short sysdma_01
3425                                 <1>
3426                                 <1>       ; redirect system call to 'sysalloc'
3427 00013D0E 89D3                 <1>       mov   ebx, edx ; virtual address of DMA buffer
3428                                 <1>       ;ecx = Buffer size in bytes
3429                                 <1>       ; DMA buffer address <= 16MB upper limit
3430 00013D10 BA00000001           <1>       mov   edx, 1024*1024*16 ; 16MB limit for DMA buff
3431                                 <1>
3432 00013D15 C705[84A00100]FFFF- <1>       mov   dword [dma_addr], 0FFFFFFFFh ; -1
3432 00013D1D FFFF                 <1>
3433                                 <1>
3434 00013D1F E9A3E5FFFF           <1>       jmp   sysalloc
3435                                 <1>
3436                                 <1> sysdma_01:
3437 00013D24 B8[00000200]         <1>       mov   eax, sb16_dma_buffer
3438                                 <1>
3439 00013D29 803D[419C0100]01     <1>       cmp   byte [audio_device], 1
3440 00013D30 722A                 <1>       jb    short sysdma_03
3441                                 <1>
3442 00013D32 3B05[609C0100]       <1>       cmp   eax, [audio_dma_buff]
3443 00013D38 7507                 <1>       jne   short sysdma_02
3444                                 <1>
3445                                 <1> sysdma_0_err:
3446 00013D3A B80B000000           <1>       mov   eax, ERR_PERM_DENIED
3447 00013D3F EBB2                 <1>       jmp   short sysdma_perm_err
3448                                 <1>
3449                                 <1> sysdma_02:
```

```
3450                            <1>       ; Only one user is permitted for audio/dma functions
3451                            <1>
3452 00013D41 833D[609C0100]00  <1>       cmp    dword [audio_dma_buff], 0
3453 00013D48 7612              <1>       jna    short sysdma_03
3454                            <1>
3455 00013D4A 8A1D[699C0100]    <1>       mov    bl, [audio_user]
3456 00013D50 08DB              <1>       or     bl, bl
3457 00013D52 7408              <1>       jz     short sysdma_03
3458                            <1>
3459 00013D54 3A1D[B3030300]    <1>       cmp    bl, [u.uno]
3460 00013D5A 75DE              <1>       jne    short sysdma_0_err
3461                            <1>
3462                            <1> sysdma_03:
3463 00013D5C 8A1D[81A00100]    <1>       mov    bl, [dma_user]
3464 00013D62 20DB              <1>       and    bl, bl
3465 00013D64 750E              <1>       jnz    short sysdma_04
3466                            <1>
3467 00013D66 8A1D[B3030300]    <1>       mov    bl, [u.uno]
3468 00013D6C 881D[81A00100]    <1>       mov    [dma_user], bl
3469                            <1>
3470 00013D72 EB15              <1>       jmp    short sysdma_05
3471                            <1>
3472                            <1> sysdma_04:
3473 00013D74 8B35[84A00100]    <1>       mov    esi, [dma_addr]
3474 00013D7A 21F6              <1>       and    esi, esi
3475 00013D7C 740B              <1>       jz     short sysdma_05
3476                            <1>
3477 00013D7E 46                <1>       inc    esi ; -1 -> 0
3478 00013D7F 7408              <1>       jz     short sysdma_05
3479                            <1>
3480 00013D81 3A1D[B3030300]    <1>       cmp    bl, [u.uno]
3481 00013D87 75B1              <1>       jne    short sysdma_0_err
3482                            <1>
3483                            <1> sysdma_05:
3484                            <1>       ; edx = virtual address (user's buffer address)
3485                            <1>       ;
3486 00013D89 81F900000100      <1>       cmp    ecx, 65536   ; byte count (buffer size)
3487 00013D8F 0F875BFFFFFF      <1>       ja     sysdma_err
3488                            <1>       ;
3489 00013D95 81C1FF0F0000      <1>       add    ecx, PAGE_SIZE-1 ; 4095
3490 00013D9B 6681E100F0        <1>       and    cx, ~PAGE_OFF ; not 4095
3491                            <1>       ;cmp   ecx, 65536
3492                            <1>       ;ja    sysdma_err ;
3493 00013DA0 51                <1>       push   ecx    ; buffer size (allocated pages * 4096)
3494 00013DA1 50                <1>       push   eax    ; offset sb16_dma_buffer
3495 00013DA2 89D3              <1>       mov    ebx, edx
3496 00013DA4 C1E90C            <1>       shr    ecx, 12 ; byte count to page count
3497                            <1>       ; eax = physical address of (audio) dma buffer
3498                            <1>       ; ebx = virtual address of (audio) dma buffer (user's pgdir)
3499                            <1>       ; ecx = page count (>0)
3500 00013DA7 E84E29FFFF        <1>       call   direct_memory_access
3501 00013DAC 58                <1>       pop    eax
3502 00013DAD 59                <1>       pop    ecx
3503 00013DAE 0F823CFFFFFF      <1>       jc     sysdma_err
3504                            <1>
3505 00013DB4 A3[84A00100]      <1>       mov    [dma_addr], eax
3506 00013DB9 890D[88A00100]    <1>       mov    [dma_size], ecx ; dma buffer size (in bytes)
3507                            <1>
3508                            <1>       ;mov   [u.r0], eax ; DMA Buffer Address (Physical)
3509                            <1>
3510                            <1>       ;mov   ebp, [u.usp]  ; ebp points to user's registers
3511                            <1>       ;mov   [ebp+24], ecx ; return to user with ecx value
3512                            <1>
3513                            <1>       ;jmp   sysret
3514                            <1>
3515                            <1>       ; 28/08/2017
3516 00013DBF E9C4000000        <1>       jmp    sysdma_51
3517                            <1>
3518                            <1> sysdma_1:
3519 00013DC4 80FF01            <1>       cmp    bh, 1
3520 00013DC7 0F87A6000000      <1>       ja     sysdma_5
3521                            <1>
3522 00013DCD F6C340            <1>       test   bl, 40h      ; record (read) mode -BL, bit 6-
3523 00013DD0 0F851AFFFFFF      <1>       jnz    sysdma_err ; not ready yet!
3524                            <1>
3525 00013DD6 A1[84A00100]      <1>       mov    eax, [dma_addr] ; physical address of dma buffer
3526 00013DDB 21C0              <1>       and    eax, eax
3527 00013DDD 0F840DFFFFFF      <1>       jz     sysdma_err
3528                            <1>
3529 00013DE3 09D2              <1>       or     edx, edx
3530 00013DE5 7504              <1>       jnz    short sysdma_11
3531                            <1>
3532 00013DE7 89C2              <1>       mov    edx, eax
3533 00013DE9 EB08              <1>       jmp    short sysdma_12
3534                            <1> sysdma_11:
3535 00013DEB 39C2              <1>       cmp    edx, eax
3536 00013DED 0F82FDFEFFFF      <1>       jb     sysdma_err
3537                            <1> sysdma_12:
3538 00013DF3 21C9              <1>       and    ecx, ecx
3539 00013DF5 7508              <1>       jnz    short sysdma_13
3540                            <1>
3541 00013DF7 8B0D[88A00100]    <1>       mov    ecx, [dma_size]
3542 00013DFD EB0C              <1>       jmp    short sysdma_14
3543                            <1> sysdma_13:
3544 00013DFF 3B0D[88A00100]    <1>       cmp    ecx, [dma_size]
3545 00013E05 0F87E5FEFFFF      <1>       ja     sysdma_err
3546                            <1> sysdma_14:
3547 00013E0B 89C6              <1>       mov    esi, eax
3548 00013E0D 0335[88A00100]    <1>       add    esi, [dma_size]
3549                            <1>
3550 00013E13 89D0              <1>       mov    eax, edx
3551 00013E15 01C8              <1>       add    eax, ecx
3552 00013E17 0F82D3FEFFFF      <1>       jc     sysdma_err ; 02/09/2017
3553                            <1>
3554 00013E1D 39F0              <1>       cmp    eax, esi
```

```
3555 00013E1F 0F87CBFEFFFF        <1>       ja    sysdma_err
3556                              <1>
3557 00013E25 8B3D[609C0100]      <1>       mov   edi, [audio_dma_buff]
3558 00013E2B 8B35[84A00100]      <1>       mov   esi, [dma_addr]
3559                              <1>
3560 00013E31 09FF                <1>       or    edi, edi
3561 00013E33 7424                <1>       jz    short sysdma_16
3562                              <1>
3563 00013E35 803D[419C0100]01    <1>       cmp   byte [audio_device], 1
3564 00013E3C 7208                <1>       jb    short sysdma_15
3565                              <1>
3566                              <1>       ; Sound Blaster 16
3567 00013E3E 39FE                <1>       cmp   esi, edi
3568 00013E40 0F84F4FEFFFF        <1>       je    sysdma_0_err ; permmission denied !
3569                              <1>
3570                              <1> sysdma_15:
3571 00013E46 C605[83A00100]48    <1>       mov   byte [dma_mode], 48h ; single mode playback
3572                              <1>
3573 00013E4D F6C380              <1>       test  bl, 80h ; DMA mode - BL, bit 7, auto init -
3574 00013E50 7407                <1>       jz    short sysdma_16
3575                              <1>       ; Auto initialized playback (write) mode
3576 00013E52 8005[83A00100]10    <1>       add   byte [dma_mode], 10h ; = 58h
3577                              <1> sysdma_16:
3578 00013E59 80E307              <1>       and   bl, 07h
3579 00013E5C 881D[82A00100]      <1>       mov   [dma_channel], bl
3580 00013E62 8915[8CA00100]      <1>       mov   [dma_start], edx
3581 00013E68 890D[90A00100]      <1>       mov   [dma_count], ecx
3582                              <1>
3583                              <1>       ; 28/08/2017
3584                              <1>       ;call  dma_init
3585                              <1>       ;jmp   sysret
3586 00013E6E E94B010000          <1>       jmp   dma_init
3587                              <1>
3588                              <1> sysdma_5:
3589 00013E73 80FF05              <1>       cmp   bh, 5
3590 00013E76 7223                <1>       jb    short sysdma_3
3591 00013E78 0F87CE000000        <1>       ja    sysdma_6
3592                              <1>
3593                              <1>       ; Get the system's default dma buffer addr and size
3594 00013E7E B8[00000200]        <1>       mov   eax, sb16_dma_buffer
3595 00013E83 B900000100          <1>       mov   ecx, 65536 ; Buffer size in bytes
3596                              <1>
3597                              <1> sysdma_51:
3598                              <1>       ; 0 = there is not a dma buffer (in use or available)
3599 00013E88 A3[64030300]        <1>       mov   [u.r0], eax
3600                              <1>
3601 00013E8D 8B2D[60030300]      <1>       mov   ebp, [u.usp]  ; ebp points to user's registers
3602 00013E93 894D18              <1>       mov   [ebp+24], ecx ; return to user with ecx value
3603                              <1>
3604 00013E96 E9619AFFFF          <1>       jmp   sysret
3605                              <1>
3606                              <1> sysdma_3:
3607 00013E9B 80FF03              <1>       cmp   bh, 3
3608 00013E9E 7231                <1>       jb    short sysdma_2
3609 00013EA0 776B                <1>       ja    short sysdma_4
3610                              <1>
3611                              <1>       ; Get current dma count down value (remain bytes)
3612                              <1>       ; 28/08/2017
3613 00013EA2 0FB635[82A00100]    <1>       movzx esi, byte [dma_channel]
3614 00013EA9 0FB696[BE490100]    <1>       movzx edx, byte [dma_flip+esi]
3615 00013EB0 EE                  <1>       out   dx, al              ; flip-flop clear
3616 00013EB1 8A96[9E490100]      <1>       mov   dl, [dma_cnt+esi] ; dma count register addr
3617 00013EB7 EC                  <1>       in    al, dx
3618 00013EB8 0FB6D8              <1>       movzx ebx, al
3619 00013EBB EC                  <1>       in    al, dx
3620 00013EBC 88C7                <1>       mov    bh, al
3621                              <1>
3622 00013EBE 6683FE04            <1>       cmp   si, 4 ; channel number ?
3623 00013EC2 7202                <1>       jb    short sysdma_31 ; 8 bit dma channel
3624                              <1>
3625 00013EC4 D1E3                <1>       shl   ebx, 1 ; word count to byte count
3626                              <1>
3627                              <1> sysdma_31:
3628 00013EC6 891D[64030300]      <1>       mov   [u.r0], ebx
3629                              <1>
3630 00013ECC E92B9AFFFF          <1>       jmp   sysret
3631                              <1>
3632                              <1> sysdma_2:
3633                              <1>       ; Get current dma buffer offset (& page)
3634                              <1>       ; 28/08/2017
3635 00013ED1 0FB635[82A00100]    <1>       movzx esi, byte [dma_channel]
3636 00013ED8 0FB696[BE490100]    <1>       movzx edx, byte [dma_flip+esi]
3637 00013EDF EE                  <1>       out   dx, al              ; flip-flop clear
3638 00013EE0 8A96[96490100]      <1>       mov   dl, [dma_adr+esi]
3639 00013EE6 EC                  <1>       in    al, dx              ; get dma position
3640 00013EE7 0FB6D8              <1>       movzx ebx, al
3641 00013EEA EC                  <1>       in    al, dx
3642 00013EEB 88C7                <1>       mov   bh, al
3643                              <1>
3644 00013EED 6683FE04            <1>       cmp   si, 4 ; channel number ?
3645 00013EF1 7202                <1>       jb    short sysdma_21 ; 8 bit dma channel
3646                              <1>
3647 00013EF3 D1E3                <1>       shl   ebx, 1 ; word offset to byte offset
3648                              <1>
3649                              <1> sysdma_21:
3650 00013EF5 891D[64030300]      <1>       mov   [u.r0], ebx
3651                              <1>
3652 00013EFB 8A96[A6490100]      <1>       mov   dl, [dma_page+esi]
3653 00013F01 EC                  <1>       in    al, dx              ; get dma page
3654                              <1>
3655                              <1>       ;add  [u.ro+2], al
3656 00013F02 0805[66030300]      <1>       or    [u.r0+2], al
3657                              <1>
3658 00013F08 E9EF99FFFF          <1>       jmp   sysret
3659                              <1>
```

```
3660                                <1> sysdma_4:
3661                                <1>       ; Get current DMA channel number
3662                                <1>       ; 28/08/2017
3663 00013F0D 8A25[81A00100]       <1>       mov    ah, [dma_user]
3664 00013F13 20E4                 <1>       and    ah, ah
3665 00013F15 750F                 <1>       jnz    short sysdma_42
3666                                <1>
3667                                <1> sysdma_41:
3668                                <1>       ; Not a valid dma channel (in use)
3669 00013F17 C705[64030300]FFFF-  <1>       mov    dword [u.r0], -1 ; 0FFFFFFFFh
3669 00013F1F FFFF                 <1>
3670 00013F21 E9D699FFFF           <1>       jmp    sysret
3671                                <1>
3672                                <1> sysdma_42:
3673 00013F26 8B35[84A00100]       <1>       mov    esi, [dma_addr]
3674 00013F2C 21F6                 <1>       and    esi, esi
3675 00013F2E 74E7                 <1>       jz     short sysdma_41
3676                                <1>
3677 00013F30 46                   <1>       inc    esi ; -1 -> 0
3678 00013F31 74E4                 <1>       jz     short sysdma_41
3679                                <1>
3680 00013F33 A0[82A00100]         <1>       mov    al, [dma_channel]
3681                                <1>
3682 00013F38 3A25[B3030300]       <1>       cmp    ah, [u.uno]
3683 00013F3E 7502                 <1>       jne    short sysdma_43
3684                                <1>
3685 00013F40 30E4                 <1>       xor    ah, ah ; DMA channel in use by current user
3686                                <1>
3687                                <1> sysdma_43:
3688 00013F42 A3[64030300]         <1>       mov    [u.r0], eax ; AL = dma channel number
3689                                <1>                           ; AH > 0 if the the channel
3690                                <1>                           ; in use by another user/process
3691 00013F47 E9B099FFFF           <1>       jmp    sysret
3692                                <1>
3693                                <1> sysdma_6:
3694 00013F4C 80FF06               <1>       cmp    bh, 6
3695 00013F4F 7710                 <1>       ja     short sysdma_7
3696                                <1>
3697                                <1>       ; 28/08/2017
3698                                <1>       ; Get current DMA buffer addr and size
3699 00013F51 A1[84A00100]         <1>       mov    eax, [dma_addr] ; dma buffer address
3700 00013F56 8B0D[88A00100]       <1>       mov    ecx, [dma_size] ; dma buffer size (in bytes)
3701                                <1>
3702 00013F5C E927FFFFFF           <1>       jmp    sysdma_51
3703                                <1>
3704                                <1> sysdma_7:
3705                                <1>       ; DMA service STOP
3706 00013F61 A0[B3030300]         <1>       mov    al, [u.uno]
3707 00013F66 3A05[81A00100]       <1>       cmp    al, [dma_user]
3708 00013F6C 751D                 <1>       jne    short sysdma_72
3709                                <1>
3710 00013F6E 28C0                 <1>       sub    al, al ; 0
3711                                <1>
3712 00013F70 A2[81A00100]         <1>       mov    [dma_user], al ; clear user
3713                                <1>
3714 00013F75 8605[83A00100]       <1>       xchg   al, [dma_mode]
3715 00013F7B 20C0                 <1>       and    al, al
3716                                <1>       ;jz    short sysdma_err
3717 00013F7D 7527                 <1>       jnz    short sysdma_73
3718                                <1>
3719                                <1> sysdma_71:
3720 00013F7F 31C0                 <1>       xor    eax, eax
3721 00013F81 A3[64030300]         <1>       mov    [u.r0], eax; 0
3722 00013F86 E97199FFFF           <1>       jmp    sysret
3723                                <1>
3724                                <1> sysdma_72:
3725                                <1>       ; 28/08/2017
3726 00013F8B 803D[81A00100]00     <1>       cmp    byte [dma_user], 0
3727 00013F92 76EB                 <1>       jna    short sysdma_71 ; Nothing to do !
3728                                <1>
3729 00013F94 833D[84A00100]00     <1>       cmp    dword [dma_addr], 0
3730 00013F9B 0F8799FDFFFF         <1>       ja     sysdma_0_err
3731                                <1>
3732 00013FA1 A2[81A00100]         <1>       mov    [dma_user], al ; reset to current user
3733                                <1>
3734                                <1> sysdma_73:
3735                                <1>       ; 28/08/2017
3736 00013FA6 0FB635[82A00100]     <1>       movzx  esi, byte [dma_channel]
3737 00013FAD 0FB696[AE490100]     <1>       movzx  edx, byte [dma_mask+esi]
3738 00013FB4 A0[82A00100]         <1>       mov    al, [dma_channel]
3739 00013FB9 0C04                 <1>       or     al, 4
3740 00013FBB EE                   <1>       out    dx, al
3741                                <1>
3742 00013FBC EBC1                 <1>       jmp    short sysdma_71
3743                                <1>
3744                                <1> dma_init:
3745                                <1>       ; 28/08/2017
3746                                <1>       ; 20/08/2017
3747                                <1>       ; DMA initialization
3748                                <1>       ; 14/08/2017
3749                                <1>       ; 03/08/2017, 06/08/2017, 08/08/2017
3750                                <1>       ; 02/07/2017, 13/07/2017, 16/07/2017, 30/07/2017
3751                                <1>       ; (Derived from 'DMA_INIT' procedure in SB16MOD.ASM)
3752                                <1>       ; Modified for TRDOS 386 DMA buffer allocation & initialization !
3753                                <1>
3754 00013FBE 8B1D[8CA00100]       <1>       mov    ebx, [dma_start]
3755 00013FC4 8B0D[90A00100]       <1>       mov    ecx, [dma_count]
3756                                <1>
3757 00013FCA 0FB635[82A00100]     <1>       movzx  esi, byte [dma_channel]
3758                                <1>
3759 00013FD1 6683FE04             <1>       cmp    si, 4
3760 00013FD5 7205                 <1>       jb     short gdmi1
3761                                <1>       ; 08/08/2017
3762 00013FD7 66D1E9               <1>       shr    cx, 1 ; word count
3763 00013FDA D1EB                 <1>       shr    ebx, 1 ; convert byte offset to word offset
```

```
3764                            <1> gdmi1:
3765                            <1>       ;mov   [dma_poff], bx ; 08/08/2017
3766 00013FDC 6649              <1>       dec    cx                 ; dma size = block size - 1
3767                            <1>
3768 00013FDE 0FB696[AE490100]  <1>       movzx  edx, byte [dma_mask+esi] ; 30/07/2017
3769 00013FE5 A0[82A00100]      <1>       mov    al, [dma_channel]
3770 00013FEA 0C04              <1>       or     al, 4
3771 00013FEC EE                <1>       out    dx, al             ; dma channel mask
3772                            <1>
3773 00013FED 30C0              <1>       xor    al, al ; 0 ; any value ! 08/08/2017
3774 00013FEF 8A96[BE490100]    <1>       mov    dl, [dma_flip+esi]
3775 00013FF5 EE                <1>       out    dx, al             ; flip-flop clear
3776                            <1>
3777 00013FF6 8A96[B6490100]    <1>       mov    dl, [dma_mod+esi]
3778 00013FFC A0[82A00100]      <1>       mov    al, [dma_channel]  ; 13/07/2017
3779 00014001 2403              <1>       and    al, 3
3780                            <1>       ; 08/08/2017
3781 00014003 0A05[83A00100]    <1>       or     al, [dma_mode] ; 58h    ; dma mode for SB16
3782 00014009 EE                <1>       out    dx, al
3783                            <1>
3784 0001400A 8A96[96490100]    <1>       mov    dl, [dma_adr+esi]
3785 00014010 88D8              <1>       mov    al, bl
3786 00014012 EE                <1>       out    dx, al             ; offset low
3787                            <1>
3788 00014013 88F8              <1>       mov    al, bh
3789 00014015 EE                <1>       out    dx, al             ; offset high
3790                            <1>
3791 00014016 8A96[9E490100]    <1>       mov    dl, [dma_cnt+esi]
3792 0001401C 88C8              <1>       mov    al, cl
3793 0001401E EE                <1>       out    dx, al             ; size low
3794                            <1>
3795 0001401F 88E8              <1>       mov    al, ch
3796 00014021 EE                <1>       out    dx, al             ; size high
3797                            <1>
3798 00014022 8A96[A6490100]    <1>       mov    dl, [dma_page+esi]
3799                            <1>       ; 14/08/2017
3800 00014028 6683FE04          <1>       cmp    si, 4
3801 0001402C 7305              <1>       jnb    short gdmi2
3802 0001402E C1EB10            <1>       shr    ebx, 16
3803 00014031 EB06              <1>       jmp    short gdmi3
3804                            <1> gdmi2:
3805                            <1>       ; 09/08/2017
3806 00014033 C1EB0F            <1>       shr    ebx, 15       ; complete 16 bit shift
3807 00014036 80E3FE            <1>       and    bl, 0FEh ; clear bit 0 (not necessary)
3808                            <1> gdmi3:
3809 00014039 88D8              <1>       mov    al, bl
3810 0001403B EE                <1>       out    dx, al             ; page
3811                            <1>
3812 0001403C 8A96[AE490100]    <1>       mov    dl, [dma_mask+esi]
3813 00014042 A0[82A00100]      <1>       mov    al, [dma_channel]  ; 13/07/2017
3814 00014047 2403              <1>       and    al, 3
3815 00014049 EE                <1>       out    dx, al             ; dma channel unmask
3816                            <1>
3817                            <1>       ;retn
3818                            <1>       ; 28/08/2017
3819 0001404A E9AD98FFFF        <1>       jmp    sysret
3820                            <1>
3821                            <1> otty:
3822                            <1> sret:
3823                            <1> ocvt:
3824                            <1> ctty:
3825                            <1> cret:
3826                            <1> ccvt:
3827                            <1> rtty:
3828                            <1> wtty:
3829                            <1> rmem:
3830                            <1> wmem:
3831                            <1> rfd:
3832                            <1> rhd:
3833                            <1> wfd:
3834                            <1> whd:
3835                            <1> rlpt:
3836                            <1> wlpt:
3837                            <1> rcvt:
3838                            <1> xmtt:
3839 0001404F C3                 <1>       retn
3087                                %include 'trdosk9.s' ; 04/01/2016
   1                            <1> ; ****************************************************************************
   2                            <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.3) - INITIALIZED DATA : trdosk9.s
   3                            <1> ; ----------------------------------------------------------------------------
   4                            <1> ; Last Update: 15/02/2021
   5                            <1> ; ----------------------------------------------------------------------------
   6                            <1> ; Beginning: 04/01/2016
   7                            <1> ; ----------------------------------------------------------------------------
   8                            <1> ; ----------------------------------------------------------------------------
   9                            <1> ; Assembler: NASM version 2.15 (trdos386.s)
  10                            <1> ; ----------------------------------------------------------------------------
  11                            <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
  12                            <1> ; TRDOS2.ASM (09/11/2011)
  13                            <1> ; ****************************************************************************
  14                            <1> ; DRV_INIT.ASM [26/09/2009] Last Update: 07/08/2011
  15                            <1> ; MAINPROG.ASM [17/01/2004] Last Update: 09/11/2011
  16                            <1> ; CMD_INTR.ASM [29/01/2005] Last Update: 09/11/2011
  17                            <1> ; FILE.ASM [29/10/2009] Last Update: 09/10/2011
  18                            <1>
  19                            <1> ; 12/02/2016
  20                            <1> Last_DOS_DiskNo:
  21 00014050 01                <1>             db 1 ; A: = 0 & B: = 1
  22                            <1>
  23                            <1> Restore_CDIR:
  24 00014051 FF                <1>             db 0FFh ; Initial value -> any number except 0
  25                            <1>
  26                            <1> msg_CRLF_temp:
  27 00014052 070D0A00          <1>             db 07h, 0Dh, 0Ah, 0
  28                            <1>
```

```
                            <1> Magic_Bytes:
00014056 04                 <1>             db 4
00014057 01                 <1>             db 1
                            <1> mainprog_Version:
00014058 07                 <1>             db 7
00014059 5B5452444F535D204D- <1>             db "[TRDOS] Main Program v2.0.150221"
00014062 61696E2050726F6772- <1>
0001406B 616D2076322E302E31- <1>
00014074 3530323231         <1>
00014079 0D0A               <1>             db 0Dh, 0Ah
0001407B 286329204572646F67- <1>             db "(c) Erdogan Tan 2005-2021"
00014084 616E2054616E203230- <1>
0001408D 30352D32303231      <1>
00014094 0D0A00             <1>             db 0Dh, 0Ah, 0
                            <1>
                            <1> MainProgCfgFile: ; 14/04/2016
00014097 4D41494E50524F472E- <1>             db "MAINPROG.CFG", 0
000140A0 43464700           <1>
                            <1>
                            <1> TRDOSPromptLabel:
000140A4 5452444F53         <1>             db "TRDOS"
000140A9 00                 <1>             db 0
000140AA 00<rept>           <1>                times 5 db 0
000140AF 00                 <1>             db 0
                            <1>
                            <1> ; INTERNAL COMMANDS
                            <1> Command_List:
000140B0 44495200           <1> Cmd_Dir:    db "DIR", 0
000140B4 434400             <1> Cmd_Cd:            db "CD", 0
000140B7 433A00             <1> Cmd_Drive:  db "C:", 0
000140BA 56455200           <1> Cmd_Ver:    db "VER", 0
000140BE 4558495400         <1> Cmd_Exit:   db "EXIT", 0
000140C3 50524F4D505400     <1> Cmd_Prompt: db "PROMPT", 0
000140CA 564F4C554D4500     <1> Cmd_Volume: db "VOLUME", 0
000140D1 4C4F4E474E414D4500 <1> Cmd_LongName:      db "LONGNAME", 0
000140DA 4441544500         <1> Cmd_Date:   db "DATE", 0
000140DF 54494D4500         <1> Cmd_Time:   db "TIME", 0
000140E4 52554E00           <1> Cmd_Run:    db "RUN", 0
000140E8 53455400           <1> Cmd_Set:    db "SET", 0
000140EC 434C5300           <1> Cmd_Cls:    db "CLS", 0
000140F0 53484F5700         <1> Cmd_Show:   db "SHOW", 0
000140F5 44454C00           <1> Cmd_Del:    db "DEL", 0
000140F9 41545452494200     <1> Cmd_Attrib: db "ATTRIB", 0
00014100 52454E414D4500     <1> Cmd_Rename: db "RENAME", 0
00014107 524D44495200       <1> Cmd_Rmdir:  db "RMDIR", 0
0001410D 4D4B44495200       <1> Cmd_Mkdir:  db "MKDIR", 0
00014113 434F505900         <1> Cmd_Copy:   db "COPY", 0
00014118 4D4F564500         <1> Cmd_Move:   db "MOVE", 0
0001411D 5041544800         <1> Cmd_Path:   db "PATH", 0
00014122 4D454D00           <1> Cmd_Mem:    db "MEM", 0
00014126 00                 <1>             db 0
00014127 46494E4400         <1> Cmd_Find:   db "FIND", 0
0001412C 4543484F00         <1> Cmd_Echo:   db "ECHO", 0
00014131 2A00               <1> Cmd_Remark: db "*", 0
00014133 3F00               <1> Cmd_Help:   db "?", 0
00014135 44455649434500     <1> Cmd_Device: db "DEVICE", 0
0001413C 4445564C49535400   <1> Cmd_DevList:db "DEVLIST", 0
00014144 434844495200       <1> Cmd_Chdir:  db "CHDIR", 0
0001414A 4245455000         <1> Cmd_Beep:   db "BEEP", 0
                            <1>
0001414F 00                 <1>             db 0
                            <1>
                            <1> ; 15/02/2016 (FILE.ASM, 09/10/2011)
                            <1> invalid_fname_chars:
00014150 222728292A2B2C2F   <1>             db 22h, 27h, 28h, 29h, 2Ah, 2Bh, 2Ch, 2Fh
00014158 3A3B3C3D3E3F40     <1>             db 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh, 40h
0001415F 5B5C5D5E60         <1>             db 5Bh, 5Ch, 5Dh, 5Eh, 60h
                            <1> sizeInvFnChars  equ ($ - invalid_fname_chars)
                            <1> ;
                            <1>
                            <1> Msg_Enter_Date:
00014164 456E746572206E6577- <1>                db 'Enter new date (dd-mm-yy): '
0001416D 206461746520286464- <1>
00014176 2D6D6D2D7979293A20  <1>
0001417F 00                 <1>                 db 0
                            <1> Msg_Show_Date:
00014180 43757272656E742064- <1>                db   'Current date is '
00014189 61746520697320      <1>
00014190 30                 <1> Day:            db   '0'
00014191 30                 <1>             db   '0'
00014192 2F                 <1>                 db   '/'
00014193 30                 <1> Month:          db   '0'
00014194 30                 <1>             db   '0'
00014195 2F                 <1>                 db   '/'
00014196 30                 <1> Century:        db   '0'
00014197 30                 <1>                 db   '0'
00014198 30                 <1> Year:           db   '0'
00014199 30                 <1>             db   '0'
0001419A 0D0A00             <1>                 db   0Dh, 0Ah, 0
                            <1>
                            <1> Msg_Enter_Time:
0001419D 456E746572206E6577- <1>             db 'Enter new time: '
000141A6 2074696D653A20      <1>
000141AD 00                 <1>             db 0
                            <1> Msg_Show_Time:
000141AE 43757272656E742074- <1>             db   'Current time is '
000141B7 696D6520697320      <1>
000141BE 30                 <1> Hour:           db   '0'
000141BF 30                 <1>             db   '0'
000141C0 3A                 <1>             db   ':'
000141C1 30                 <1> Minute:         db   '0'
000141C2 30                 <1>             db   '0'
000141C3 3A                 <1>             db   ':'
000141C4 30                 <1> Second:         db   '0'
000141C5 30                 <1>             db   '0'
```

```
000141C6 0D0A00              <1>             db   0Dh, 0Ah, 0
                             <1>
                             <1> ;VolSize_Unit1:   dd 0
                             <1> ;VolSize_Unit2:   dd 0
                             <1>
                             <1> VolSize_KiloBytes:
000141C9 206B696C6F62797465- <1>             db " kilobytes", 0Dh, 0Ah, 0
000141D2 730D0A00            <1>
                             <1> VolSize_Bytes:
000141D6 2062797465730D0A00  <1>             db " bytes", 0Dh, 0Ah, 0
                             <1> Volume_in_drive:
000141DF 0D0A                <1>             db 0Dh, 0Ah
                             <1> Vol_FS_Name:
000141E1 54522046533120      <1>             db "TR FS1 "
000141E8 566F6C756D6520696E- <1>             db "Volume in drive "
000141F1 20647269766520      <1>
000141F8 30                  <1> Vol_Drv_Name:   db 30h
000141F9 3A                  <1>             db ":"
000141FA 20697320            <1>             db " is "
000141FE 0D0A00              <1>             db 0Dh, 0Ah, 0
                             <1> Dir_Drive_Str:
00014201 54522D444F53204472- <1>                 db "TR-DOS Drive "
0001420A 69766520            <1>
                             <1> Dir_Drive_Name:
0001420E 303A                <1>                 db "0:"
00014210 0D0A                <1>                 db  0Dh, 0Ah
                             <1> Vol_Str_Header:
00014212 566F6C756D65204E61- <1>                 db "Volume Name: "
0001421B 6D653A20            <1>
                             <1> Vol_Name:
0001421F 00<rept>            <1>             times 64 db 0
0001425F 00                  <1>             db 0
                             <1> Vol_Serial_Header:
00014260 0D0A                <1>             db 0Dh, 0Ah
00014262 566F6C756D65205365- <1>             db "Volume Serial No: "
0001426B 7269616C204E6F3A20  <1>
                             <1> Vol_Serial1:
00014274 30303030            <1>             db "0000"
00014278 2D                  <1>             db "-"
                             <1> Vol_Serial2:
00014279 30303030            <1>             db "0000"
0001427D 0D0A00              <1>             db 0Dh, 0Ah, 0
                             <1>
                             <1> ;Vol_Tot_Sec_Str_Start:
                             <1> ;           dd 0
                             <1> Vol_Total_Sector_Header:
00014280 0D0A                <1>             db 0Dh, 0Ah
00014282 566F6C756D65205369- <1>             db "Volume Size : ", 0
0001428B 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:
00014291 467265652053706163- <1>             db "Free Space  : ", 0
0001429A 6520203A2000        <1>
                             <1> ;Vol_Free_Sectors_Str:
                             <1> ;           db "0000000000"
                             <1> ;Vol_Free_Sectors_Str_End:
                             <1> ;           db 0
                             <1>
                             <1> Dir_Str_Header:
000142A0 4469726563746F7279- <1>                 db "Directory: "
000142A9 3A20                <1>
000142AB 2F                  <1> Dir_Str_Root:   db "/"
000142AC 00<rept>            <1> Dir_Str:        times 64 db 0
000142EC 00000000            <1>                 dd 0
000142F0 00                  <1>                 db 0
                             <1>
                             <1> Msg_Bad_Command:
000142F1 42616420636F6D6D61- <1>                 db "Bad command or file name!"
000142FA 6E64206F722066696C- <1>
00014303 65206E616D6521      <1>
0001430A 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1>
                             <1> msgl_drv_not_ready:
0001430D 070D0A              <1>             db 07h, 0Dh, 0Ah
                             <1>
                             <1> ; CMD_INTR.ASM - 09/11/2011 - Messages
                             <1>
                             <1> Msg_Not_Ready_Read_Err:
00014310 4472697665206E6F74- <1>                 db "Drive not ready or read error!"
00014319 207265616479206F72- <1>
00014322 207265616420657272- <1>
0001432B 6F7221              <1>
0001432E 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1>
                             <1> Msg_Not_Ready_Write_Err:
00014331 4472697665206E6F74- <1>                 db "Drive not ready or write error!"
0001433A 207265616479206F72- <1>
00014343 207772697465206572- <1>
0001434C 726F7221            <1>
00014350 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1>
                             <1> Msg_Dir_Not_Found:
00014353 4469726563746F7279- <1>                 db "Directory not found!"
0001435C 206E6F7420666F756E- <1>
00014365 6421                <1>
00014367 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1>
                             <1> Msg_File_Not_Found:
0001436A 46696C65206E6F7420- <1>                 db "File not found!"
00014373 666F756E6421        <1>
```

```
                                <1>                 db 0Dh, 0Ah, 0
                                <1>
                                <1> Msg_File_Directory_Not_Found:
0001437C 46696C65206F722064-    <1>                 db "File or directory not found!"
00014385 69726563746F727920-    <1>
0001438E 6E6F7420666F756E64-    <1>
00014397 21                     <1>
00014398 0D0A00                 <1>                 db 0Dh, 0Ah, 0
                                <1>
                                <1> Msg_LongName_Not_Found:
0001439B 4C6F6E67206E616D65-    <1>                 db "Long name not found!"
000143A4 206E6F7420666F756E-    <1>
000143AD 6421                   <1>
000143AF 0D0A00                 <1>                 db 0Dh, 0Ah, 0
                                <1>
                                <1> beep_Insufficient_Memory: ; 20/02/2017
000143B2 0D0A                   <1>                 db 0Dh, 0Ah
000143B4 07                     <1>                 db 07h
                                <1> Msg_Insufficient_Memory:
000143B5 496E73756666696369-    <1>                 db "Insufficient memory!"
000143BE 656E74206D656D6F72-    <1>
000143C7 7921                   <1>
000143C9 0D0A00                 <1>                 db 0Dh, 0Ah, 0
                                <1>
                                <1> Msg_Error_Code:
000143CC 436F6D6D616E642066-    <1>                 db 'Command failed! Error code : '
000143D5 61696C656421204572-    <1>
000143DE 726F7220636F646520-    <1>
000143E7 3A20                   <1>
000143E9 303068                 <1> error_code_hex: db '00h'
000143EC 0A0A00                 <1>                 db 0Ah, 0Ah, 0
                                <1>
000143EF 90                     <1> align 2
                                <1>
                                <1> ; 10/02/2016
                                <1> ; DIR.ASM - 09/10/2011
                                <1>
000143F0 3C4449523E20202020-    <1> Type_Dir:       db '<DIR>     ' ; 10 bytes
000143F9 20                     <1>
                                <1>
                                <1> File_Name:
000143FA 20<rept>               <1>                 times 12 db 20h
00014406 20                     <1>                 db 20h
                                <1> Dir_Or_FileSize:
00014407 20<rept>               <1>                 times 10 db 20h
00014411 20                     <1>                 db 20h
                                <1> File_Attribute:
00014412 20202020               <1>                 dd 20202020h
00014416 20                     <1>                 db 20h
                                <1> File_Day:
00014417 3030                   <1>                 db '0','0'
00014419 2F                     <1>                 db '/'
                                <1> File_Month:
0001441A 3030                   <1>                 db '0','0'
0001441C 2F                     <1>                 db '/'
                                <1> File_Year:
0001441D 30303030               <1>                 db '0','0','0','0'
00014421 20                     <1>                 db 20h
                                <1> File_Hour:
00014422 3030                   <1>                 db '0','0'
00014424 3A                     <1>                 db ':'
                                <1> File_Minute:
00014425 3030                   <1>                 db '0','0'
00014427 00                     <1>                 db 0
                                <1>
                                <1> Decimal_File_Count_Header:
00014428 0D0A                   <1>                 db 0Dh, 0Ah
                                <1> Decimal_File_Count:
0001442A 00<rept>               <1>                 times 6 db 0
                                <1>
00014430 2066696C6528732920-    <1> str_files:   db " file(s) & "
00014439 2620                   <1>
                                <1> Decimal_Dir_Count:
0001443B 00<rept>               <1>                 times 6 db 0
                                <1> str_dirs:
00014441 206469726563746F72-    <1>                 db " directory(s) "
0001444A 7928732920             <1>
0001444F 0D0A00                 <1>                 db 0Dh, 0Ah, 0
                                <1>
00014452 206279746528732920-    <1> str_bytes:   db " byte(s) in file(s)"
0001445B 696E2066696C652873-    <1>
00014464 29                     <1>
00014465 0D0A00                 <1>                 db 0Dh, 0Ah, 0
                                <1>
                                <1> ; CMD_INTR.ASM - 09/11/2011
                                <1> ; 07/10/2010
                                <1> Msg_invalid_name_chars:
00014468 496E76616C69642066-    <1>                     db "Invalid file or directory name characters!"
00014471 696C65206F72206469-    <1>
0001447A 726563746F7279206E-    <1>
00014483 616D65206368617261-    <1>
0001448C 637465727321           <1>
00014492 0D0A00                 <1>                 db 0Dh, 0Ah, 0
                                <1> ; 21/02/2016
00014495 46696C65206F722064-    <1> Msg_Name_Exists: db "File or directory name exists!"
0001449E 69726563746F727920-    <1>
000144A7 6E616D652065786973-    <1>
000144B0 747321                 <1>
000144B3 0D0A00                 <1>                 db 0Dh, 0Ah, 0
                                <1> Msg_DoYouWantMkdir:
000144B6 446F20796F75207761-    <1>                 db "Do you want to make directory ", 0
000144BF 6E7420746F206D616B-    <1>
000144C8 65206469726563746F-    <1>
000144D1 72792000               <1>
000144D5 2028592F4E29203F20-    <1> Msg_YesNo:      db " (Y/N) ? ", 0
```

```
288 000144DE 00                  <1>
289 000144DF 000D0A00            <1> Y_N_nextline:      db 0, 0Dh, 0Ah, 0
290 000144E3 4F4B2E0D0A00        <1> Msg_OK:            db "OK.", 0Dh, 0Ah, 0
291                              <1>
292                              <1> ; 27/02/2016
293                              <1> Msg_DoYouWantRmDir:
294 000144E9 446F20796F75207761- <1>                db "Do you want to delete directory ", 0
294 000144F2 6E7420746F2064656C- <1>
294 000144FB 657465206469726563- <1>
294 00014504 746F72792000        <1>
295                              <1> Msg_Dir_Not_Empty:
296 0001450A 4469726563746F7279- <1>                db "Directory not empty!"
296 00014513 206E6F7420656D7074- <1>
296 0001451C 7921                <1>
297 0001451E 0D0A00              <1>                db 0Dh, 0Ah, 0
298                              <1>
299                              <1> Msg_DoYouWantDelete:
300 00014521 446F20796F75207761- <1>                db "Do you want to delete file ",0
300 0001452A 6E7420746F2064656C- <1>
300 00014533 6574652066696C6520- <1>
300 0001453C 00                  <1>
301                              <1>
302 0001453D 44656C657465642E2E- <1> Msg_Deleted:    db "Deleted...", 0Dh, 0Ah, 0
302 00014546 2E0D0A00            <1>
303                              <1>
304                              <1> Msg_Permission_Denied:
305 0001454A 07                  <1>                db 7
306 0001454B 5065726D697373696F- <1>                db "Permission denied!", 0Dh, 0Ah, 0
306 00014554 6E2064656E69656421- <1>
306 0001455D 0D0A00              <1>
307                              <1>
308                              <1> ; 04/03/2016
309 00014560 4E657720            <1> Msg_New:        db "New "
310 00014564 00                  <1>                db 0
311                              <1> Str_Attributes:
312 00014565 417474726962757465- <1>                db "Attributes : "
312 0001456E 73203A20            <1>
313 00014572 4E4F524D414C        <1> Attr_Chars:     db "NORMAL"
314 00014578 00                  <1>                db 0
315                              <1>
316                              <1> ; 06/03/2016
317                              <1> ; CMD_INTR.ASM - 16/11/2010
318                              <1> Msg_DoYouWantRename:
319 00014579 446F20796F75207761- <1>                db "Do you want to rename ", 0
319 00014582 6E7420746F2072656E- <1>
319 0001458B 616D652000          <1>
320 00014590 66696C652000        <1> Rename_File:    db "file ", 0
321 00014596 6469726563746F7279- <1> Rename_Directory: db "directory ", 0
321 0001459F 2000                <1>
322 000145A1 00<rept>            <1> Rename_OldName: times 13 db 0
323 000145AE 20617320            <1> Msg_File_rename_as: db " as "
324 000145B2 00<rept>            <1> Rename_NewName: times 13 db 0
325                              <1>
326                              <1> ; 08/03/2016
327                              <1> ; CMD_INTR.ASM - 01/08/2010 - 23/04/2011
328                              <1> msg_not_same_drv:
329 000145BF 4E6F742073616D6520- <1>                db "Not same drive!"
329 000145C8 647269766521        <1>
330 000145CE 0D0A00              <1>                db 0Dh, 0Ah, 0
331                              <1>
332                              <1> Msg_DoYouWantMoveFile:
333 000145D1 446F20796F75207761- <1>                db "Do you want to move file", 0
333 000145DA 6E7420746F206D6F76- <1>
333 000145E3 652066696C6500      <1>
334                              <1>
335                              <1> msg_insufficient_disk_space:
336 000145EA 496E73756666696369- <1>                db "Insufficient disk space!"
336 000145F3 656E74206469736B20- <1>
336 000145FC 737061636521        <1>
337 00014602 0D0A00              <1>                db 0Dh, 0Ah, 0
338                              <1>
339                              <1> ; 01/08/2010
340                              <1> msg_source_file:
341 00014605 0D0A536F7572636520- <1>                db 0Dh, 0Ah, "Source file name      :   "
341 0001460E 66696C65206E616D65- <1>
341 00014617 2020202020203A2020- <1>
341 00014620 20                  <1>
342                              <1> msg_source_file_drv:
343 00014621 203A00              <1>                db " :", 0
344                              <1> msg_destination_file:
345 00014624 0D0A44657374696E61- <1>                db 0Dh, 0Ah, "Destination file name :   "
345 0001462D 74696F6E2066696C65- <1>
345 00014636 206E616D65203A2020- <1>
345 0001463F 20                  <1>
346                              <1> msg_destination_file_drv:
347 00014640 203A00              <1>                db " :", 0
348                              <1> msg_copy_nextline:
349 00014643 0D0A00              <1>                db 0Dh, 0Ah, 0
350                              <1>
351                              <1> ; 15/03/2016
352                              <1> ; CMD_INTR.ASM
353                              <1>
354                              <1> Msg_DoYouWantOverWriteFile:
355 00014646 446F20796F75207761- <1>                    db "Do you want to overwrite file ",0
355 0001464F 6E7420746F206F7665- <1>
355 00014658 727772697465206669- <1>
355 00014661 6C652000            <1>
356                              <1>
357                              <1> Msg_DoYouWantCopyFile:
358 00014665 446F20796F75207761- <1>                    db "Do you want to copy file",0
358 0001466E 6E7420746F20636F70- <1>
358 00014677 792066696C6500      <1>
359                              <1>
360                              <1> Msg_read_file_error_before_EOF:
361 0001467E 46696C652072656164- <1>                db "File reading error! (before EOF)"
```

```
00014687 696E67206572726F72- <1>
00014690 2120286265666F7265- <1>
00014699 20454F4629          <1>
0001469E 0A0A00              <1>             db 0Ah, 0Ah, 0
                             <1>
                             <1> ; 18/03/2016
                             <1> ; TRDOS 386 (v2.0) mainprog copy procedure
                             <1> msg_reading:
000146A1 52656164696E672E2E- <1>             db "Reading... ", 0
000146AA 2E2000              <1>
                             <1> msg_writing:
000146AD 57726974696E672E2E- <1>             db "Writing... ", 0
000146B6 2E2000              <1>
                             <1> percentagestr:
000146B9 2020202500          <1>             db "   %", 0  ; "  0%" .. "100%"
                             <1> ; 11/04/2016
                             <1> Msg_No_Set_Space:
000146BE 496E73756666696369- <1>                 db "Insufficient environment space!"
000146C7 656E7420656E766972- <1>
000146D0 6F6E6D656E74207370- <1>
000146D9 61636521            <1>
000146DD 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1> ; 18/04/2016
                             <1> isc_msg:
000146E0 0D0A                <1>             db 0Dh, 0Ah
000146E2 494E56414C49442053- <1>             db "INVALID SYSTEM CALL", 0
000146EB 595354454D2043414C- <1>
000146F4 4C00                <1>
                             <1> usi_msg:
000146F6 0D0A                <1>             db 0Dh, 0Ah
000146F8 554E444546494E4544- <1>             db "UNDEFINED SOFTWARE INTERRUPT", 0
00014701 20534F465457415245- <1>
0001470A 20494E544552525550- <1>
00014713 5400                <1>
                             <1> ifc_msg:
00014715 0D0A                <1>             db 0Dh, 0Ah
00014717 494E56414C49442046- <1>             db "INVALID FUNCTION CALL"
00014720 554E4354494F4E2043- <1>
00014729 414C4C              <1>
                             <1> inv_msg_for_trdos_v2:
0001472C 20                  <1>             db 20h
0001472D 666F72205452444F53- <1>             db "for TRDOS v2!"
00014736 20763221            <1>
0001473A 07                  <1>             db 07h
0001473B 0D0A                <1>             db 0Dh, 0Ah
0001473D 0D0A                <1>             db 0Dh, 0Ah
0001473F 494E5420            <1>             db "INT "
00014743 303068              <1> int_num_str: db "00h"
00014746 0D0A                <1>             db 0Dh, 0Ah
00014748 454158203A20        <1>             db "EAX : "
0001474E 303030303030303068- <1> eax_str:    db "00000000h", 0Dh, 0Ah
00014757 0D0A                <1>
00014759 454950203A20        <1>             db "EIP : "
0001475F 303030303030303068- <1> eip_str:    db "00000000h", 0Dh, 0Ah, 0
00014768 0D0A00              <1>
                             <1>
                             <1> ; 07/10/2016
                             <1> ; Device names & parameters (for kernel devices)
                             <1>
0001476B 90                  <1> align 2
                             <1> KDEV_NAME:
0001476C 5454590000000000    <1>             db 'TTY',0,0,0,0,0 ; 1
00014774 4D454D0000000000    <1>             db 'MEM',0,0,0,0,0 ; 2
0001477C 4644300000000000    <1>             db 'FD0',0,0,0,0,0 ; 3
00014784 4644310000000000    <1>             db 'FD1',0,0,0,0,0 ; 4
0001478C 4844300000000000    <1>             db 'HD0',0,0,0,0,0 ; 5
00014794 4844310000000000    <1>             db 'HD1',0,0,0,0,0 ; 6
0001479C 4844320000000000    <1>             db 'HD2',0,0,0,0,0 ; 7
000147A4 4844330000000000    <1>             db 'HD3',0,0,0,0,0 ; 8
000147AC 4C50540000000000    <1>             db 'LPT',0,0,0,0,0 ; 9
000147B4 5454593000000000    <1>             db 'TTY0',0,0,0,0 ; 10
000147BC 5454593100000000    <1>             db 'TTY1',0,0,0,0 ; 11
000147C4 5454593200000000    <1>             db 'TTY2',0,0,0,0 ; 12
000147CC 5454593300000000    <1>             db 'TTY3',0,0,0,0 ; 13
000147D4 5454593400000000    <1>             db 'TTY4',0,0,0,0 ; 14
000147DC 5454593500000000    <1>             db 'TTY5',0,0,0,0 ; 15
000147E4 5454593600000000    <1>             db 'TTY6',0,0,0,0 ; 16
000147EC 5454593700000000    <1>             db 'TTY7',0,0,0,0 ; 17
000147F4 5454593800000000    <1>             db 'TTY8',0,0,0,0 ; 18
000147FC 5454593900000000    <1>             db 'TTY9',0,0,0,0 ; 19
00014804 434F4D3100000000    <1>             db 'COM1',0,0,0,0 ; 18
0001480C 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
```

```
449                                     <1>             ;db 'SD1',0,0,0,0,0 ; 36
450                                     <1>             ;db 'SD2',0,0,0,0,0 ; 37
451                                     <1>             ;db 'SD3',0,0,0,0,0 ; 38
452                                     <1>             ;db 'SATA0'    ; 35
453                                     <1>             ;db 'SATA1'    ; 36
454                                     <1>             ;db 'SATA2'        ; 37
455                                     <1>             ;db 'SATA3'        ; 38
456                                     <1>             ;db 'PATA0',0,0,0  ; 5
457                                     <1>             ;db 'PATA1',0,0,0  ; 6
458                                     <1>             ;db 'PATA2',0,0,0  ; 7
459                                     <1>             ;db 'PATA3',0,0,0  ; 8
460                                     <1>             ;db 'WIRELESS'       ; 39
461                                     <1>             ;db 'HDMI',0,0,0,0 ; 40
462 00014814 4E554C4C00000000           <1>             db 'NULL',0,0,0,0 ; 0
463                                     <1>
464                                     <1> NumOfKernelDevNames equ ($-KDEV_NAME) / 8 ; 20 (07/10/2016)
465                                     <1>
466                                     <1> KDEV_NUMBER:
467 0001481C 010203040506070809         <1>             db 1,2,3,4,5,6,7,8,9
468 00014825 0A0B0C0D0E0F101112-        <1>             db 10,11,12,13,14,15,16,17,18,19
468 0001482E 13                         <1>
469 0001482F 121300                     <1>             db 18,19,0
470                                     <1>
471                                     <1> NumOfKernelDevices equ $ - KDEV_NUMBER
472                                     <1>
473                                     <1> KDEV_OADDR:
474 00014832 [4F400100]                 <1>             dd otty ;tty  ; 1
475 00014836 [4F400100]                 <1>             dd sret ;mem  ; 2
476 0001483A [4F400100]                 <1>             dd sret ;fd0  ; 3
477 0001483E [4F400100]                 <1>             dd sret ;fd1  ; 4
478 00014842 [4F400100]                 <1>             dd sret ;hd0  ; 5
479 00014846 [4F400100]                 <1>             dd sret ;hd1  ; 6
480 0001484A [4F400100]                 <1>             dd sret ;hd2  ; 7
481 0001484E [4F400100]                 <1>             dd sret ;hd3  ; 8
482 00014852 [4F400100]                 <1>             dd sret ;lpt  ; 9
483 00014856 [4F400100]                 <1>             dd ocvt ;tty0 ; 10
484 0001485A [4F400100]                 <1>             dd ocvt ;tty1 ; 11
485 0001485E [4F400100]                 <1>             dd ocvt ;tty2 ; 12
486 00014862 [4F400100]                 <1>             dd ocvt ;tty3 ; 13
487 00014866 [4F400100]                 <1>             dd ocvt ;tty4 ; 14
488 0001486A [4F400100]                 <1>             dd ocvt ;tty5 ; 15
489 0001486E [4F400100]                 <1>             dd ocvt ;tty6 ; 16
490 00014872 [4F400100]                 <1>             dd ocvt ;tty7 ; 17
491 00014876 [4F400100]                 <1>             dd ocvt ;tty8 ; 18
492 0001487A [4F400100]                 <1>             dd ocvt ;tty9 ; 19
493                                     <1>             ;dd ocvt ;com1 ; 18
494                                     <1>             ;dd ocvt ;com2 ; 19
495 0001487E [4F400100]                 <1>             dd sret ;null ; 20
496                                     <1> KDEV_CADDR:
497 00014882 [4F400100]                 <1>             dd ctty ;tty  ; 1
498 00014886 [4F400100]                 <1>             dd cret ;mem  ; 2
499 0001488A [4F400100]                 <1>             dd cret ;fd0  ; 3
500 0001488E [4F400100]                 <1>             dd cret ;fd1  ; 4
501 00014892 [4F400100]                 <1>             dd cret ;hd0  ; 5
502 00014896 [4F400100]                 <1>             dd cret ;hd1  ; 6
503 0001489A [4F400100]                 <1>             dd cret ;hd2  ; 7
504 0001489E [4F400100]                 <1>             dd cret ;hd3  ; 8
505 000148A2 [4F400100]                 <1>             dd cret ;lpt  ; 9
506 000148A6 [4F400100]                 <1>             dd ocvt ;tty0 ; 10
507 000148AA [4F400100]                 <1>             dd ccvt ;tty1 ; 11
508 000148AE [4F400100]                 <1>             dd ccvt ;tty2 ; 12
509 000148B2 [4F400100]                 <1>             dd ccvt ;tty3 ; 13
510 000148B6 [4F400100]                 <1>             dd ccvt ;tty4 ; 14
511 000148BA [4F400100]                 <1>             dd ccvt ;tty5 ; 15
512 000148BE [4F400100]                 <1>             dd ccvt ;tty6 ; 16
513 000148C2 [4F400100]                 <1>             dd ccvt ;tty7 ; 17
514 000148C6 [4F400100]                 <1>             dd ccvt ;tty8 ; 18
515 000148CA [4F400100]                 <1>             dd ccvt ;tty9 ; 19
516                                     <1>             ;dd ccvt ;com1 ; 18
517                                     <1>             ;dd ccvt ;com2 ; 19
518 000148CE [4F400100]                 <1>             dd cret ;null ; 20
519                                     <1>
520                                     <1> KDEV_RADDR:
521 000148D2 [4F400100]                 <1>             dd rtty ;tty  ; 1
522 000148D6 [4F400100]                 <1>             dd rmem ;mem  ; 2
523 000148DA [4F400100]                 <1>             dd rfd  ;fd0  ; 3
524 000148DE [4F400100]                 <1>             dd rfd  ;fd1  ; 4
525 000148E2 [4F400100]                 <1>             dd rhd  ;hd0  ; 5
526 000148E6 [4F400100]                 <1>             dd rhd  ;hd1  ; 6
527 000148EA [4F400100]                 <1>             dd rhd  ;hd2  ; 7
528 000148EE [4F400100]                 <1>             dd rhd  ;hd3  ; 8
529 000148F2 [4F400100]                 <1>             dd rlpt ;lpt  ; 9
530 000148F6 [4F400100]                 <1>             dd rcvt ;tty0 ; 10
531 000148FA [4F400100]                 <1>             dd rcvt ;tty1 ; 11
532 000148FE [4F400100]                 <1>             dd rcvt ;tty2 ; 12
533 00014902 [4F400100]                 <1>             dd rcvt ;tty3 ; 13
534 00014906 [4F400100]                 <1>             dd rcvt ;tty4 ; 14
535 0001490A [4F400100]                 <1>             dd rcvt ;tty5 ; 15
536 0001490E [4F400100]                 <1>             dd rcvt ;tty6 ; 16
537 00014912 [4F400100]                 <1>             dd rcvt ;tty7 ; 17
538 00014916 [4F400100]                 <1>             dd rcvt ;tty8 ; 18
539 0001491A [4F400100]                 <1>             dd rcvt ;tty9 ; 19
540                                     <1>             ;dd rcvt ;com1 ; 18
541                                     <1>             ;dd rcvt ;com2 ; 19
542 0001491E [33340100]                 <1>             dd rnull ;null ; 20
543                                     <1> KDEV_WADDR:
544 00014922 [4F400100]                 <1>             dd wtty ;tty  ; 1
545 00014926 [4F400100]                 <1>             dd wmem ;mem  ; 2
546 0001492A [4F400100]                 <1>             dd wfd  ;fd0  ; 3
547 0001492E [4F400100]                 <1>             dd wfd  ;fd1  ; 4
548 00014932 [4F400100]                 <1>             dd whd  ;hd0  ; 5
549 00014936 [4F400100]                 <1>             dd whd  ;hd1  ; 6
550 0001493A [4F400100]                 <1>             dd whd  ;hd2  ; 7
551 0001493E [4F400100]                 <1>             dd whd  ;hd3  ; 8
552 00014942 [4F400100]                 <1>             dd wlpt ;lpt  ; 9
```

```
                                        <1>             dd xmtt ;tty0 ; 10
                                        <1>             dd xmtt ;tty1 ; 11
                                        <1>             dd xmtt ;tty2 ; 12
                                        <1>             dd xmtt ;tty3 ; 13
                                        <1>             dd xmtt ;tty4 ; 14
                                        <1>             dd xmtt ;tty5 ; 15
                                        <1>             dd xmtt ;tty6 ; 16
                                        <1>             dd xmtt ;tty7 ; 17
                                        <1>             dd xmtt ;tty8 ; 18
                                        <1>             dd xmtt ;tty9 ; 19
                                        <1>             ;dd xmtt ;com1 ; 18
                                        <1>             ;dd xmtt ;com2 ; 19
                                        <1>             dd wnull ;null ; 20
                                        <1>
                                        <1> ; DEV_ACCESS bits:
                                        <1>       ; 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
                                        <1>
                                        <1> KDEV_ACCESS: ; 08/10/2016
00014972 07                             <1>             db  00000111b; tty, 1
00014973 07                             <1>             db  00000111b; mem, 2
00014974 8F                             <1>             db  10001111b; fd0, 3
00014975 8F                             <1>             db  10001111b; fd1, 4
00014976 8F                             <1>             db  10001111b; hd0, 5
00014977 8F                             <1>             db  10001111b; hd1, 6
00014978 8F                             <1>             db  10001111b; hd2, 7
00014979 8F                             <1>             db  10001111b; hd3, 8
0001497A 07                             <1>             db  00000111b   ; lpt, 9
0001497B 07                             <1>             db  00000111b; tty0, 10
0001497C 07                             <1>             db  00000111b; tty1, 11
0001497D 07                             <1>             db  00000111b; tty2, 12
0001497E 07                             <1>             db  00000111b; tty3, 13
0001497F 07                             <1>             db  00000111b; tty4, 14
00014980 07                             <1>             db  00000111b; tty5, 15
00014981 07                             <1>             db  00000111b; tty6, 16
00014982 07                             <1>             db  00000111b; tty7, 17
00014983 07                             <1>             db  00000111b; tty8, 18
00014984 07                             <1>             db  00000111b; tty9, 19
                                        <1>             ;db 00000111b; com1, 18
                                        <1>             ;db 00000111b; com2, 19
00014985 00                             <1>             db  00000000b   ; null, 0
                                        <1>
                                        <1> ; 07/10/2016
                                        <1> NumOfInstallableDevices equ 8
                                        <1> NUMIDEV             equ NumOfInstallableDevices ; 8
                                        <1> NUMOFDEVICES equ NumOfKernelDevices + NumOfInstallableDevices
                                        <1>
                                        <1> ; 26/02/2017
                                        <1> ; IRQ Callback (& Signal Response Byte) service availibity
                                        <1> ; 'syscalbac'
                                        <1> ; ************************************************
                                        <1> ; IRQ 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
                                        <1> ; --- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
                                        <1> ; --- 00 00 00 01 02 03 00 04 00 05 06 07 08 09 00 00
                                        <1> ; ************************************************
                                        <1> IRQenum:
00014986 000000010203000400-            <1>       db  0,0,0,1,2,3,0,4,0,5,6,7,8,9,0,0
0001498F 05060708090000                 <1>
                                        <1>
                                        <1> ; 28/08/2017
                                        <1> ; 20/08/2017
                                        <1> ; DMA Registers (for 'sysdma')
                                        <1> ; 02/07/2017 (sb16mod.s)
00014996 00020406C0C4C8CC               <1> dma_adr:    db 0,2,4,6,0C0h,0C4h,0C8h,0CCh
0001499E 01030507C2C6CACE               <1> dma_cnt:    db 1,3,5,7,0C2h,0C6h,0CAh,0CEh
000149A6 878381828F8B898A               <1> dma_page:   db 87h,83h,81h,82h,8Fh,8Bh,89h,8Ah ; 03/08/2017
000149AE 0A0A0A0AD4D4D4D4               <1> dma_mask:   db 0Ah,0Ah,0Ah,0Ah,0D4h,0D4h,0D4h,0D4h
000149B6 0B0B0B0BD6D6D6D6               <1> dma_mod:    db 0Bh,0Bh,0Bh,0Bh,0D6h,0D6h,0D6h,0D6h
000149BE 0C0C0C0CD8D8D8D8               <1> dma_flip:   db 0Ch,0Ch,0Ch,0Ch,0D8h,0D8h,0D8h,0D8h
                                    ; 27/08/2014
                                    scr_row:
000149C6 E0810B00                       dd 0B8000h + 0A0h + 0A0h + 0A0h ; Row 3
                                    scr_col:
000149CA 00000000                       dd 0

000149CE 90<rept>                   Align 4
                                        ; 15/04/2016
                                        ; TRDOS 386 (TRDOS v2.0)

                                        ; 21/08/2014
                                    ilist:
                                        ;times       32 dd cpu_except ; INT 0 to INT 1Fh
                                        ;
                                        ; Exception list
                                        ; 25/08/2014
000149D0 [E20B0000]                     dd     exc0   ; 0h,  Divide-by-zero Error
000149D4 [E90B0000]                     dd     exc1
000149D8 [F00B0000]                     dd     exc2
000149DC [F70B0000]                     dd     exc3
000149E0 [FB0B0000]                     dd     exc4
000149E4 [FF0B0000]                     dd     exc5
000149E8 [030C0000]                     dd     exc6   ; 06h,  Invalid Opcode
000149EC [070C0000]                     dd     exc7
000149F0 [0B0C0000]                     dd     exc8
000149F4 [0F0C0000]                     dd     exc9
000149F8 [130C0000]                     dd     exc10
000149FC [170C0000]                     dd     exc11
00014A00 [1B0C0000]                     dd     exc12
```

```
3118 00014A04 [1F0C0000]                    dd     exc13  ; 0Dh, General Protection Fault
3119 00014A08 [230C0000]                    dd     exc14  ; 0Eh, Page Fault
3120 00014A0C [270C0000]                    dd     exc15
3121 00014A10 [2B0C0000]                    dd     exc16
3122 00014A14 [2F0C0000]                    dd     exc17
3123 00014A18 [330C0000]                    dd     exc18
3124 00014A1C [370C0000]                    dd     exc19
3125 00014A20 [3B0C0000]                    dd     exc20
3126 00014A24 [3F0C0000]                    dd     exc21
3127 00014A28 [430C0000]                    dd     exc22
3128 00014A2C [470C0000]                    dd     exc23
3129 00014A30 [4B0C0000]                    dd     exc24
3130 00014A34 [4F0C0000]                    dd     exc25
3131 00014A38 [530C0000]                    dd     exc26
3132 00014A3C [570C0000]                    dd     exc27
3133 00014A40 [5B0C0000]                    dd     exc28
3134 00014A44 [5F0C0000]                    dd     exc29
3135 00014A48 [630C0000]                    dd     exc30
3136 00014A4C [670C0000]                    dd     exc31
3137                                    IRQ_list: ; 28/02/2017 ('syscalbac')
3138                                          ; Interrupt list
3139 00014A50 [56090000]                      dd     timer_int    ; INT 20h
3140                                                 ;dd    irq0
3141 00014A54 [CE100000]                      dd     kb_int       ; 24/01/2016
3142                                                 ;dd    irq1
3143 00014A58 [380B0000]                      dd     irq2
3144                                                 ; COM2 int
3145 00014A5C [3C0B0000]                      dd     irq3
3146                                                 ; COM1 int
3147 00014A60 [470B0000]                      dd     irq4
3148 00014A64 [520B0000]                      dd     irq5
3149                                    ;DISKETTE_INT: ;06/02/2015
3150 00014A68 [D3510000]                      dd     fdc_int            ; 16/02/2015, IRQ 6 handler
3151                                                 ;dd    irq6
3152                                    ; Default IRQ 7 handler against spurious IRQs (from master PIC)
3153                                    ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
3154 00014A6C [C10E0000]                      dd     default_irq7 ; 25/02/2015
3155                                                 ;dd    irq7
3156                                    ; Real Time Clock Interrupt
3157 00014A70 [C10A0000]                      dd     rtc_int            ; 23/02/2015, IRQ 8 handler
3158                                                 ;dd    irq8   ; INT 28h
3159 00014A74 [620B0000]                      dd     irq9
3160 00014A78 [660B0000]                      dd     irq10
3161 00014A7C [6A0B0000]                      dd     irq11
3162 00014A80 [6E0B0000]                      dd     irq12
3163 00014A84 [720B0000]                      dd     irq13
3164                                    ;HDISK_INT1:  ;06/02/2015
3165 00014A88 [865B0000]                      dd     hdc1_int     ; 21/02/2015, IRQ 14 handler
3166                                                 ;dd    irq14
3167                                    ;HDISK_INT2:  ;06/02/2015
3168 00014A8C [AD5B0000]                      dd     hdc2_int     ; 21/02/2015, IRQ 15 handler
3169                                                 ;dd    irq15  ; INT 2Fh
3170                                                 ; 14/08/2015
3171                                                 ;dd    sysent       ; INT 30h (system calls)
3172
3173                                                 ; 15/04/2016
3174                                                 ; TRDOS 386(TRDOS v2.0) Software Interrupts
3175
3176 00014A90 [ED4A0100]                           dd     int30h       ; Reserved for
3177                                                                     ; !!! Retro UNIX (RUNIX) !!!
3178                                                                     ; !!! SINGLIX !!! System Calls
3179 00014A94 [C1170000]                           dd     int31h       ; Video BIOS (IBM PC/AT, Int 10h)
3180 00014A98 [ED0E0000]                           dd     int32h       ; Keyboard Functions (IBM PC/AT, Int 16h)
3181 00014A9C [8A520000]                           dd     int33h       ; DISK I/O (IBM PC/AT, Int 13h)
3182 00014AA0 [E62C0100]                           dd     int34h       ; #IOCTL# (I/O port access support for ring 3)
3183 00014AA4 [3E6A0000]                           dd     int35h       ; Time/Date Functions (IBM PC/AT, Int 1Ah)
3184 00014AA8 [750D0000]                           dd     ignore_int   ; INT 36h : Timer Functions
3185 00014AAC [750D0000]                           dd     ignore_int   ; INT 37h
3186 00014AB0 [750D0000]                           dd     ignore_int   ; INT 38h
3187 00014AB4 [750D0000]                           dd     ignore_int   ; INT 39h
3188 00014AB8 [750D0000]                           dd     ignore_int   ; INT 3Ah
3189 00014ABC [750D0000]                           dd     ignore_int   ; INT 3Bh
3190 00014AC0 [750D0000]                           dd     ignore_int   ; INT 3Ch
3191 00014AC4 [750D0000]                           dd     ignore_int   ; INT 3Dh
3192 00014AC8 [750D0000]                           dd     ignore_int   ; INT 3Eh
3193 00014ACC [750D0000]                           dd     ignore_int   ; INT 3Fh
3194 00014AD0 [AAD70000]                           dd     sysent       ; INT 40h : !!! TRDOS 386 System Calls !!!
3195                                                      ;dd    ignore_int
3196 00014AD4 00000000                             dd     0
3197
3198                                         ; 20/08/2014
3199                                           ; /* This is the default interrupt "handler" :-) */
3200                                           ; Linux v0.12 (head.s)
3201                                         int_msg:
3202 00014AD8 556E6B6E6F776E2069-                      db "Unknown interrupt ! ", 0
3202 00014AE1 6E7465727275707420-
3202 00014AEA 212000
3203
3204                                         ; 15/04/2016
3205                                         ; TRDOS 386 (TRDOS v2.0)
3206
3207                                         ; 29/04/2016
3208                                         int30h:
3209                                         trdos_isc_routine:
3210                                               ; 02/05/2016
3211                                               ; 01/05/2016
3212                                               ; 29/04/2016
3213                                               ; 18/04/2016
3214                                               ; 15/04/2016 (TRDOS 386 = TRDOS v2.0)
3215                                               ; 17/04/2011 (TRDOS v1.0, 'IFC.ASM')
3216                                               ; 03/02/2011 ('trdos_ifc_routine')
3217                                               ;
3218 00014AED 8B1C24                               mov    ebx, [esp] ; EIP (next)
3219 00014AF0 83EB02                               sub    ebx, 2 ; EIP (CD ##h)
3220
```

```
3221 00014AF3 89C1                      mov   ecx, eax
3222 00014AF5 8A4301                    mov   al, [ebx+1] ; CDh ##h
3223
3224 00014AF8 66BA1000                  mov   dx, KDATA
3225 00014AFC 8EDA                      mov   ds, dx
3226 00014AFE 8EC2                      mov   es, dx
3227
3228 00014B00 FC                        cld
3229 00014B01 8B15[C0890100]            mov   edx, [k_page_dir]
3230 00014B07 0F22DA                    mov   cr3, edx
3231
3232 00014B0A E8BEF7FEFF                call  bytetohex
3233 00014B0F 66A3[43470100]            mov   [int_num_str], ax
3234
3235 00014B15 89D8                      mov   eax, ebx ; EIP
3236 00014B17 E8F1F7FEFF                call  dwordtohex
3237 00014B1C 8915[5F470100]            mov   [eip_str], edx
3238 00014B22 A3[63470100]              mov   [eip_str+4], eax
3239
3240 00014B27 89C8                      mov   eax, ecx
3241 00014B29 E8DFF7FEFF                call  dwordtohex
3242 00014B2E 8915[4E470100]            mov   [eax_str], edx
3243 00014B34 A3[52470100]              mov   [eax_str+4], eax
3244
3245 00014B39 43                        inc   ebx
3246 00014B3A 8A03                      mov   al, [ebx] ; Interrupt number
3247
3248                                trdos_isc_handler:
3249 00014B3C 80FE30                        cmp   dh, 30h ; Retro UNIX, SINGLIX System calls
3250 00014B3F 7507                          jne   short trdos_usi_handler
3251 00014B41 BE[E0460100]                  mov   esi, isc_msg
3252 00014B46 EB05                          jmp   short loc_write_inv_system_call_msg
3253
3254                                trdos_usi_handler:
3255 00014B48 BE[F6460100]                  mov   esi, usi_msg
3256
3257                                loc_write_inv_system_call_msg:
3258 00014B4D E8032AFFFF                    call  print_msg
3259                                        ; 29/04/2016
3260 00014B52 BE[2C470100]                  mov   esi, inv_msg_for_trdos_v2
3261 00014B57 E8F929FFFF                    call  print_msg
3262
3263                                loc_ifc_terminate_process:
3264                                        ; u.uno = process number
3265                                        ; 29/04/2016
3266
3267                                        ; 02/05/2016
3268 00014B5C FE05[5B030300]                inc   byte [sysflg] ; 0FFh -> 0
3269
3270 00014B62 B801000000                    mov   eax, 1
3271 00014B67 E9178FFFFF                    jmp   sysexit
3272
3273                                ; 07/03/2015
3274                                ; Temporary Code
3275                                display_disks:
3276 00014B6C 803D[FE6D0000]00              cmp   byte [fd0_type], 0
3277 00014B73 7605                          jna   short ddsks1
3278 00014B75 E87D000000                    call  pdskm
3279                                ddsks1:
3280 00014B7A 803D[FF6D0000]00              cmp   byte [fd1_type], 0
3281 00014B81 760C                          jna   short ddsks2
3282 00014B83 C605[C74C0100]31              mov   byte [dskx], '1'
3283 00014B8A E868000000                    call  pdskm
3284                                ddsks2:
3285 00014B8F 803D[006E0000]00              cmp   byte [hd0_type], 0
3286 00014B96 7654                          jna   short ddsk6
3287 00014B98 66C705[C54C0100]68-           mov   word [dsktype], 'hd'
3287 00014BA0 64
3288 00014BA1 C605[C74C0100]30              mov   byte [dskx], '0'
3289 00014BA8 E84A000000                    call  pdskm
3290                                ddsks3:
3291 00014BAD 803D[016E0000]00              cmp   byte [hd1_type], 0
3292 00014BB4 7636                          jna   short ddsk6
3293 00014BB6 C605[C74C0100]31              mov   byte [dskx], '1'
3294 00014BBD E835000000                    call  pdskm
3295                                ddsks4:
3296 00014BC2 803D[026E0000]00              cmp   byte [hd2_type], 0
3297 00014BC9 7621                          jna   short ddsk6
3298 00014BCB C605[C74C0100]32              mov   byte [dskx], '2'
3299 00014BD2 E820000000                    call  pdskm
3300                                ddsks5:
3301 00014BD7 803D[036E0000]00              cmp   byte [hd3_type], 0
3302 00014BDE 760C                          jna   short ddsk6
3303 00014BE0 C605[C74C0100]33              mov   byte [dskx], '3'
3304 00014BE7 E80B000000                    call  pdskm
3305                                ddsk6:
3306 00014BEC BE[EF4C0100]                  mov   esi, nextline
3307 00014BF1 E806000000                    call  pdskml
3308                                pdskm_ok:
3309 00014BF6 C3                            retn
3310                                pdskm:
3311 00014BF7 BE[C34C0100]                  mov   esi, dsk_ready_msg
3312                                pdskml:
3313 00014BFC AC                            lodsb
3314 00014BFD 08C0                          or    al, al
3315 00014BFF 74F5                          jz    short pdskm_ok
3316 00014C01 56                            push  esi
3317                                        ; 13/05/2016
3318 00014C02 BB07000000                      mov     ebx, 7  ; Black background,
3319                                                     ; light gray forecolor
3320                                                     ; Video page 0 (bh=0)
3321 00014C07 E8F1D6FEFF                    call  _write_tty
3322 00014C0C 5E                            pop   esi
3323 00014C0D EBED                          jmp   short pdskml
3324
```

```
3325 00014C0F 90                        Align 2
3326                                    ; 21/08/2014
3327                                    exc_msg:
3328 00014C10 435055206578636570-       db "CPU exception ! "
3328 00014C19 74696F6E202120
3329                                    excnstr:          ; 25/08/2014
3330 00014C20 3F3F68202045495020-       db "??h", "  EIP : "
3330 00014C29 3A20
3331                                    EIPstr: ; 29/08/2014
3332 00014C2B 00<rept>                  times 12 db 0
3333
3334                                    ; 23/02/2015
3335                                    ; 25/08/2014
3336                                    ;scounter:
3337                                    ;     db 5
3338                                    ;     db 19
3339
3340                                    ; 06/11/2014
3341                                    ; Memory Information message
3342                                    ; 14/08/2015
3343                                    msg_memory_info:
3344 00014C37 07                        db    07h
3345 00014C38 0D0A                      db    0Dh, 0Ah
3346                                    ;db    "MEMORY ALLOCATION INFO", 0Dh, 0Ah, 0Dh, 0Ah
3347 00014C3A 546F74616C206D656D-       db    "Total memory : "
3347 00014C43 6F7279203A20
3348                                    mem_total_b_str: ; 10 digits
3349 00014C49 303030303030303030-       db    "0000000000 bytes", 0Dh, 0Ah
3349 00014C52 302062797465730D0A
3350 00014C5B 202020202020202020-       db    "               ", 20h, 20h, 20h
3350 00014C64 202020202020202020
3351                                    mem_total_p_str: ; 7 digits
3352 00014C6D 303030303030302070-       db    "0000000 pages", 0Dh, 0Ah
3352 00014C76 616765730D0A
3353 00014C7C 0D0A                      db    0Dh, 0Ah
3354 00014C7E 46726565206D656D6F-       db    "Free memory  : "
3354 00014C87 727920203A20
3355                                    free_mem_b_str:  ; 10 digits
3356 00014C8D 3F3F3F3F3F3F3F3F3F-       db    "?????????? bytes", 0Dh, 0Ah
3356 00014C96 3F2062797465730D0A
3357 00014C9F 202020202020202020-       db    "               ", 20h, 20h, 20h
3357 00014CA8 202020202020202020
3358                                    free_mem_p_str:  ; 7 digits
3359 00014CB1 3F3F3F3F3F3F3F2070-       db    "??????? pages", 0Dh, 0Ah
3359 00014CBA 616765730D0A
3360 00014CC0 0D0A00                    db    0Dh, 0Ah, 0
3361
3362                                    dsk_ready_msg:
3363 00014CC3 0D0A                      db    0Dh, 0Ah
3364                                    dsktype:
3365 00014CC5 6664                      db    'fd'
3366                                    dskx:
3367 00014CC7 30                        db    '0'
3368 00014CC8 20                        db    20h
3369 00014CC9 697320524541445920-       db    'is READY ...'
3369 00014CD2 2E2E2E
3370 00014CD5 00                        db    0
3371
3372                                    setup_error_msg:
3373 00014CD6 0D0A                      db 0Dh, 0Ah
3374 00014CD8 4469736B2053657475-       db 'Disk Setup Error !'
3374 00014CE1 70204572726F722021
3375 00014CEA 0D0A00                    db 0Dh, 0Ah,0
3376
3377                                    next2line: ; 08/02/2016
3378 00014CED 0D0A                      db    0Dh, 0Ah
3379                                    nextline:
3380 00014CEF 0D0A00                    db    0Dh, 0Ah, 0
3381
3382                                    ; temporary
3383                                    ; 19/12/2020
3384                                    msg_lfb_addr:
3385                                    ;db    0Dh, 0Ah
3386 00014CF2 4C696E656172206672-       db    "Linear frame buffer at "
3386 00014CFB 616D65206275666665-
3386 00014D04 7220617420
3387                                    lfb_addr_str: ; 8 (hex) digits
3388 00014D09 303030303030303068-       db    "00000000h", 0Dh, 0Ah
3388 00014D12 0D0A
3389 00014D14 0D0A00                    db    0Dh, 0Ah, 0
3390
3391                                    ; KERNEL - SYSINIT Messages
3392                                    ; 24/08/2015
3393                                    ; 13/04/2015 - (Retro UNIX 386 v1 Beginning)
3394                                    ; 14/07/2013
3395                                    ;kernel_init_err_msg:
3396                                    ;     db 0Dh, 0Ah
3397                                    ;     db 07h
3398                                    ;     db 'Kernel initialization ERROR !'
3399                                    ;     db 0Dh, 0Ah, 0
3400
3401                                    ;welcome_msg:
3402                                    ;     db 0Dh, 0Ah
3403                                    ;     db 07h
3404                                    ;     db 'Welcome to TRDOS 386 Operating System !'
3405                                    ;     db 0Dh, 0Ah
3406                                    ;     db 'by Erdogan Tan - 31/12/2017 (v2.0.0)'
3407                                    ;     db 0Dh, 0Ah, 0
3408
3409                                    panic_msg:
3410 00014D17 0D0A07                    db 0Dh, 0Ah, 07h
3411 00014D1A 4552524F523A204B65-       db 'ERROR: Kernel Panic !'
3411 00014D23 726E656C2050616E69-
3411 00014D2C 632021
3412 00014D2F 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:
00014D32 5475726B6973682052-             db "Turkish Rational DOS v2.0 [15/02/2021] ...", 0
00014D3B 6174696F6E616C2044-
00014D44 4F532076322E30205B-
00014D4D 31352F30322F323032-
00014D56 315D202E2E2E00
                                    NextLine:
00014D5D 0D0A00                          db 0Dh, 0Ah, 0

                                    %include 'audio.s' ; 03/04/2017
                                <1> ; ****************************************************************************
                                <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.2 - audio.s
                                <1> ; ----------------------------------------------------------------------------
                                <1> ; Last Update: 01/09/2020
                                <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    03FFFFFFh ; 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>                                     ; 3D Audio Channel slots 3/4
                        <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_STOP_IDX     equ   10h    ;; RO ; 27/07/2020
                        <1>                                  ; current index = stop index
                        <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_REG_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> ; 22/07/2020
                        <1> ; REALTEK ALC655 and ADI SOUNDMAX AD1980 CODEC MIXER 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 ; AD1980
                        <1> CODEC_MASTER_MONO_VOL_REG       equ     06h  ; master mono volume (mono-out)
                        <1> ;CODEC_MASTER_TONE_REG          equ     08h  ; master tone (R+L) ; (not used)
                        <1> CODEC_PCBEEP_VOL_REG            equ     0Ah  ; PC beep volume ; ALC655
                        <1> CODEC_PHONE_VOL_REG             equ     0Ch  ; phone volume
                        <1> CODEC_MIC_VOL_REG               equ     0Eh  ; mic volume
                        <1> CODEC_LINE_IN_VOL_REG           equ     10h  ; line in volume
                        <1> CODEC_CD_VOL_REG                equ     12h  ; CD volume
                        <1> ;CODEC_VID_VOL_REG              equ     14h  ; video volume ; (not used)
                        <1> CODEC_AUX_VOL_REG               equ     16h  ; aux volume
                        <1> CODEC_PCM_OUT_REG               equ     18h  ; PCM out volume
                        <1> CODEC_RECORD_SELECT_REG         equ     1Ah  ; record select
                        <1> CODEC_RECORD_VOL_REG            equ     1Ch  ; record volume (record gain)
                        <1> ;CODEC_RECORD_MIC_VOL_REG       equ     1Eh  ; record mic volume ; (not used)
                        <1> CODEC_GP_REG                    equ     20h  ; general purpose
                        <1> ;CODEC_3D_CONTROL_REG           equ     22h  ; 3D control
                        <1> ;;CODEC_AUDIO_INT_PAGING_REG    equ    24h   ; audio int & paging ; (not used)
                        <1> CODEC_POWER_CTRL_REG            equ     26h  ; power down control
                        <1> CODEC_EXT_AUDIO_REG             equ     28h  ; extended audio ID
                        <1> CODEC_EXT_AUDIO_CTRL_REG        equ     2Ah  ; extended audio status/control
                        <1> CODEC_PCM_FRONT_DACRATE_REG     equ     2Ch  ; PCM front sample rate
                        <1> CODEC_PCM_SURND_DACRATE_REG     equ     2Eh  ; PCM surround sample rate
                        <1> CODEC_PCM_LFE_DACRATE_REG       equ     30h  ; PCM Center/LFE sample rate
                        <1> CODEC_LR_ADCRATE_REG            equ     32h  ; PCM input sample rate
                        <1> CODEC_MIC_ADCRATE_REG           equ     34h  ; mic in sample rate  ; AD1980
                        <1> CODEC_PCM_LFE_VOL_REG           equ     36h  ; PCM Center/LFE volume
                        <1> CODEC_PCM_SURND_VOL_REG         equ     38h  ; PCM surround volume
                        <1> ;CODEC_SPDIF_CTRL_REG           equ     3Ah  ; S/PDIF control
                        <1> ; 22/07/2020
                        <1> CODEC_MISC_CRTL_BITS_REG  equ    76h    ; misc control bits ; AD1980
                        <1> ;
                        <1> CODEC_VENDOR_ID1          equ    7Ch    ; REALTEK: 414Ch, ADI: 4144h
                        <1> CODEC_VENDOR_ID2          equ    7Eh    ; REALTEK: 4760h, ADI: 5370h
                        <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
00014D60 B886801524     <1>       mov     eax, (ICH_DID << 16) + INTEL_VID
00014D65 E876000000     <1>         call    pciFindDevice
00014D6A 730D           <1>         jnc     short d_ac97_1
                        <1> d_ac97_0:
                        <1> ; couldn't find the audio device!
00014D6C 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
00014D6D B806115930     <1>       mov     eax, (VT8233_DID << 16) + VIA_VID
00014D72 E869000000     <1>         call    pciFindDevice
                        <1> ;       jnc     short d_vt8233_0
                        <1> ; couldn't find the audio device!
                        <1> ;     retn
00014D77 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>
00014D79 A3[489C0100]   <1>       mov   [audio_dev_id], eax
00014D7E 8915[4C9C0100] <1>       mov   [audio_vendor], edx
                        <1>
                        <1>       ; init controller
00014D84 B004           <1>       mov   al, PCI_CMD_REG ; command register (04h)
00014D86 E8E2000000     <1>       call  pciRegRead32
```

```
                                <1>
                                <1>       ; eax = BUS/DEV/FN/REG
                                <1>       ; edx = STATUS/COMMAND
                                <1>       ;     SSSSSSSSSSSSSSSSCCCCCCCCCCCCCCCC
00014D8B 8915[509C0100]         <1>       mov   [audio_stats_cmd], edx
                                <1>
00014D91 B010                   <1>       mov   al, PCI_IO_BASE ; IO base address register (10h)
                                <1>       ;mov  al, NAMBAR_REG      ; Native Audio Mixer BAR (10h)
00014D93 E8D5000000             <1>       call  pciRegRead32
                                <1>
00014D98 66813D[4C9C0100]86-    <1>       cmp   word [audio_vendor], 8086h ; AC'97 ?
00014DA0 80                     <1>
00014DA1 751F                   <1>       jne   short d_vt8233_1
                                <1>
00014DA3 6683E2FE               <1>       and   dx, 0FFFEh ; Audio Codec IO_ADDR_MASK
00014DA7 668915[789C0100]       <1>       mov   [NAMBAR], dx
                                <1>
00014DAE B014                   <1>       mov   al, NABMBAR_REG ; Native Audio Bus Mastering BAR (14h)
00014DB0 E8B8000000             <1>       call  pciRegRead32
                                <1>
00014DB5 6683E2C0               <1>       and   dx, 0FFC0h ; Audio Controller IO_ADDR_MASK
00014DB9 668915[7A9C0100]       <1>       mov   [NABMBAR], dx
                                <1>         ;mov [audio_io_base], dx
                                <1>
00014DC0 EB0B                   <1>       jmp   short d_ac97_2
                                <1>
                                <1> d_vt8233_1:
00014DC2 6683E2C0               <1>       and    dx, 0FFC0h ; Audio Controller IO_ADDR_MASK
00014DC6 668915[469C0100]       <1>         mov    [audio_io_base], dx
                                <1>
                                <1> d_ac97_2:
                                <1>       ; 10/06/2017
00014DCD B03C                   <1>       mov   al, AC97_INT_LINE ; Interrupt Line Register (3Ch)
                                <1>       ;call pciRegRead32
00014DCF E886000000             <1>       call  pciRegRead8
                                <1>
                                <1>       ;and  edx, 0FFh
00014DD4 6681E2FF00             <1>       and   dx, 0FFh
                                <1>
00014DD9 8815[439C0100]         <1>       mov    [audio_intr], dl
                                <1>
00014DDF 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
00014DE0 50                     <1>       push   eax
                                <1>       ;push  esi
                                <1>       ;push  edi
                                <1>
00014DE1 89C6                   <1>         mov     esi, eax                ; save off vend+device ID
00014DE3 BF00FFFF7F             <1>         mov     edi, (80000000h - 100h) ; start with bus 0, dev 0 func 0
                                <1>
                                <1> nextPCIdevice:
00014DE8 81C700010000           <1>         add     edi, 100h
00014DEE 81FF00F8FF80           <1>         cmp     edi, 80FFF800h         ; scanned all devices?
00014DF4 F9                     <1>         stc
00014DF5 740C                   <1>         je      short PCIScanExit       ; not found
                                <1>
00014DF7 89F8                   <1>         mov     eax, edi                ; read PCI registers
00014DF9 E86F000000             <1>         call    pciRegRead32
00014DFE 39F2                   <1>         cmp     edx, esi                ; found device?
00014E00 75E6                   <1>         jne     short nextPCIdevice
00014E02 F8                     <1>         clc
                                <1>
                                <1> PCIScanExit:
00014E03 9C                     <1>       pushf
00014E04 B8FFFFFF7F             <1>       mov    eax, NOT_BIT31      ; 19/03/2017
00014E09 21F8                   <1>       and    eax, edi     ; return only bus/dev/fn #
00014E0B 9D                     <1>       popf
                                <1>
                                <1>       ;pop   edi
                                <1>       ;pop   esi
00014E0C 5A                     <1>       pop    edx
                                <1>       ;pop   ecx
00014E0D 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>
00014E0E 53                     <1>       push   ebx
00014E0F 51                     <1>       push   ecx
00014E10 89C3                   <1>         mov     ebx, eax        ; save eax, dh
00014E12 88F1                   <1>         mov     cl, dh
                                <1>
00014E14 25FFFFFF3F             <1>         and     eax, NOT_PCI32_PCI16   ; clear out data size request
00014E19 0D00000080             <1>         or      eax, BIT31             ; make a PCI access request
00014E1E 24FC                   <1>         and     al, ~3 ; NOT 3         ; force index to be dword
                                <1>
00014E20 66BAF80C               <1>         mov     dx, PCI_INDEX_PORT
00014E24 EF                     <1>         out  dx, eax                   ; write PCI selector
                                <1>
00014E25 66BAFC0C               <1>         mov     dx, PCI_DATA_PORT
00014E29 88D8                   <1>         mov     al, bl
00014E2B 2403                   <1>         and     al, 3                  ; figure out which port to
00014E2D 00C2                   <1>         add     dl, al                 ; read to
                                <1>
00014E2F F7C3000000C0           <1>       test    ebx, PCI32+PCI16
00014E35 7507                   <1>         jnz     short _pregr0
00014E37 EC                     <1>       in     al, dx             ; return 8 bits of data
00014E38 88C2                   <1>         mov  dl, al
00014E3A 88CE                   <1>       mov     dh, cl                 ; restore dh for 8 bit read
00014E3C EB12                   <1>       jmp    short _pregr2
                                <1> _pregr0:
00014E3E F7C300000080           <1>       test    ebx, PCI32
00014E44 7507                   <1>         jnz  short _pregr1
00014E46 66ED                   <1>       in     ax, dx
00014E48 6689C2                 <1>         mov     dx, ax                 ; return 16 bits of data
00014E4B EB03                   <1>       jmp    short _pregr2
                                <1> _pregr1:
00014E4D ED                     <1>       in     eax, dx                  ; return 32 bits of data
00014E4E 89C2                   <1>       mov    edx, eax
                                <1> _pregr2:
00014E50 89D8                   <1>       mov     eax, ebx          ; restore eax
00014E52 25FFFFFF3F             <1>         and     eax, NOT_PCI32_PCI16   ; clear out data size request
00014E57 59                     <1>       pop    ecx
00014E58 5B                     <1>       pop    ebx
00014E59 C3                     <1>       retn
                                <1>
                                <1> pciRegRead8:
```

```
00014E5A 25FFFFFF3F          <1>         and     eax, NOT_PCI32_PCI16   ; set up 8 bit read size
00014E5F EBAD                <1>         jmp     short pciRegRead; call generic PCI access
                             <1>
                             <1> pciRegRead16:
00014E61 25FFFFFF3F          <1>         and     eax, NOT_PCI32_PCI16   ; set up 16 bit read size
00014E66 0D00000040          <1>         or      eax, PCI16             ; call generic PCI access
00014E6B EBA1                <1>         jmp     short pciRegRead
                             <1>
                             <1> pciRegRead32:
00014E6D 25FFFFFF3F          <1>         and     eax, NOT_PCI32_PCI16   ; set up 32 bit read size
00014E72 0D00000080          <1>         or      eax, PCI32             ; call generic PCI access
00014E77 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>
00014E79 53                  <1>         push   ebx
00014E7A 51                  <1>         push   ecx
00014E7B 89C3                <1>         mov     ebx, eax        ; save eax, edx
00014E7D 89D1                <1>         mov     ecx, edx
00014E7F 25FFFFFF3F          <1>       and     eax, NOT_PCI32_PCI16     ; clear out data size request
00014E84 0D00000080          <1>         or      eax, BIT31             ; make a PCI access request
00014E89 24FC                <1>         and     al, ~3 ; NOT 3         ; force index to be dword
                             <1>
00014E8B 66BAF80C            <1>         mov     dx, PCI_INDEX_PORT
00014E8F EF                  <1>         out  dx, eax                   ; write PCI selector
                             <1>
00014E90 66BAFC0C            <1>         mov     dx, PCI_DATA_PORT
00014E94 88D8                <1>         mov     al, bl
00014E96 2403                <1>         and     al, 3                  ; figure out which port to
00014E98 00C2                <1>         add     dl, al                 ; write to
                             <1>
00014E9A F7C3000000C0        <1>       test    ebx, PCI32+PCI16
00014EA0 7505                <1>         jnz     short _pregw0
00014EA2 88C8                <1>       mov    al, cl                   ; put data into al
00014EA4 EE                  <1>       out    dx, al
00014EA5 EB12                <1>       jmp    short _pregw2
                             <1> _pregw0:
00014EA7 F7C300000080        <1>       test    ebx, PCI32
00014EAD 7507                <1>         jnz     short _pregw1
00014EAF 6689C8              <1>       mov    ax, cx             ; put data into ax
00014EB2 66EF                <1>       out    dx, ax
00014EB4 EB03                <1>       jmp    short _pregw2
                             <1> _pregw1:
00014EB6 89C8                <1>       mov    eax, ecx           ; put data into eax
00014EB8 EF                  <1>       out    dx, eax
                             <1> _pregw2:
00014EB9 89D8                <1>         mov     eax, ebx        ; restore eax
00014EBB 25FFFFFF3F          <1>         and     eax, NOT_PCI32_PCI16   ; clear out data size request
00014EC0 89CA                <1>         mov     edx, ecx        ; restore dx
00014EC2 59                  <1>       pop    ecx
00014EC3 5B                  <1>       pop    ebx
00014EC4 C3                  <1>       retn
                             <1>
                             <1> pciRegWrite8:
00014EC5 25FFFFFF3F          <1>         and     eax, NOT_PCI32_PCI16   ; set up 8 bit write size
00014ECA EBAD                <1>         jmp  short pciRegWrite   ; call generic PCI access
                             <1>
                             <1> pciRegWrite16:
00014ECC 25FFFFFF3F          <1>         and     eax, NOT_PCI32_PCI16   ; set up 16 bit write size
00014ED1 0D00000040          <1>         or      eax, PCI16             ; call generic PCI access
00014ED6 EBA1                <1>         jmp  short pciRegWrite
                             <1>
                             <1> pciRegWrite32:
00014ED8 25FFFFFF3F          <1>         and     eax, NOT_PCI32_PCI16   ; set up 32 bit write size
00014EDD 0D00000080          <1>         or      eax, PCI32             ; call generic PCI access
00014EE2 EB95                <1>         jmp  pciRegWrite
                             <1>
                             <1> init_codec:
                             <1>         ; 05/06/2017
                             <1>         ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)
                             <1>         ;
00014EE4 A1[489C0100]        <1>       mov    eax, [audio_dev_id]
00014EE9 B041                <1>       mov    al, VIA_ACLINK_CTRL
00014EEB E86AFFFFFF          <1>       call   pciRegRead8
                             <1>       ; ?
00014EF0 B040                <1>       mov    al, VIA_ACLINK_STAT
00014EF2 E863FFFFFF          <1>       call   pciRegRead8
00014EF7 F6C201              <1>       test   dl, VIA_ACLINK_C00_READY
00014EFA 7508                <1>         jnz     short _codec_ready_1
00014EFC E80E000000          <1>       call   reset_codec
00014F01 7306                <1>       jnc    short _codec_ready_2 ; eax = 1
00014F03 C3                  <1>       retn
                             <1> _codec_ready_1:
00014F04 B801000000          <1>       mov    eax, 1
                             <1> _codec_ready_2:
00014F09 E886000000          <1>       call   codec_io_w16
                             <1> detect_codec:
00014F0E 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)
00014F0F A1[489C0100]   <1>       mov    eax, [audio_dev_id]
00014F14 B041           <1>       mov    al, VIA_ACLINK_CTRL
00014F16 B2E0           <1>              mov    dl, VIA_ACLINK_CTRL_ENABLE + VIA_ACLINK_CTRL_RESET + VIA_ACLINK_CTRL_SYNC
00014F18 E8A8FFFFFF     <1>       call   pciRegWrite8
                        <1>
00014F1D E849000000     <1>       call   delay_100ms  ; wait 100 ms
                        <1> _rc_cold:
00014F22 E814000000     <1>         call    cold_reset
00014F27 7301           <1>         jnc     short _reset_codec_ok
                        <1>
                        <1>       ; 16/04/2017
                        <1>         ;xor     eax, eax         ; timeout error
                        <1>             ;stc
00014F29 C3             <1>       retn
                        <1>
                        <1> _reset_codec_ok:
                        <1>       ; 01/09/2020
                        <1>       ; 15/08/2020
                        <1>       ; 27/07/2020
                        <1>       ; also reset codec by using index control register 0 of AD1980 or ALC655
                        <1>       ; (to fix line out -2 channels audio playing- problem on AD1980 codec)
                        <1>
00014F2A 29C0           <1>       sub    eax, eax
00014F2C BA00000000     <1>       mov    edx, CODEC_RESET_REG ; 00h ; Reset register
00014F31 E8CA000000     <1>       call   codec_write
                        <1>
                        <1>       ;sub   eax, eax
                        <1>       ; 01/09/2020
                        <1>       ; 15/08/2020
                        <1>       ; AD1980 BugFix
                        <1>       ; (set HPSEL -headphone amp to be driven from mixer- and
                        <1>       ;      CLDIS - center and LFE disable- bits)
                        <1>       ;mov   eax, 0C00h ; HPSEL = bit 10, CLDIS = bit 11 ; 01/09/2020
                        <1>       ;mov   edx, CODEC_MISC_CRTL_BITS_REG ; 76h ; Misc Ctrl Bits ; AD1980
                        <1>       ;call  codec_write
                        <1>
00014F36 31C0           <1>         xor     eax, eax
                        <1>         ;mov al, VIA_ACLINK_C00_READY ; 1
00014F38 FEC0           <1>         inc  al
00014F3A 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
00014F3B 30D2           <1>       xor    dl, dl ; 0
00014F3D E883FFFFFF     <1>       call   pciRegWrite8
                        <1>
00014F42 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
00014F47 B2CC           <1>       mov    dl, VIA_ACLINK_CTRL_INIT
00014F49 E877FFFFFF     <1>       call   pciRegWrite8
                        <1>
00014F4E B910000000     <1>       mov    ecx, 16      ; total 2s
                        <1>
                        <1> _crst_wait:
                        <1>       ;mov   eax, [audio_dev_id]
00014F53 B040           <1>       mov    al, VIA_ACLINK_STAT
00014F55 E800FFFFFF     <1>       call   pciRegRead8
                        <1>
00014F5A F6C201         <1>         test    dl, VIA_ACLINK_C00_READY
00014F5D 750B           <1>         jnz     short _crst_ok
                        <1>
00014F5F 51             <1>       push   ecx
00014F60 E806000000     <1>       call   delay_100ms
00014F65 59             <1>       pop    ecx
                        <1>
00014F66 49             <1>         dec     ecx
00014F67 75EA           <1>         jnz     short _crst_wait
                        <1>
                        <1> _crst_fail:
00014F69 F9             <1>         stc
                        <1> _crst_ok:
00014F6A C3             <1>       retn
                        <1>
                        <1> delay_100ms:
                        <1>       ; 29/05/2017
                        <1>       ; 24/03/2017 ('codec.asm')
                        <1>       ; wait 100 ms
00014F6B B990010000     <1>       mov    ecx, 400  ; 400*0.25ms
                        <1> _delay_x_ms:
00014F70 E803000000     <1>       call   delay1_4ms
00014F75 E2F9           <1>         loop _delay_x_ms
00014F77 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:
00014F78 50             <1>         push    eax
00014F79 51             <1>         push    ecx
00014F7A B110           <1>         mov  cl, 16       ; close enough.
                        <1>
00014F7C E461           <1>       in     al, PORTB ; 61h
                        <1>
00014F7E 2410           <1>       and    al, REFRESH_STATUS ; 10h
00014F80 88C5           <1>       mov    ch, al       ; Start toggle state
                        <1> _d4ms1:
00014F82 E461           <1>       in     al, PORTB    ; Read system control port
                        <1>
00014F84 2410           <1>       and    al, REFRESH_STATUS ; Refresh toggles 15.085 microseconds
00014F86 38C5           <1>       cmp    ch, al
00014F88 74F8           <1>       je     short _d4ms1 ; Wait for state change
                        <1>
00014F8A 88C5           <1>       mov    ch, al       ; Update with new state
00014F8C FEC9           <1>       dec    cl
00014F8E 75F2           <1>       jnz    short _d4ms1
                        <1>
00014F90 F8             <1>       clc    ; 29/05/2017
                        <1>
00014F91 59             <1>         pop     ecx
00014F92 58             <1>         pop     eax
00014F93 C3             <1>         retn
                        <1>
                        <1> ; 10/04/2017 (TRDOS 386)
                        <1> ; 12/11/2016
                        <1>
                        <1> codec_io_w16: ;w32
                        <1>       ; ('codec.asm')
00014F94 668B15[469C0100]   <1>         mov  dx, [audio_io_base]
00014F9B 6681C28000         <1>         add     dx, VIA_REG_AC97
00014FA0 EF                 <1>       out    dx, eax
00014FA1 C3                 <1>         retn
                        <1>
                        <1> codec_io_r16: ;r32
                        <1>       ; ('codec.asm')
00014FA2 668B15[469C0100]   <1>         mov     dx, [audio_io_base]
00014FA9 6681C28000         <1>         add     dx, VIA_REG_AC97
00014FAE ED                 <1>         in   eax, dx
00014FAF C3                 <1>         retn
                        <1>
                        <1> ctrl_io_w8:
                        <1>       ; ('codec.asm')
00014FB0 660315[469C0100]   <1>         add     dx, [audio_io_base]
00014FB7 EE                 <1>         out  dx, al
00014FB8 C3                 <1>         retn
                        <1>
                        <1> ctrl_io_r8:
                        <1>       ; ('codec.asm')
00014FB9 660315[469C0100]   <1>         add     dx, [audio_io_base]
00014FC0 EC                 <1>         in   al, dx
00014FC1 C3                 <1>         retn
                        <1>
                        <1> ctrl_io_w32:
                        <1>       ; ('codec.asm')
00014FC2 660315[469C0100]   <1>         add     dx, [audio_io_base]
00014FC9 EF                 <1>         out  dx, eax
00014FCA C3                 <1>         retn
                        <1>
                        <1> ctrl_io_r32:
                        <1>       ; ('codec.asm')
00014FCB 660315[469C0100]   <1>         add  dx, [audio_io_base]
00014FD2 ED                 <1>       in     eax, dx
00014FD3 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
00014FD4 C1E010             <1>         shl     eax, VIA_REG_AC97_CMD_SHIFT
00014FD7 0D00008002         <1>         or      eax, VIA_REG_AC97_PRIMARY_VALID + VIA_REG_AC97_READ
                        <1>
00014FDC E8B3FFFFFF         <1>       call    codec_io_w16
                        <1>
                        <1>             ; codec_valid
00014FE1 E831000000         <1>       call   codec_check_ready
00014FE6 7301               <1>         jnc  short _cr_ok
                        <1>
00014FE8 C3                 <1>       retn
                        <1>
                        <1> _cr_ok:
                        <1>       ; wait 25 ms
00014FE9 B950000000         <1>       mov    ecx, 80 ; (100*0.25 ms)
                        <1> _cr_wloop:
00014FEE E885FFFFFF         <1>       call   delay1_4ms
00014FF3 E2F9               <1>       loop   _cr_wloop
                        <1>
00014FF5 E8A8FFFFFF         <1>         call    codec_io_r16
00014FFA 25FFFF0000         <1>         and     eax, 0FFFFh
00014FFF 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>
00015000 C1E210         <1>       shl     edx, VIA_REG_AC97_CMD_SHIFT
                        <1>
00015003 C1E000         <1>         shl     eax, VIA_REG_AC97_DATA_SHIFT ; shl eax, 0
00015006 09C2           <1>         or      edx, eax
                        <1>
00015008 B800000000     <1>         mov     eax, VIA_REG_AC97_CODEC_ID_PRIMARY
0001500D C1E01E         <1>         shl     eax, VIA_REG_AC97_CODEC_ID_SHIFT
00015010 09D0           <1>         or      eax, edx
                        <1>
00015012 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:
00015017 B914000000     <1>       mov    ecx, 20      ; total 2s
                        <1> _ccr_wait:
0001501C 51             <1>       push   ecx
                        <1>
0001501D E880FFFFFF     <1>         call    codec_io_r16
00015022 A900000001     <1>         test    eax, VIA_REG_AC97_BUSY
00015027 740B           <1>         jz      short _ccr_ok
                        <1>
00015029 E83DFFFFFF     <1>       call   delay_100ms
                        <1>
0001502E 59             <1>       pop    ecx
                        <1>
0001502F 49             <1>       dec     ecx
00015030 75EA           <1>         jnz     short _ccr_wait
                        <1>
00015032 F9             <1>         stc
00015033 C3             <1>         retn
                        <1>
                        <1> _ccr_ok:
00015034 59             <1>       pop    ecx
00015035 25FFFF0000     <1>       and     eax, 0FFFFh
0001503A 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>
0001503B B802020000     <1>       mov     eax, 0202h
00015040 66A3[769C0100] <1>       mov    [audio_master_volume], ax
00015046 66B81F1F       <1>       mov    ax, 1F1Fh ; 31,31
0001504A BA02000000     <1>       mov    edx, CODEC_MASTER_VOL_REG ; 02h ; Line Out
0001504F E8ACFFFFFF     <1>       call   codec_write
                        <1>       ;jc    short cconfig_error
                        <1>
                        <1>       ;mov    eax, 0202h
00015054 66B80202       <1>       mov     ax, 0202h
00015058 BA18000000     <1>       mov    edx, CODEC_PCM_OUT_REG ; 18h ; Wave Output (Stereo)
0001505D E89EFFFFFF     <1>       call   codec_write
                        <1>       ;jc    short cconfig_error
                        <1>
                        <1>       ;mov    eax, 0202h
00015062 66B80202       <1>       mov    ax, 0202h
00015066 BA04000000     <1>       mov    edx, CODEC_AUX_VOL ; 04h ; CODEC_HP_VOL_REG ; HeadPhone
0001506B E890FFFFFF     <1>       call   codec_write
                        <1>       ;jc    short cconfig_error
                        <1>
                        <1>       ;mov    eax, 08h
                        <1>         ;mov    ax,  08h
00015070 66B80880       <1>       mov    ax, 8008h ; Mute
00015074 BA0C000000     <1>       mov    edx, 0Ch  ; AC97_PHONE_VOL ; TAD Input (Mono)
00015079 E882FFFFFF     <1>       call   codec_write
                        <1>       ;jc    short cconfig_error
                        <1>
                        <1>       ;mov    eax, 0808h
0001507E 66B80808       <1>         mov     ax,  0808h
00015082 BA10000000     <1>         mov  edx, CODEC_LINE_IN_VOL_REG ; 10h ; Line Input (Stereo)
00015087 E874FFFFFF     <1>       call   codec_write
                        <1>       ;jc    short cconfig_error
                        <1>
                        <1>       ;mov    eax, 0808h
0001508C 66B80808       <1>       mov     ax,  0808h
00015090 BA12000000     <1>         mov  edx, CODEC_CD_VOL_REG ; 12h ; CR Input (Stereo)
00015095 E866FFFFFF     <1>       call   codec_write
                        <1>       ;jc    short cconfig_error
                        <1>
                        <1>       ;mov    eax, 0808h
0001509A 66B80808       <1>       mov     ax,  0808h
0001509E BA16000000     <1>         mov  edx, CODEC_AUX_VOL_REG ; 16h ; Aux Input (Stereo)
                        <1>       ;call  codec_write
                        <1>       ;;jc   short cconfig_error
000150A3 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>         ; 27/07/2020
                                <1>         ; 22/07/2020
                                <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>
000150A8 66BA0000               <1>           mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STATUS
000150AC E808FFFFFF             <1>           call    ctrl_io_r8
                                <1>
000150B1 A880                   <1>         test    al, VIA_REG_STAT_ACTIVE
000150B3 7417                   <1>           jz      short _ih0 ; 09/10/2017
                                <1>
000150B5 2407                   <1>           and     al, VIA_REG_STAT_EOL + VIA_REG_STAT_FLAG + VIA_REG_STAT_STOPPED
000150B7 A2[759C0100]           <1>         mov   [audio_flag_eol], al
000150BC 740E                   <1>           jz  short _ih0 ; 09/10/2017
                                <1>
                                <1>         ; 09/10/2017
                                <1>         ;mov  byte [audio_busy], 1
                                <1>
000150BE 803D[749C0100]01       <1>         cmp   byte [audio_play_cmd], 1
000150C5 7315                   <1>         jnb   short _ih1 ; 10/10/2017
                                <1>
000150C7 E84A000000             <1>         call  channel_reset
                                <1> _ih0:
                                <1>         ; 09/10/2017
000150CC A0[759C0100]           <1>           mov     al, [audio_flag_eol]   ;; ack ;;
000150D1 66BA0000               <1>           mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STATUS
000150D5 E8D6FEFFFF             <1>           call    ctrl_io_w8
000150DA EB39                   <1>         jmp   short _ih4
                                <1> _ih1:
                                <1> vt8233_tuneLoop:
000150DC A0[759C0100]           <1>           mov     al, [audio_flag_eol]   ;; ack ;;
000150E1 66BA0000               <1>           mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STATUS
000150E5 E8C6FEFFFF             <1>           call    ctrl_io_w8
                                <1>
                                <1>         ; 22/07/2020
                                <1>         ;; 12/10/2017
                                <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>         ; 22/07/2020
                                <1>         ; 14/10/2017
                                <1>         ;test  byte [audio_flag_eol], VIA_REG_STAT_EOL
                                <1>         ;jnz   short _ih2 ; EOL
                                <1>         ;               ; (Half Buffer 2 has been completed
                                <1>         ;               ; and Half Buffer 1 will be played.)
                                <1>
                                <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
                                <1>
                                <1>         ; 22/07/2020
                                <1>         ;inc   byte [audio_flag] ; = 1
                                <1> _ih2:
                                <1>         ; 10/10/2017
000150EA 8B3D[609C0100]         <1>         mov   edi, [audio_dma_buff]
000150F0 8B0D[649C0100]         <1>         mov   ecx, [audio_dmabuff_size]
```

```
1037 000150F6 D1E9               <1>       shr    ecx, 1 ; dma buff size / 2 = half buffer size
1038                             <1>
1039                             <1>       ; 22/07/2020
1040                             <1>       ; 12/10/2017
1041                             <1>       ;cmp   byte [audio_flag], 0
1042                             <1>       ;ja    short _ih3 ; Playing Half Buffer 2 (Current: FLAG)
1043                             <1>
1044                             <1>       ; 27/07/2020
1045                             <1>       ; 22/07/2020
1046 000150F8 F605[689C0100]01   <1>       test   byte [audio_flag], 1  ; Current flag value
1047 000150FF 7402               <1>       jz     short _ih3 ; Half Buffer 1 must be filled
1048                             <1>
1049                             <1>       ; Half Buffer 2 must be filled
1050 00015101 01CF               <1>       add    edi, ecx
1051                             <1> _ih3:
1052                             <1>       ; Update half buffer 2 while playing half buffer 1
1053                             <1>       ; Update half buffer 1 while playing half buffer 2
1054                             <1>
1055 00015103 8B35[589C0100]     <1>       mov    esi, [audio_p_buffer] ; phy addr of audio buff
1056 00015109 C1E902             <1>       shr    ecx, 2 ; half buff size / 4
1057 0001510C F3A5               <1>       rep    movsd
1058                             <1>
1059                             <1>       ; switch flag value ;
1060 0001510E 8035[689C0100]01   <1>       xor    byte [audio_flag], 1
1061                             <1>       ; 12/10/2017
1062                             <1>       ; [audio_flag] = 0 : Playing dma half buffer 2
1063                             <1>                          ; Next buffer (to update) is dma half buff 1
1064                             <1>       ;             = 1 : Playing dma half buffer 1
1065                             <1>                            ; Next buffer (to update) is dma half buff 2
1066                             <1> _ih4:
1067                             <1>       ; 28/05/2017
1068                             <1>       ;mov   byte [audio_busy], 0 ; 09/10/2017
1069                             <1>       ;
1070                             <1>       ;pop   edi
1071                             <1>       ;pop   esi
1072                             <1>       ;pop   ebx ; * must be restored !
1073                             <1>       ;pop   ecx
1074                             <1>       ;pop   edx
1075                             <1>       ;pop   eax ; * must be restored !
1076                             <1>
1077 00015115 C3                <1>       retn
1078                             <1>
1079                             <1> channel_reset:
1080                             <1>       ; 24/06/2017
1081                             <1>       ; 29/05/2017
1082                             <1>       ; 23/03/2017
1083                             <1>       ; 14/11/2016 - Erdogan Tan
1084                             <1>       ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)
1085 00015116 BA01000000         <1>         mov  edx, VIA_REG_OFFSET_CONTROL
1086                             <1>         ;mov eax, VIA_REG_CTRL_PAUSE + VIA_REG_CTRL_TERMINATE + VIA_REG_CTRL_RESET
1087 0001511B B848000000         <1>         mov  eax, VIA_REG_CTRL_PAUSE + VIA_REG_CTRL_TERMINATE ; 24/06/2017
1088 00015120 E88BFEFFFF         <1>       call    ctrl_io_w8
1089                             <1>
1090                             <1>         ;mov edx, VIA_REG_OFFSET_CONTROL
1091                             <1>         ;call   ctrl_io_r8
1092                             <1>
1093                             <1>       ; wait for 50 ms
1094 00015125 B9A0000000         <1>       mov    ecx, 160 ; (200*0.25 ms) ; 29/05/2017
1095                             <1> _ch_rst_wait:
1096 0001512A E849FEFFFF         <1>       call   delay1_4ms
1097 0001512F 49                 <1>       dec    ecx
1098 00015130 75F8               <1>       jnz    short _ch_rst_wait
1099                             <1>
1100                             <1>         ; disable interrupts
1101 00015132 BA01000000         <1>         mov  edx, VIA_REG_OFFSET_CONTROL
1102 00015137 31C0               <1>         xor     eax, eax
1103 00015139 E872FEFFFF         <1>         call    ctrl_io_w8
1104                             <1>
1105                             <1>         ; clear interrupts
1106 0001513E BA00000000         <1>         mov  edx, VIA_REG_OFFSET_STATUS
1107 00015143 B803000000         <1>       mov    eax, 3
1108 00015148 E863FEFFFF         <1>         call ctrl_io_w8
1109                             <1>
1110                             <1>       ;mov   edx, VIA_REG_OFFSET_CURR_PTR
1111                             <1>       ;xor   eax, eax
1112                             <1>       ;call  ctrl_io_w32
1113                             <1>
1114 0001514D C3                <1>         retn
1115                             <1>
1116                             <1> vt8233_stop: ; 22/04/2017
1117 0001514E C605[749C0100]00   <1>       mov    byte [audio_play_cmd], 0 ; stop !
1118                             <1> _tlp2:
1119                             <1>       ; 24/06/2017
1120                             <1>         ; finished with song, stop everything
1121                             <1>       ;mov   al, VIA_REG_CTRL_INT
1122                             <1>         ;or  al, VIA_REG_CTRL_TERMINATE
1123                             <1>       ;mov   dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_CONTROL
1124                             <1>         ;call      ctrl_io_w8
1125                             <1>
1126                             <1>         ;call      channel_reset
1127                             <1>       ;retn
1128                             <1>
1129 00015155 EBBF               <1>       jmp    short channel_reset
1130                             <1>
1131                             <1> set_vt8233_bdl: ; Set VT8237R Buffer Descriptor List
1132                             <1>       ; 22/07/2020 - TRDOS 386 v2.0.2
1133                             <1>       ; 28/05/2017
1134                             <1>       ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')
1135                             <1>       ; 24/03/2017 - 'PLAYER.COM' ('via_wav.asm' - 29/11/2016)
1136                             <1>
1137                             <1>       ; eax = dma buffer address = [audio_DMA_buff]
1138                             <1>       ; ecx = dma buffer buffer size = [audio_dmabuff_size]
1139                             <1>
1140 00015157 D1E9               <1>       shr    ecx, 1 ; dma half buffer size
1141 00015159 89CE               <1>       mov    esi, ecx
```

```
1142                                    <1>
1143 0001515B BF[7C9C0100]             <1>         mov     edi, audio_bdl_buff    ; get BDL address
1144 00015160 B910000000               <1>         mov     ecx, 32 / 2            ; make 32 entries in BDL
1145                                    <1>
1146 00015165 EB05                     <1>       jmp    short s_vt8233_bdl1
1147                                    <1>
1148                                    <1> s_vt8233_bdl0:
1149                                    <1>       ; set buffer descriptor 0 to start of data file in memory
1150                                    <1>
1151 00015167 A1[609C0100]             <1>       mov    eax, [audio_dma_buff]     ; Physical address of DMA buffer
1152                                    <1>
1153                                    <1> s_vt8233_bdl1:
1154 0001516C AB                       <1>       stosd                      ; store dmabuffer1 address
1155                                    <1>
1156 0001516D 89C2                     <1>       mov    edx, eax
1157                                    <1>
1158                                    <1> ; VIA VT8235.PDF: (Page 110) (Erdogan Tan, 29/11/2016)
1159                                    <1>       ;
1160                                    <1>       ;      Audio SGD Table Format
1161                                    <1>       ;      -------------------------------
1162                                    <1>       ;      63   62    61-56    55-32  31-0
1163                                    <1>       ;      --   --   --------  -----  ----
1164                                    <1>       ;      EOL FLAG -reserved- Base   Base
1165                                    <1>       ;                         Count  Address
1166                                    <1>       ;                          [23:0] [31:0]
1167                                    <1>       ;      EOL: End Of Link.
1168                                    <1>       ;           1 indicates this block is the last of the link.
1169                                    <1>       ;           If the channel "Interrupt on EOL" bit is set, then
1170                                    <1>       ;           an interrupt is generated at the end of the transfer.
1171                                    <1>       ;
1172                                    <1>       ;      FLAG: Block Flag. If set, transfer pauses at the end of this
1173                                    <1>       ;            block. If the channel "Interrupt on FLAG" bit is set,
1174                                    <1>       ;            then an interrupt is generated at the end of this block.
1175                                    <1>
1176 0001516F 89F0                     <1>       mov    eax, esi ; DMA half buffer size
1177 00015171 01C2                     <1>       add    edx, eax
1178 00015173 0D00000040               <1>       or     eax, FLAG
1179                                    <1>       ;or    eax, EOL
1180 00015178 AB                       <1>       stosd
1181                                    <1>
1182                                    <1> ; 2nd buffer:
1183                                    <1>
1184 00015179 89D0                     <1>         mov  eax, edx ; Physical address of the 2nd half of DMA buffer
1185 0001517B AB                       <1>       stosd         ; store dmabuffer2 address
1186                                    <1>
1187                                    <1> ; set length to [audio_dmabuff_size]/2
1188                                    <1> ; Set control (bits 31:16) to BUP, bits 15:0=number of samples
1189                                    <1> ;
1190 0001517C 89F0                     <1>       mov    eax, esi ; DMA half buffer size
1191                                    <1>       ; 22/07/2020
1192                                    <1>       ;or    eax, EOL
1193 0001517E 0D00000040               <1>       or     eax, FLAG
1194 00015183 AB                       <1>       stosd
1195                                    <1>
1196 00015184 E2E1                     <1>       loop    s_vt8233_bdl0
1197                                    <1>
1198                                    <1>       ; 22/07/2020
1199 00015186 814FFC00000080           <1>       or     dword [edi-4], EOL
1200                                    <1>
1201 0001518D C3                       <1>       retn
1202                                    <1>
1203                                    <1> vt8233_start_play:
1204                                    <1>       ; 01/09/2020
1205                                    <1>       ; 22/07/2020
1206                                    <1>       ; start to play audio data via VT8233 audio controller
1207                                    <1>       ; 13/06/2017
1208                                    <1>       ; 10/06/2017
1209                                    <1>       ; 24/04/2017
1210                                    <1>       ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')
1211                                    <1>       ; 24/03/2017 - 'PLAYER.COM' ('via_wav.asm' - 29/11/2016)
1212                                    <1>       ; write buffer descriptor list address
1213                                    <1>
1214                                    <1>       ; Extended Audio Status (2Ah)
1215 0001518E B82A000000               <1>       mov    eax, CODEC_EXT_AUDIO_CTRL_REG ; 2Ah
1216 00015193 E83CFEFFFF               <1>       call   codec_read
1217 00015198 25FDFF0000               <1>       and     eax, 0FFFFh - 2          ; clear DRA (BIT1)
1218                                    <1>       ;or     eax, 1                   ; set VRA (BIT0)
1219                                    <1>       ;or    eax, 5       ; VRA (BIT0) & S/PDIF (BIT2) ; 14/11/2016
1220 0001519D 0C05                     <1>       or     al, 5
1221                                    <1>       ; 01/09/2020
1222                                    <1>       ;or    eax, 3805h ; AD1980 (PRK, PRJ, PRI = 1 .. only front DAC)
1223                                    <1>       ; 01/09/2020
1224                                    <1>       ;mov   edx, CODEC_EXT_AUDIO_CTRL_REG
1225                                    <1>       ;cmp   word [audio_freq], 0BB80h ; 48 kHz
1226                                    <1>       ;jne   short set_extd_audio_status_1
1227                                    <1>       ;and   al, 0FEh ; disable VRA bit (set sample rate to 48000 Hz)
1228                                    <1>       ;jmp   short set_extd_audio_status_2
1229                                    <1> ;set_extd_audio_status_1:
1230 0001519F BA2A000000               <1>       mov    edx, CODEC_EXT_AUDIO_CTRL_REG
1231 000151A4 E857FEFFFF               <1>       call   codec_write
1232                                    <1>       ;jc    short cconfig_error
1233                                    <1>
1234                                    <1> set_sample_rate:
1235                                    <1>       ;movzx eax, word [audio_freq]
1236 000151A9 66A1[729C0100]           <1>       mov    ax, [audio_freq]
1237 000151AF BA2C000000               <1>       mov    edx, CODEC_PCM_FRONT_DACRATE_REG ; 2Ch ; PCM Front DAC Rate
1238                                    <1> ;set_extd_audio_status_2:
1239 000151B4 E847FEFFFF               <1>       call   codec_write
1240                                    <1>
1241                                    <1>       ; 01/09/2020
1242                                    <1>       ; set AD1980 MCB register (Index 76h) to 0C00h
1243                                    <1>       ; (CLDIS, HPSEL)
1244                                    <1>       ;mov   ax, 0C00h
1245                                    <1>       ;mov   edx, CODEC_MISC_CRTL_BITS_REG ; 76h
1246                                    <1>       ;                  ; Miscellaneous Control Bit Register
```

```
1247                                    <1>       ;call  codec_write
1248                                    <1>       ;
1249                                    <1>
1250 000151B9 B8[7C9C0100]              <1>         mov  eax, audio_bdl_buff
1251                                    <1>
1252                                    <1>       ; 12/11/2016 - Erdogan Tan
1253                                    <1>       ; (Ref: KolibriOS, vt823x.asm, 'create_primary_buff')
1254 000151BE BA04000000                <1>       mov    edx, VIADEV_PLAYBACK + VIA_REG_OFFSET_TABLE_PTR
1255 000151C3 E8FAFDFFFF                <1>         call ctrl_io_w32
1256                                    <1>
1257                                    <1>       ;call  codec_check_ready
1258                                    <1>
1259 000151C8 66BA0200                  <1>       mov    dx, VIADEV_PLAYBACK + VIA_REG_OFS_PLAYBACK_VOLUME_L
1260                                    <1>         ;mov eax, 2 ; 31
1261 000151CC B01F                      <1>       mov    al, 31
1262 000151CE 2A05[769C0100]            <1>         sub  al, [audio_master_volume_l]
1263 000151D4 E8D7FDFFFF                <1>       call   ctrl_io_w8
1264                                    <1>
1265                                    <1>       ;call  codec_check_ready
1266                                    <1>
1267 000151D9 66BA0300                  <1>         mov     dx, VIADEV_PLAYBACK + VIA_REG_OFS_PLAYBACK_VOLUME_R
1268                                    <1>         ;mov ax, 2  ; 31
1269 000151DD B01F                      <1>       mov    al, 31
1270 000151DF 2A05[779C0100]            <1>         sub  al, [audio_master_volume_r]
1271 000151E5 E8C6FDFFFF                <1>       call    ctrl_io_w8
1272                                    <1>
1273                                    <1>       ;call  codec_check_ready
1274                                    <1> ;
1275                                    <1> ;
1276                                    <1> ; All set. Let's play some music.
1277                                    <1> ;
1278                                    <1> ;
1279                                    <1>             ;mov    dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STOP_IDX
1280                                    <1>         ;mov    ax, VIA8233_REG_TYPE_16BIT or VIA8233_REG_TYPE_STEREO or 0xfffff or 0xff000000
1281                                    <1>         ;call   ctrl_io_w32
1282                                    <1>
1283                                    <1>       ;call  codec_check_ready
1284                                    <1>
1285                                    <1>       ; 08/12/2016
1286                                    <1>       ; 07/10/2016
1287                                    <1>         ;;mov    al, 1
1288                                    <1>         ;mov al, 31
1289                                    <1>       ; 22/07/2020
1290 000151EA B0FF                      <1>       mov    al, 0FFh
1291 000151EC E8130000000               <1>       call    set_VT8233_LastValidIndex
1292                                    <1>
1293 000151F1 C605[749C0100]01          <1>       mov    byte [audio_play_cmd], 1 ; play command (do not stop) !
1294                                    <1>
1295                                    <1>       ; 22/07/2020
1296                                    <1>       ;mov   byte [audio_flag], 0  ; clear half buffer flag
1297                                    <1>
1298                                    <1> vt8233_play: ; continue to play
1299                                    <1>       ; 22/04/2017
1300                                    <1>       ;mov   al, VIA_REG_CTRL_INT
1301                                    <1>             ;or    al, VIA_REG_CTRL_START
1302                                    <1>         ;;mov      al, VIA_REG_CTRL_AUTOSTART + VIA_REG_CTRL_START
1303                                    <1>       ; 22/07/2020
1304 000151F8 B0A1                      <1>       mov    al, VIA_REG_CTRL_AUTOSTART + VIA_REG_CTRL_START + VIA_REG_CTRL_INT_FLAG
1305                                    <1>
1306 000151FA 66BA0100                  <1>       mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_CONTROL
1307 000151FE E8ADFDFFFF                <1>         call    ctrl_io_w8
1308                                    <1>       ;call  codec_check_ready
1309                                    <1>       ;retn
1310                                    <1>       ;jmp   codec_check_ready
1311 00015203 C3                        <1>       retn
1312                                    <1>
1313                                    <1> ;input AL = index # to stop on
1314                                    <1> set_VT8233_LastValidIndex:
1315                                    <1>       ; 23/07/2020
1316                                    <1>       ; 10/06/2017
1317                                    <1>       ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')
1318                                    <1>       ; 24/03/2017 - 'PLAYER.COM' ('via_wav.asm' - 29/11/2016)
1319                                    <1>       ; 19/11/2016
1320                                    <1>       ; 14/11/2016 - Erdogan Tan (Ref: VIA VT8235.PDF, Page 110)
1321                                    <1>       ; 12/11/2016 - Erdogan Tan
1322                                    <1>       ; (Ref: KolibriOS, vt823x.asm, 'create_primary_buff')
1323                                    <1>       ;push  edx
1324                                    <1>       ;push  ax
1325 00015204 50                        <1>       push   eax ; 23/07/2020
1326                                    <1>       ;push  ecx
1327 00015205 0FB705[729C0100]          <1>       movzx  eax, word [audio_freq] ; Hertz
1328 0001520C BA00001000                <1>       mov    edx, 100000h ; 2^20 = 1048576
1329 00015211 F7E2                      <1>       mul    edx
1330 00015213 B980BB0000                <1>       mov    ecx, 48000
1331 00015218 F7F1                      <1>       div    ecx
1332                                    <1>       ;and   eax, 0FFFFFh
1333                                    <1>       ;pop   ecx
1334                                    <1>       ;pop   dx
1335 0001521A 5A                        <1>       pop    edx ; 23/07/2020
1336 0001521B C1E218                    <1>       shl    edx, 24  ; STOP Index Setting: Bit 24 to 31
1337 0001521E 09D0                      <1>       or     eax, edx
1338                                    <1>       ; 19/11/2016
1339 00015220 803D[709C0100]10          <1>       cmp    byte [audio_bps], 16
1340 00015227 7505                      <1>       jne    short sLVI_1
1341 00015229 0D00002000                <1>       or     eax, VIA8233_REG_TYPE_16BIT
1342                                    <1> sLVI_1:
1343 0001522E 803D[719C0100]02          <1>       cmp    byte [audio_stmo], 2
1344 00015235 7505                      <1>       jne    short sLVI_2
1345 00015237 0D00001000                <1>       or     eax, VIA8233_REG_TYPE_STEREO
1346                                    <1> sLVI_2:
1347 0001523C BA08000000                <1>       mov     edx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STOP_IDX
1348 00015241 E87CFDFFFF                <1>         call    ctrl_io_w32
1349                                    <1>       ;call  codec_check_ready
1350                                    <1>       ;pop   edx
1351 00015246 C3                        <1>       retn
```

```
                                <1>
                                <1> vt8233_pause: ; pause
                                <1>       ; 10/06/2017
                                <1>       ; 22/04/2017
                                <1>       ;mov     al, VIA_REG_CTRL_INT
                                <1>         ;or      al, VIA_REG_CTRL_PAUSE
                                <1>       ; 23/07/2020
00015247 B029                   <1>       mov    al, VIA_REG_CTRL_PAUSE+VIA_REG_CTRL_INT_FLAG+VIA_REG_CTRL_AUTOSTART
                                <1>
00015249 66BA0100               <1>       mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_CONTROL
0001524D E85EFDFFFF             <1>         call    ctrl_io_w8
                                <1>       ;call  codec_check_ready
                                <1>       ;retn
                                <1>       ;jmp   codec_check_ready
00015252 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
00015253 C605[749C0100]00       <1>       mov    byte [audio_play_cmd], 0 ; stop !
                                <1> vt8233_rst_0:
0001525A E8B0FCFFFF             <1>       call   reset_codec
0001525F 720A                   <1>       jc     short vt8233_rst_1 ; codec error !
                                <1>       ; eax = 1
00015261 E82EFDFFFF             <1>       call   codec_io_w16 ; w32
00015266 E8ABFEFFFF             <1>       call   channel_reset
                                <1> vt8233_rst_1:
0001526B 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>
0001526C 08DB                   <1>       or     bl, bl
0001526E 7520                   <1>       jnz    short vt8233_vol_1 ; temporary !
00015270 66B81F1F               <1>       mov    ax, 1F1Fh ; 31,31
00015274 38C1                   <1>       cmp    cl, al
00015276 7718                   <1>       ja     short vt8233_vol_1 ; temporary !
00015278 38E5                   <1>       cmp    ch, ah
0001527A 7714                   <1>       ja     short vt8233_vol_1 ; temporary !
0001527C 66890D[769C0100]       <1>       mov    [audio_master_volume], cx
00015283 6629C8                 <1>       sub    ax, cx
00015286 BA02000000             <1>       mov    edx, CODEC_MASTER_VOL_REG ; 02h ; Line Out
0001528B E870FDFFFF             <1>       call   codec_write
                                <1> vt8233_vol_1:
00015290 C3                     <1>       retn
                                <1>
                                <1> ; CODE for SOUND BLASTER 16
                                <1>
                                <1> DetectSB:
                                <1>       ; 24/04/2017
                                <1>       ;pushad
                                <1> ScanPort:
00015291 66BB1002               <1>       mov     bx, 210h    ; start scanning ports
                                <1>                           ; 210h, 220h, .. 260h
                                <1> ResetDSP:
00015295 6689DA                 <1>       mov     dx, bx            ; try to reset the DSP.
00015298 6683C206               <1>       add     dx, 06h
0001529C B001                   <1>       mov    al, 1
0001529E EE                     <1>       out    dx, al
                                <1>
0001529F EC                     <1>       in     al, dx
000152A0 EC                     <1>       in     al, dx
000152A1 EC                     <1>       in     al, dx
000152A2 EC                     <1>       in     al, dx
                                <1>
000152A3 30C0                   <1>       xor     al, al
000152A5 EE                     <1>       out    dx, al
                                <1>
000152A6 6683C208               <1>       add     dx, 08h
000152AA 66B96400               <1>       mov    cx, 100
                                <1> WaitID:
000152AE EC                     <1>       in     al, dx
000152AF 08C0                   <1>       or      al, al
000152B1 7804                   <1>       js      short GetID
000152B3 E2F9                   <1>       loop    WaitID
000152B5 EB0F                   <1>       jmp     short NextPort
                                <1> GetID:
000152B7 6683EA04               <1>       sub     dx, 04h
000152BB EC                     <1>       in     al, dx
000152BC 3CAA                   <1>       cmp     al, 0AAh
000152BE 7413                   <1>       je      short Found
000152C0 6683C204               <1>       add     dx, 04h
000152C4 E2E8                   <1>       loop    WaitID
                                <1> NextPort:
000152C6 6683C310               <1>       add     bx, 10h           ; if not response,
000152CA 6681FB6002             <1>       cmp     bx, 260h    ; try the next port.
000152CF 76C4                   <1>       jbe     short ResetDSP
000152D1 F9                     <1>       stc
000152D2 C3                     <1>       retn
                                <1> Found:
000152D3 66891D[469C0100]       <1>       mov     [audio_io_base], bx      ; SB Port Address Found!
                                <1> ScanIRQ:
                                <1> SetIrqs:
000152DA 28C0                   <1>       sub    al, al ; 0
000152DC A2[3E9C0100]           <1>       mov    [IRQnum], al ; reset
000152E1 A2[439C0100]           <1>       mov    [audio_intr], al ; reset
                                <1>
                                <1>       ; ah > 0 -> set IRQ vector
```

```
1457                                <1>       ; al = IRQ number
1458                                <1>       ;mov   ax, 103h ; IRQ 3
1459                                <1>       ;call  set_hardware_int_vector
1460                                <1>       ;mov   ax, 104h ; IRQ 4
1461                                <1>       ;call  set_hardware_int_vector
1462 000152E6 66B80501             <1>       mov    ax, 105h ; IRQ 5
1463 000152EA E8C8DDFFFF           <1>       call   set_hardware_int_vector
1464 000152EF 66B80701             <1>       mov    ax, 107h ; IRQ 7
1465 000152F3 E8BFDDFFFF           <1>       call   set_hardware_int_vector
1466                                <1>
1467 000152F8 668B15[469C0100]     <1>       mov     dx, [audio_io_base] ; tells to the SB to
1468 000152FF 6683C20C             <1>       add     dx, 0Ch               ; generate a IRQ!
1469                                <1> WaitSb:
1470 00015303 EC                   <1>       in     al, dx
1471 00015304 08C0                 <1>       or      al, al
1472 00015306 78FB                 <1>       js      short WaitSb
1473 00015308 B0F2                 <1>       mov     al, 0F2h
1474 0001530A EE                   <1>       out    dx, al
1475                                <1>
1476 0001530B 31C9                 <1>       xor     ecx, ecx    ; wait until IRQ level
1477                                <1> WaitIRQ:
1478 0001530D A0[3E9C0100]         <1>       mov    al, [IRQnum]
1479 00015312 3C00                 <1>       cmp     al, 0 ; is changed or timeout.
1480 00015314 7706                 <1>       ja     short IrqOk
1481 00015316 6649                 <1>       dec    cx
1482 00015318 75F3                 <1>       jnz    short WaitIRQ
1483 0001531A EB15                 <1>       jmp    short RestoreIrqs
1484                                <1> IrqOk:
1485 0001531C A2[439C0100]         <1>       mov    [audio_intr], al ; set
1486 00015321 668B15[469C0100]     <1>       mov     dx, [audio_io_base]
1487 00015328 6683C20E             <1>       add     dx, 0Eh
1488 0001532C EC                   <1>       in     al, dx ; SB acknowledge.
1489 0001532D B020                 <1>       mov    al, 20h
1490 0001532F E620                 <1>       out    20h, al      ; Hardware acknowledge.
1491                                <1>
1492                                <1> RestoreIrqs:
1493                                <1>       ; ah = 0 -> reset IRQ vector
1494                                <1>       ; al = IRQ number
1495                                <1>       ;mov   ax, 3 ; IRQ 3
1496                                <1>       ;call  set_hardware_int_vector
1497                                <1>       ;mov   ax, 4 ; IRQ 4
1498                                <1>       ;call  set_hardware_int_vector
1499 00015331 66B80500             <1>       mov    ax, 5 ; IRQ 5
1500 00015335 E87DDDFFFF           <1>       call   set_hardware_int_vector
1501 0001533A 66B80700             <1>       mov    ax, 7 ; IRQ 7
1502 0001533E E874DDFFFF           <1>       call   set_hardware_int_vector
1503                                <1>
1504 00015343 31D2                 <1>       xor    edx, edx
1505 00015345 8915[489C0100]       <1>       mov    [audio_dev_id], edx ; 0
1506 0001534B 8915[4C9C0100]       <1>       mov    [audio_vendor], edx ; 0
1507 00015351 8915[509C0100]       <1>       mov    [audio_stats_cmd], edx ; 0
1508                                <1>
1509                                <1>       ;popad
1510                                <1>
1511 00015357 803D[439C0100]01     <1>       cmp     byte [audio_intr], 1 ; IRQ level was changed?
1512                                <1>
1513 0001535E C3                   <1>       retn
1514                                <1>
1515                                <1> %macro      SbOut  1
1516                                <1> %%Wait:
1517                                <1>       in     al, dx
1518                                <1>       or     al, al
1519                                <1>       js     short %%Wait
1520                                <1>       mov    al, %1
1521                                <1>       out    dx, al
1522                                <1> %endmacro
1523                                <1>
1524                                <1> SbInit_play:
1525                                <1>       ; 22/10/2017
1526                                <1>       ; 20/10/2017
1527                                <1>       ; 06/10/2017
1528                                <1>       ; 13/07/2017, 09/08/2017
1529                                <1>       ; 24/04/2017, 15/05/2017, 24/06/2017
1530                                <1>       ;pushad
1531                                <1> SetBuffer:
1532                                <1>       ;mov   byte [DmaFlag], 0
1533                                <1>
1534 0001535F 8B1D[609C0100]       <1>       mov    ebx, [audio_dma_buff] ; physical addr of DMA buff
1535 00015365 89DF                 <1>       mov    edi, ebx
1536 00015367 8B0D[649C0100]       <1>       mov     ecx, [audio_dmabuff_size]
1537                                <1>
1538 0001536D 803D[709C0100]10     <1>       cmp    byte [audio_bps], 16
1539 00015374 7531                 <1>       jne    short sbInit_0 ; set 8 bit DMA buffer
1540                                <1>
1541                                <1>       ; 09/08/2017
1542                                <1>       ; convert byte count to word count
1543 00015376 D1E9                 <1>       shr    ecx, 1
1544 00015378 49                   <1>       dec    ecx ; word count - 1
1545                                <1>       ; convert byte offset to word offset
1546 00015379 D1EB                 <1>       shr    ebx, 1
1547                                <1>
1548                                <1>       ; 16 bit DMA buffer setting (DMA channel 5)
1549 0001537B B005                 <1>       mov     al, 05h  ; set mask bit for channel 5  (4+1)
1550 0001537D E6D4                 <1>       out    0D4h, al
1551                                <1>
1552 0001537F 30C0                 <1>       xor     al, al   ; stops all DMA processes on selected channel
1553 00015381 E6D8                 <1>       out    0D8h, al ; clear selected channel register
1554                                <1>
1555 00015383 88D8                 <1>       mov     al, bl       ; byte 0 of DMA buffer offset in words (physical)
1556 00015385 E6C4                 <1>       out    0C4h, al ; DMA channel 5 port number
1557                                <1>
1558 00015387 88F8                 <1>       mov     al, bh   ; byte 1 of DMA buffer offset in words (physical)
1559 00015389 E6C4                 <1>       out    0C4h, al
1560                                <1>
1561                                <1>       ; 09/08/2017
```

```
1562 0001538B C1EB0F            <1>       shr    ebx, 15       ; complete 16 bit shift
1563 0001538E 80E3FE            <1>       and    bl, 0FEh ; clear bit 0 (not necessary, it will be ignored)
1564                            <1>
1565 00015391 88D8              <1>       mov     al, bl   ; byte 2 of DMA buffer address (physical)
1566 00015393 E68B              <1>       out    8Bh, al  ; page register port addr for channel 5 ; 13/07/2017
1567                            <1>
1568 00015395 88C8              <1>       mov     al, cl   ; low byte of DMA count - 1
1569 00015397 E6C6              <1>       out    0C6h, al ; count register port addr for channel 1
1570                            <1>
1571 00015399 88E8              <1>       mov     al, ch   ; high byte of DMA count - 1
1572 0001539B E6C6              <1>       out    0C6h, al
1573                            <1>
1574                            <1>       ; channel 5, read, autoinitialized, single mode
1575                            <1>       ;mov   al, 49h
1576 0001539D B059              <1>       mov    al, 59h  ; 06/10/2017
1577 0001539F E6D6              <1>       out    0D6h, al ; DMA mode register port address
1578                            <1>
1579 000153A1 B001              <1>       mov     al, 01h  ; clear mask bit for channel 1
1580 000153A3 E6D4              <1>       out    0D4h, al ; DMA mask register port address
1581                            <1>
1582 000153A5 EB28              <1>       jmp    short ClearBuffer
1583                            <1>
1584                            <1> sbInit_0:
1585 000153A7 49                <1>       dec     ecx ; 09/08/2017
1586                            <1>
1587                            <1>       ; 8 bit DMA buffer setting (DMA channel 1)
1588 000153A8 B005              <1>       mov     al, 05h ; set mask bit for channel 1  (4+1)
1589 000153AA E60A              <1>       out    0Ah, al ; DMA mask register
1590                            <1>
1591 000153AC 30C0              <1>       xor     al, al  ; stops all DMA processes on selected channel
1592 000153AE E60C              <1>       out    0Ch, al ; clear selected channel register
1593                            <1>
1594 000153B0 88D8              <1>       mov     al, bl      ; byte 0 of DMA buffer address (physical)
1595 000153B2 E602              <1>       out    02h, al ; DMA channel 1 port number
1596                            <1>
1597 000153B4 88F8              <1>       mov     al, bh  ; byte 1 of DMA buffer address (physical)
1598 000153B6 E602              <1>       out    02h, al
1599                            <1>
1600 000153B8 C1EB10            <1>       shr    ebx, 16
1601                            <1>
1602 000153BB 88D8              <1>       mov     al, bl  ; byte 2 of DMA buffer address (physical)
1603 000153BD E683              <1>       out    83h, al ; page register port addr for channel 1
1604                            <1>
1605 000153BF 88C8              <1>       mov     al, cl  ; low byte of DMA count - 1
1606 000153C1 E603              <1>       out    03h, al ; count register port addr for channel 1
1607                            <1>
1608 000153C3 88E8              <1>       mov     al, ch  ; high byte of DMA count - 1
1609 000153C5 E603              <1>       out    03h, al
1610                            <1>
1611                            <1>       ; channel 1, read, autoinitialized, single mode
1612                            <1>       ;mov   al, 49h
1613 000153C7 B059              <1>       mov    al, 59h ; 06/10/2017
1614 000153C9 E60B              <1>       out    0Bh, al ; DMA mode register port address
1615                            <1>
1616 000153CB B001              <1>       mov     al, 01h ; clear mask bit for channel 1
1617 000153CD E60A              <1>       out    0Ah, al ; DMA mask register port address
1618                            <1>
1619                            <1> ClearBuffer:
1620                            <1>       ;;mov  edi, [audio_dma_buff]
1621                            <1>       ;;mov  ecx, [audio_dmabuff_size]
1622                            <1>       ;inc   ecx
1623                            <1>       ;mov     al, 80h
1624                            <1>       ;;cld
1625                            <1>       ;rep     stosb
1626                            <1> SetIrq:
1627                            <1>       ;mov   ebx, SbIrqhandler
1628                            <1>       ;mov   al, [audio_intr] ; IRQ number
1629                            <1>       ;call  set_dev_IRQ_service
1630                            <1>       ;; SETUP (audio) INTERRUPT CALLBACK SERVICE
1631                            <1>       ;mov   bl, [audio_intr] ; IRQ number
1632                            <1>       ;mov   bh, [audio_cb_mode]
1633                            <1>       ;inc   bh  ; 1 = Signal Response Byte method (fixed value)
1634                            <1>       ;          ; 2 = Callback service method
1635                            <1>       ;          ; 3 = Auto Increment S.R.B. method
1636                            <1>       ;mov   cl, [audio_srb]
1637                            <1>       ;mov   edx, [audio_cb_addr]
1638                            <1>       ;mov   al, [audio_user]
1639                            <1>       ;call  set_irq_callback_service
1640                            <1> ResetDsp:
1641 000153CF 668B15[469C0100]  <1>       mov     dx, [audio_io_base]
1642 000153D6 6683C206          <1>       add     dx, 06h
1643 000153DA B001              <1>       mov     al, 1
1644 000153DC EE                <1>       out    dx, al
1645                            <1>
1646 000153DD EC                <1>       in     al, dx
1647 000153DE EC                <1>       in     al, dx
1648 000153DF EC                <1>       in     al, dx
1649 000153E0 EC                <1>       in     al, dx
1650                            <1>
1651 000153E1 30C0              <1>       xor     al, al
1652 000153E3 EE                <1>       out    dx, al
1653                            <1>
1654 000153E4 66B96400          <1>       mov     cx, 100
1655 000153E8 28E4              <1>       sub    ah, ah ; 0
1656                            <1> WaitId:
1657 000153EA 668B15[469C0100]  <1>       mov     dx, [audio_io_base]
1658 000153F1 6683C20E          <1>       add     dx, 0Eh
1659 000153F5 EC                <1>       in     al, dx
1660 000153F6 08C0              <1>       or      al, al
1661 000153F8 7807              <1>       js      short sb_GetId
1662 000153FA E2EE              <1>       loop    WaitId
1663 000153FC E9B4000000        <1>       jmp     sb_Exit
1664                            <1> sb_GetId:
1665 00015401 668B15[469C0100]  <1>       mov     dx, [audio_io_base]
1666 00015408 6683C20A          <1>       add     dx, 0Ah
```

```
1667 0001540C EC                  <1>       in     al, dx
1668 0001540D 3CAA                <1>       cmp     al, 0AAh
1669 0001540F 7407                <1>       je      short SbOk
1670 00015411 E2D7                <1>       loop    WaitId
1671 00015413 E99D000000          <1>       jmp    sb_Exit
1672                              <1> SbOk:
1673 00015418 668B15[469C0100]    <1>       mov     dx, [audio_io_base]
1674 0001541F 6683C20C            <1>       add     dx, 0Ch
1675                              <1>       SbOut   0D1h ; Turn on speaker
1675                              <2> %%Wait:
1675 00015423 EC                  <2>  in al, dx
1675 00015424 08C0                <2>  or al, al
1675 00015426 78FB                <2>  js short %%Wait
1675 00015428 B0D1                <2>  mov al, %1
1675 0001542A EE                  <2>  out dx, al
1676                              <1>       SbOut   41h ; 8 bit or 16 bit transfer
1676                              <2> %%Wait:
1676 0001542B EC                  <2>  in al, dx
1676 0001542C 08C0                <2>  or al, al
1676 0001542E 78FB                <2>  js short %%Wait
1676 00015430 B041                <2>  mov al, %1
1676 00015432 EE                  <2>  out dx, al
1677 00015433 668B1D[729C0100]    <1>       mov    bx, [audio_freq] ; sampling rate (Hz)
1678                              <1>       SbOut  bh ; sampling rate high byte
1678                              <2> %%Wait:
1678 0001543A EC                  <2>  in al, dx
1678 0001543B 08C0                <2>  or al, al
1678 0001543D 78FB                <2>  js short %%Wait
1678 0001543F 88F8                <2>  mov al, %1
1678 00015441 EE                  <2>  out dx, al
1679                              <1>       SbOut  bl ; sampling rate low byte
1679                              <2> %%Wait:
1679 00015442 EC                  <2>  in al, dx
1679 00015443 08C0                <2>  or al, al
1679 00015445 78FB                <2>  js short %%Wait
1679 00015447 88D8                <2>  mov al, %1
1679 00015449 EE                  <2>  out dx, al
1680                              <1>
1681                              <1>       ; 22/05/2017
1682 0001544A E8C0000000          <1>       call   sb16_volume_initial ; 15/05/2017
1683                              <1>       ; 20/05/2017
1684                              <1>       ;call  sb16_volume
1685                              <1>
1686                              <1> StartDma:
1687                              <1>       ; autoinitialized mode
1688 0001544F 803D[709C0100]10    <1>       cmp    byte [audio_bps], 16 ; 16 bit samples
1689 00015456 7411                <1>       je     short sb_play_1
1690                              <1>       ; 8 bit samples
1691 00015458 66BBC600            <1>       mov    bx, 0C6h ; 8 bit output (0C6h)
1692 0001545C 803D[719C0100]02    <1>       cmp    byte [audio_stmo], 2 ; 1 = mono, 2 = stereo
1693 00015463 7214                <1>       jb     short sb_play_2
1694 00015465 B720                <1>       mov    bh, 20h      ; 8 bit stereo (20h)
1695 00015467 EB10                <1>       jmp    short sb_play_2
1696                              <1> sb_play_1:
1697                              <1>       ; 16 bit samples
1698 00015469 66BBB610            <1>       mov    bx, 10B6h ; 16 bit output (0B6h)
1699 0001546D 803D[719C0100]02    <1>       cmp    byte [audio_stmo], 2 ; 1 = mono, 2 = stereo
1700 00015474 7203                <1>       jb     short sb_play_2
1701 00015476 80C720              <1>       add    bh, 20h      ; 16 bit stereo (30h)
1702                              <1> sb_play_2:
1703                              <1>       ; PCM output (8/16 bit mono autoinitialized transfer)
1704                              <1>       SbOut   bl ; bCommand
1704                              <2> %%Wait:
1704 00015479 EC                  <2>  in al, dx
1704 0001547A 08C0                <2>  or al, al
1704 0001547C 78FB                <2>  js short %%Wait
1704 0001547E 88D8                <2>  mov al, %1
1704 00015480 EE                  <2>  out dx, al
1705                              <1>       SbOut  bh ; bMode
1705                              <2> %%Wait:
1705 00015481 EC                  <2>  in al, dx
1705 00015482 08C0                <2>  or al, al
1705 00015484 78FB                <2>  js short %%Wait
1705 00015486 88F8                <2>  mov al, %1
1705 00015488 EE                  <2>  out dx, al
1706 00015489 8B1D[649C0100]      <1>       mov    ebx, [audio_dmabuff_size]  ; 15/05/2017
1707 0001548F D1EB                <1>       shr    ebx, 1 ; half buffer size
1708                              <1>       ; 20/10/2017
1709 00015491 803D[709C0100]10    <1>       cmp    byte [audio_bps], 16 ; 16 bit DMA
1710 00015498 7502                <1>       jne    short sb_play_3
1711 0001549A D1EB                <1>       shr    ebx, 1 ; byte count to word count
1712                              <1> sb_play_3:
1713 0001549C 664B                <1>       dec    bx  ; wBlkSize is one less than the actual size
1714                              <1>       SbOut   bl
1714                              <2> %%Wait:
1714 0001549E EC                  <2>  in al, dx
1714 0001549F 08C0                <2>  or al, al
1714 000154A1 78FB                <2>  js short %%Wait
1714 000154A3 88D8                <2>  mov al, %1
1714 000154A5 EE                  <2>  out dx, al
1715                              <1>       SbOut   bh
1715                              <2> %%Wait:
1715 000154A6 EC                  <2>  in al, dx
1715 000154A7 08C0                <2>  or al, al
1715 000154A9 78FB                <2>  js short %%Wait
1715 000154AB 88F8                <2>  mov al, %1
1715 000154AD EE                  <2>  out dx, al
1716                              <1>
1717 000154AE C605[749C0100]01    <1>       mov    byte [audio_play_cmd], 1 ; playing !
1718                              <1>
1719                              <1>       ;; Set Voice and master volumes
1720                              <1>       ;mov   dx, [audio_io_base]
1721                              <1>       ;add   dl, 4 ; Mixer chip Register Address Port
1722                              <1>       ;SbOut 30h   ; select Master Volume Register (L)
1723                              <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
000154B5 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>
000154B6 668B15[469C0100]       <1>       mov     dx, [audio_io_base]
                                <1>       ; 20/10/2017
000154BD 80C20F                 <1>       add     dl, 0Fh ; 2xFh (DSP 16 bit intr ack)
000154C0 803D[709C0100]10       <1>       cmp    byte [audio_bps], 16
000154C7 7402                   <1>       je     short sb_irq_16bit_ack
                                <1> sb_irq_8bit_ack:
000154C9 FECA                   <1>       dec    dl  ; 2xEh (DSP 8 bit intr ack)
                                <1> sb_irq_16bit_ack:
000154CB 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>
000154CC 803D[749C0100]01       <1>       cmp    byte [audio_play_cmd], 1
000154D3 7307                   <1>       jnb    short sb_irq_h1
                                <1> sb_irq_h0:
000154D5 E8A9000000             <1>       call   sb16_stop
000154DA EB2B                   <1>       jmp    short sb_irq_h3
                                <1> sb_irq_h1:
                                <1>       ;call  sb16_tuneloop
                                <1>       ; 09/10/2017
                                <1> sb16_tuneloop:
000154DC 8B3D[609C0100]         <1>       mov    edi, [audio_dma_buff]
000154E2 8B0D[649C0100]         <1>       mov    ecx, [audio_dmabuff_size]
000154E8 D1E9                   <1>       shr    ecx, 1 ; dma buff size / 2 = half buffer size
                                <1>
                                <1>       ; 22/05/2017
000154EA F605[689C0100]01       <1>       test   byte [audio_flag], 1  ; Current flag value
000154F1 7402                   <1>       jz     short sb_tlp1 ; EOL (Half Buffer 1 must be filled)
                                <1>       ; FLAG (Half Buffer 2 must be filled)
000154F3 01CF                   <1>       add    edi, ecx
                                <1>       ; 15/05/2017
                                <1> sb_tlp1:
000154F5 8B35[589C0100]         <1>       mov    esi, [audio_p_buffer] ; phy addr of audio buff
                                <1>       ;rep   movsb
000154FB C1E902                 <1>       shr    ecx, 2 ; half buff size / 4
000154FE F3A5                   <1>       rep    movsd
                                <1>       ;retn
                                <1>
                                <1>       ; 10/10/2017
                                <1>       ; switch flag value
00015500 8035[689C0100]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
```

```
1829                                <1>
1830                                <1>       ;pop   edi
1831                                <1>       ;pop   esi
1832                                <1>       ;pop   edx
1833                                <1>       ;pop   ecx
1834                                <1>       ;pop   ebx ; * must be restored !
1835                                <1>       ;pop   eax ; * must be restored !
1836                                <1>
1837 00015507 C3                   <1>       retn
1838                                <1>
1839                                <1> sb16_volume:
1840                                <1>       ; 22/10/2017
1841                                <1>       ; mov [audio_master_volume_l], cl
1842                                <1>       ; mov [audio_master_volume_h], ch
1843 00015508 66890D[769C0100]     <1>       mov    [audio_master_volume], cx
1844                                <1> sb16_volume_initial:
1845 0001550F 6652                 <1>       push   dx ; DX (port address) must be saved
1846 00015511 668B15[469C0100]     <1>       mov    dx, [audio_io_base]
1847 00015518 6683C204             <1>       add    dx, 4 ; Mixer chip address port
1848 0001551C B022                 <1>       mov    al, 22h ; master volume
1849 0001551E EE                   <1>       out    dx, al
1850 0001551F 6642                 <1>       inc    dx
1851 00015521 8A25[769C0100]       <1>       mov    ah, [audio_master_volume_l]
1852 00015527 C0EC02               <1>       shr    ah, 2 ; 32 -> 8 level
1853 0001552A C0E405               <1>       shl    ah, 5 ; bit 5 to 7
1854 0001552D A0[779C0100]         <1>       mov    al, [audio_master_volume_r]
1855 00015532 C0E802               <1>       shr    al, 2 ; 32 -> 8 level
1856                                <1>       ;and   al, 0Fh
1857 00015535 D0E0                 <1>       shl    al, 1 ; bit 1 to 3
1858 00015537 08E0                 <1>       or     al, ah
1859 00015539 EE                   <1>       out    dx, al
1860 0001553A 665A                 <1>       pop    dx ; DX (port address) must be restored
1861 0001553C C3                   <1>       retn
1862                                <1>
1863                                <1> sb16_pause:
1864 0001553D 668B15[469C0100]     <1>       mov    dx, [audio_io_base]
1865 00015544 6683C20C             <1>       add    dx, 0Ch ; Command & Data Port
1866 00015548 803D[709C0100]10     <1>       cmp    byte [audio_bps], 16 ; 16 bit samples
1867 0001554F 7404                 <1>       je     short sb_pause_1
1868                                <1>       ; 8 bit samples
1869 00015551 B3D0                 <1>       mov    bl, 0D0h ; 8 bit DMA mode
1870 00015553 EB02                 <1>       jmp    short sb_pause_2
1871                                <1> sb_pause_1:
1872                                <1>       ; 16 bit samples
1873 00015555 B3D5                 <1>       mov    bl, 0D5h ; 16 bit DMA mode
1874                                <1> sb_pause_2:
1875                                <1>       SbOut   bl ; bCommand
1875                                <2> %%Wait:
1875 00015557 EC                   <2>  in al, dx
1875 00015558 08C0                 <2>  or al, al
1875 0001555A 78FB                 <2>  js short %%Wait
1875 0001555C 88D8                 <2>  mov al, %1
1875 0001555E EE                   <2>  out dx, al
1876                                <1> sb_pause_3:
1877 0001555F C3                   <1>       retn
1878                                <1>
1879                                <1> sb16_continue:
1880 00015560 668B15[469C0100]     <1>       mov    dx, [audio_io_base]
1881 00015567 6683C20C             <1>       add    dx, 0Ch ; Command & Data Port
1882 0001556B 803D[709C0100]10     <1>       cmp    byte [audio_bps], 16 ; 16 bit samples
1883 00015572 7404                 <1>       je     short sb_cont_1
1884                                <1>       ; 8 bit samples
1885 00015574 B3D4                 <1>       mov    bl, 0D4h ; 8 bit DMA mode
1886 00015576 EB02                 <1>       jmp    short sb_cont_2
1887                                <1> sb_cont_1:
1888                                <1>       ; 16 bit samples
1889 00015578 B3D6                 <1>       mov    bl, 0D6h ; 16 bit DMA mode
1890                                <1> sb_cont_2:
1891                                <1>       SbOut   bl ; bCommand
1891                                <2> %%Wait:
1891 0001557A EC                   <2>  in al, dx
1891 0001557B 08C0                 <2>  or al, al
1891 0001557D 78FB                 <2>  js short %%Wait
1891 0001557F 88D8                 <2>  mov al, %1
1891 00015581 EE                   <2>  out dx, al
1892                                <1> sb_cont_3:
1893 00015582 C3                   <1>       retn
1894                                <1>
1895                                <1> sb16_stop:
1896                                <1>       ; 24/04/2017
1897 00015583 803D[749C0100]00     <1>       cmp    byte [audio_play_cmd], 0
1898 0001558A 7648                 <1>       jna    short sb16_stop_4
1899                                <1>
1900                                <1>       ; 22/05/2017
1901 0001558C 668B15[469C0100]     <1>       mov    dx, [audio_io_base]
1902 00015593 6683C20C             <1>       add    dx, 0Ch
1903                                <1>
1904 00015597 B3D9                 <1>       mov    bl, 0D9h ; exit auto-initialize 16 bit transfer
1905                                <1>       ; stop  autoinitialized DMA transfer mode
1906 00015599 803D[709C0100]10     <1>       cmp    byte [audio_bps], 16 ; 16 bit samples
1907 000155A0 7402                 <1>       je     short sb16_stop_1
1908                                <1>       ;mov   bl, 0DAh ; exit auto-initialize 8 bit transfer
1909 000155A2 FEC3                 <1>       inc    bl
1910                                <1> sb16_stop_1:
1911                                <1>       SbOut  bl ; exit auto-initialize transfer command
1911                                <2> %%Wait:
1911 000155A4 EC                   <2>  in al, dx
1911 000155A5 08C0                 <2>  or al, al
1911 000155A7 78FB                 <2>  js short %%Wait
1911 000155A9 88D8                 <2>  mov al, %1
1911 000155AB EE                   <2>  out dx, al
1912                                <1>
1913 000155AC 30C0                 <1>       xor     al, al ; stops all DMA processes on selected channel
1914                                <1>
1915 000155AE 803D[709C0100]10     <1>       cmp    byte [audio_bps], 16 ; 16 bit samples
```

```
1916 000155B5 7404                <1>       je     short sb16_stop_2
1917 000155B7 E60C                <1>       out    0Ch, al ; clear selected channel register
1918 000155B9 EB02                <1>       jmp    short sb16_stop_3
1919                              <1>
1920                              <1> sb16_stop_2:
1921 000155BB E6D8                <1>       out    0D8h, al ; clear selected channel register
1922                              <1>
1923                              <1> sb16_stop_3:
1924 000155BD C605[749C0100]00    <1>       mov    byte [audio_play_cmd], 0 ; stop !
1925                              <1> SbDone:
1926                              <1>       ;mov   dx, [audio_io_base]
1927                              <1>       ;add   dx, 0Ch
1928                              <1>       SbOut   0D0h
1928                              <2> %%Wait:
1928 000155C4 EC                  <2>  in al, dx
1928 000155C5 08C0                <2>  or al, al
1928 000155C7 78FB                <2>  js short %%Wait
1928 000155C9 B0D0                <2>  mov al, %1
1928 000155CB EE                  <2>  out dx, al
1929                              <1>       SbOut   0D3h
1929                              <2> %%Wait:
1929 000155CC EC                  <2>  in al, dx
1929 000155CD 08C0                <2>  or al, al
1929 000155CF 78FB                <2>  js short %%Wait
1929 000155D1 B0D3                <2>  mov al, %1
1929 000155D3 EE                  <2>  out dx, al
1930                              <1> sb16_stop_4:
1931 000155D4 C3                  <1>       retn
1932                              <1>
1933                              <1> sb16_reset:
1934                              <1>       ; 24/04/2017
1935 000155D5 668B15[469C0100]    <1>       mov     dx, [audio_io_base] ; try to reset the DSP.
1936 000155DC 6683C206            <1>       add     dx, 06h
1937 000155E0 B001                <1>       mov    al, 1
1938 000155E2 EE                  <1>       out    dx, al
1939                              <1>
1940 000155E3 EC                  <1>       in     al, dx
1941 000155E4 EC                  <1>       in     al, dx
1942 000155E5 EC                  <1>       in     al, dx
1943 000155E6 EC                  <1>       in     al, dx
1944                              <1>
1945 000155E7 30C0                <1>       xor     al, al
1946 000155E9 EE                  <1>       out    dx, al
1947                              <1>
1948 000155EA 6683C208            <1>       add     dx, 08h
1949 000155EE 66B96400            <1>       mov    cx, 100
1950                              <1> sbrstWaitID:
1951 000155F2 EC                  <1>       in     al, dx
1952 000155F3 08C0                <1>       or      al, al
1953 000155F5 7804                <1>       js      short sbrstGetID
1954 000155F7 E2F9                <1>       loop    sbrstWaitID
1955 000155F9 F9                  <1>       stc
1956 000155FA C3                  <1>       retn
1957                              <1> sbrstGetID:
1958 000155FB 6683EA04            <1>       sub     dx, 04h
1959 000155FF EC                  <1>       in     al, dx
1960 00015600 3CAA                <1>       cmp     al, 0AAh
1961 00015602 7406                <1>       je      short sb_rst_retn
1962 00015604 6683C204            <1>       add     dx, 04h
1963 00015608 E2E8                <1>       loop    sbrstWaitID
1964                              <1> sb_rst_retn:
1965 0001560A C3                  <1>       retn
1966                              <1>
1967                              <1> ac97_codec_config:
1968                              <1>       ; 10/06/2017
1969                              <1>       ; 05/06/2017
1970                              <1>       ; 29/05/2017
1971                              <1>       ; 28/05/2017 (TRDOS 386, 'audio.s')
1972                              <1>       ; 07/11/2016 (Erdogan Tan)
1973                              <1>       ; Derived from 'codecConfig' procedure in 'CODEC.ASM'
1974                              <1>       ; .wav player for DOS by Jeff Leyda (02/09/2002)
1975                              <1>
1976                              <1>       ;; 'PLAYER.ASM'
1977                              <1>       ;; get ICH base address regs for mixer and bus master
1978                              <1>
1979                              <1> init_ac97_controller: ; 10/06/2017
1980 0001560B A1[489C0100]        <1>       mov    eax, [audio_dev_id]
1981                              <1>         ;mov al, NAMBAR_REG
1982                              <1>         ;;call  pciRegRead16                ; read PCI registers 10-11
1983                              <1>         ;call      pciRegRead32
1984                              <1>       ;and   dx, IO_ADDR_MASK          ; mask off BIT0
1985                              <1>       ;;and  edx, IO_ADDR_MASK
1986                              <1>
1987                              <1>         ;mov [NAMBAR], dx              ; save audio mixer base addr
1988                              <1>
1989                              <1>         ;mov    al, NABMBAR_REG
1990                              <1>         ;;call     pciRegRead16
1991                              <1>         ;call      pciRegRead32
1992                              <1>       ;and   dx, 0FFC0h ; IO_ADDR_MASK
1993                              <1>       ;;and  edx, 0FFC0h
1994                              <1>
1995                              <1>         ;mov    [NABMBAR], dx               ; save bus master base addr
1996                              <1>
1997                              <1>       ;mov   eax, [audio_dev_id]
1998 00015610 B004                <1>         mov     al, PCI_CMD_REG
1999                              <1>         ;call      pciRegRead8              ; read PCI command register
2000 00015612 E84AF8FFFF          <1>         call pciRegRead16
2001 00015617 80CA05              <1>       or      dl, IO_ENA+BM_ENA              ; enable IO and bus master
2002                              <1>         ;call      pciRegWrite8
2003 0001561A E8ADF8FFFF          <1>       call   pciRegWrite16
2004                              <1>
2005                              <1>       ; 'CODEC.ASM'
2006                              <1>
2007                              <1>       ; enable codec, unmute stuff, set output rate
2008                              <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>
0001561F E847F9FFFF             <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>         ;
00015624 66BA2C00               <1>         mov     dx, GLOB_CNT_REG ; 2Ch
00015628 660315[7A9C0100]       <1>         add     dx, [NABMBAR]
0001562F ED                     <1>         in      eax, dx
                                <1>         ; ?
00015630 66BA3000               <1>         mov     dx, GLOB_STS_REG ; 30h
00015634 660315[7A9C0100]       <1>         add     dx, [NABMBAR]
0001563B ED                     <1>         in      eax, dx
                                <1>
0001563C 83F8FF                 <1>         cmp     eax, 0FFFFFFFFh ; -1
0001563F 744B                   <1>         je      short init_ac97_codec_err1
                                <1>
00015641 A900030010             <1>         test    eax, CTRL_ST_CREADY
00015646 7507                   <1>         jnz     short _ac97_codec_ready
                                <1>
00015648 E8EF020000             <1>         call    reset_ac97_codec
0001564D 723E                   <1>         jc      short init_ac97_codec_err2
                                <1>
                                <1> _ac97_codec_ready:
0001564F 668B15[789C0100]       <1>         mov     dx, [NAMBAR]
                                <1>         ;add    dx, 0 ; ac_reg_0 ; reset register
00015656 66EF                   <1>         out     dx, ax
                                <1>
00015658 31C0                   <1>         xor     eax, eax ; 0
0001565A 668B15[789C0100]       <1>         mov     dx, [NAMBAR]
00015661 6683C226               <1>         add     dx, CODEC_REG_POWERDOWN
00015665 66EF                   <1>         out     dx, ax
                                <1>
                                <1>         ; 10/06/2017
                                <1>         ; 29/05/2017
                                <1>         ; wait for 1 second
00015667 B9E8030000             <1>         mov     ecx, 1000 ; 1000*0.25ms = 1s
                                <1> _ac97_codec_rloop:
0001566C E807F9FFFF             <1>         call    delay1_4ms
00015671 E802F9FFFF             <1>         call    delay1_4ms
00015676 E8FDF8FFFF             <1>         call    delay1_4ms
0001567B E8F8F8FFFF             <1>         call    delay1_4ms
                                <1>         ;mov    dx, [NAMBAR]
                                <1>         ;add    dx, CODEC_REG_POWERDOWN
00015680 66ED                   <1>         in      ax, dx
00015682 6683E00F               <1>         and     ax, 0Fh
00015686 3C0F                   <1>         cmp     al, 0Fh
00015688 7404                   <1>         je      short _ac97_codec_init_ok
0001568A E2E0                   <1>         loop    _ac97_codec_rloop
                                <1>
                                <1> init_ac97_codec_err1:
0001568C F9                     <1>         stc
                                <1> init_ac97_codec_err2:
0001568D C3                     <1>         retn
                                <1>
                                <1> _ac97_codec_init_ok:
0001568E B002                   <1>           mov     al, 2 ; force set 16-bit 2-channel PCM
00015690 66BA2C00               <1>         mov     dx, GLOB_CNT_REG ; 2Ch
00015694 660315[7A9C0100]       <1>         add     dx, [NABMBAR]
0001569B EF                     <1>         out     dx, eax
                                <1>
                                <1>         ;call   delay1_4ms
                                <1>
                                <1>         ; 10/06/2017
0001569C 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>       ; 22/07/2020
                            <1>       ; 10/06/2017
                            <1>       ; 29/05/2017
000156A1 B802020000         <1>       mov     eax, 0202h
000156A6 66A3[769C0100]     <1>       mov    [audio_master_volume], ax
000156AC 66B81F1F           <1>       mov    ax, 1F1Fh ; 31, 31
                            <1>
000156B0 668B15[789C0100]   <1>       mov     dx, [NAMBAR]
000156B7 6683C202           <1>       add     dx, CODEC_MASTER_VOL_REG ;02h
000156BB 6631C0             <1>       xor    ax, ax       ; volume attenuation = 0 (max. volume)
000156BE 66EF               <1>       out     dx, ax
                            <1>
000156C0 668B15[789C0100]   <1>       mov     dx, [NAMBAR]
000156C7 6683C206           <1>       add     dx, CODEC_MASTER_MONO_VOL_REG   ;06h
                            <1>       ;xor   ax, ax
000156CB 66EF               <1>       out     dx, ax
                            <1>
000156CD 668B15[789C0100]   <1>       mov     dx, [NAMBAR]
000156D4 6683C20A           <1>       add     dx, CODEC_PCBEEP_VOL_REG ;0Ah
                            <1>       ;xor    ax, ax
000156D8 66EF               <1>       out     dx, ax
                            <1>
000156DA 668B15[789C0100]   <1>       mov     dx, [NAMBAR]
000156E1 6683C218           <1>       add     dx, CODEC_PCM_OUT_REG          ;18h
                            <1>       ;xor    ax, ax
000156E5 66EF               <1>       out     dx, ax
                            <1>
000156E7 66B80880           <1>       mov     ax, 8008h ; Mute
000156EB 668B15[789C0100]   <1>       mov     dx, [NAMBAR]
                            <1>       ; 22/07/2020
000156F2 6683C20C           <1>       add    dx, CODEC_PHONE_VOL_REG          ;0Ch
                            <1>                           ; AC97_PHONE_VOL ; TAD Input (Mono)
000156F6 66EF               <1>       out     dx, ax
                            <1>
000156F8 66B80808           <1>         mov  ax, 0808h
000156FC 668B15[789C0100]   <1>       mov     dx, [NAMBAR]
00015703 6683C210           <1>         add  dx, CODEC_LINE_IN_VOL_REG ;10h ; Line Input (Stereo)
00015707 66EF               <1>       out     dx, ax
                            <1>
                            <1>       ;mov   ax, 0808h
00015709 668B15[789C0100]   <1>       mov     dx, [NAMBAR]
00015710 6683C212           <1>         add  dx, CODEC_CD_VOL_REG ;12h ; CR Input (Stereo)
00015714 66EF               <1>       out     dx, ax
                            <1>
                            <1>       ;mov   ax, 0808h
00015716 668B15[789C0100]   <1>       mov     dx, [NAMBAR]
0001571D 6683C216           <1>         add  dx, CODEC_AUX_VOL_REG ;16h ; Aux Input (Stereo)
00015721 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:
00015723 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>
00015724 D1E9               <1>       shr    ecx, 1 ; dma half buffer size
00015726 89CE               <1>       mov    esi, ecx
                            <1>
00015728 BF[7C9C0100]       <1>         mov     edi, audio_bdl_buff    ; get BDL address
0001572D B910000000         <1>         mov     ecx, 32 / 2            ; make 32 entries in BDL
                            <1>
00015732 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>
00015734 A1[609C0100]       <1>       mov    eax, [audio_dma_buff]     ; Physical address of DMA buffer
                            <1>
                            <1> s_ac97_bdl1:
00015739 AB                 <1>       stosd                      ; store dmabuffer1 address
                            <1>
0001573A 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>
0001573C 89F0                   <1>       mov    eax, esi ; DMA half buffer size
0001573E 01C2                   <1>       add    edx, eax
00015740 D1E8                   <1>       shr    eax, 1 ; count of 16 bit samples
                                <1>       ;or    eax, IOC+BUP
00015742 0D00000080             <1>       or     eax, IOC ; 11/06/2017
00015747 AB                     <1>       stosd
                                <1>
                                <1> ; 2nd buffer:
                                <1>
00015748 89D0                   <1>         mov  eax, edx ; Physical address of the 2nd half of DMA buffer
0001574A 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> ;
0001574B 89F0                   <1>       mov    eax, esi ; DMA half buffer size
0001574D D1E8                   <1>       shr    eax, 1 ; count of 16 bit samples
                                <1>       ;or    eax, IOC+BUP
0001574F 0D00000080             <1>       or     eax, IOC ; 11/06/2017
00015754 AB                     <1>       stosd
                                <1>
00015755 E2DD                   <1>       loop    s_ac97_bdl0
                                <1>
00015757 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>
00015758 668B15[789C0100]       <1>       mov          dx, [NAMBAR]
0001575F 6683C22A               <1>       add          dx, CODEC_EXT_AUDIO_CTRL_REG     ; 2Ah
00015763 66ED                   <1>       in           ax, dx
00015765 6683C801               <1>       or     ax, 1
00015769 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>
0001576B 66A1[729C0100]         <1>       mov    ax, [audio_freq]          ; sample rate
                                <1>
00015771 668B15[789C0100]       <1>       mov          dx, [NAMBAR]
00015778 6683C22C               <1>       add          dx, CODEC_PCM_FRONT_DACRATE_REG  ; 2Ch
0001577C 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> ;
0001577E 668B15[7A9C0100]       <1>        mov     dx, [NABMBAR]
00015785 6683C21B               <1>        add     dx, PO_CR_REG                   ; set pointer to Cntl reg
00015789 B002                   <1>        mov     al, RR                          ; set reset
0001578B 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> ;
0001578C B8[7C9C0100]           <1>         mov  eax, audio_bdl_buff
00015791 668B15[7A9C0100]       <1>       mov    dx, [NABMBAR]
```

```
2324 00015798 6683C210            <1>       add    dx, PO_BDBAR_REG
2325 0001579C EF                  <1>       out    dx, eax
2326                              <1> ;
2327                              <1> ; All set. Let's play some music.
2328                              <1> ;
2329                              <1> ;
2330 0001579D B81F000000          <1>       mov    eax, 31
2331 000157A2 E816000000          <1>       call    set_ac97_LastValidIndex
2332                              <1>
2333 000157A7 C605[749C0100]01    <1>       mov    byte [audio_play_cmd], 1 ; play command (do not stop) !
2334                              <1>
2335                              <1> ac97_play: ; continue to play (after pause)
2336                              <1>       ; 11/06/2017
2337                              <1>       ; 29/05/2017
2338                              <1>       ; 28/05/2017
2339 000157AE 668B15[7A9C0100]    <1>         mov     dx, [NABMBAR]
2340 000157B5 6683C21B            <1>         add     dx, PO_CR_REG          ; PCM out control register
2341 000157B9 B011                <1>         mov  al, IOCE+RPBM ; 29/05/2017
2342                              <1>         ;mov al, 1Dh ; (Ref: KolibriOS, intelac97.asm, 'play:')
2343 000157BB EE                  <1>       out     dx, al                   ; set start!
2344                              <1>
2345                              <1>       ;mov   byte [audio_play_cmd], 1 ; play command (do not stop) !
2346                              <1>
2347 000157BC C3                  <1>       retn
2348                              <1>
2349                              <1> ;input AL = index # to stop on
2350                              <1> set_ac97_LastValidIndex:
2351                              <1>       ; 28/05/2017
2352                              <1>       ; Derived from 'setLastValidIndex' procedure in 'ICHWAV.ASM'
2353                              <1>       ; .wav player for DOS by Jeff Leyda (02/09/2002)
2354 000157BD 668B15[7A9C0100]    <1>       mov    dx, [NABMBAR]
2355 000157C4 6683C215            <1>       add    dx, PO_LVI_REG
2356 000157C8 EE                  <1>         out     dx, al
2357                              <1>       ;mov   [audio_lvi], al ; for ac97_int_handler
2358 000157C9 C3                  <1>       retn
2359                              <1>
2360                              <1> ac97_volume:
2361                              <1>       ; 28/05/2017
2362                              <1>       ; bl = component (0 = master/playback/lineout volume)
2363                              <1>       ; cl = left channel volume level (0 to 31)
2364                              <1>       ; ch = right channel volume level (0 to 31)
2365                              <1>
2366 000157CA 08DB                <1>       or     bl, bl
2367 000157CC 7523                <1>       jnz    short ac97_vol_1 ; temporary !
2368 000157CE 66B81F1F            <1>       mov    ax, 1F1Fh ; 31,31
2369 000157D2 38C1                <1>       cmp    cl, al
2370 000157D4 771B                <1>       ja     short ac97_vol_1 ; temporary !
2371 000157D6 38E5                <1>       cmp    ch, ah
2372 000157D8 7717                <1>       ja     short ac97_vol_1 ; temporary !
2373 000157DA 66890D[769C0100]    <1>       mov    [audio_master_volume], cx
2374 000157E1 6629C8              <1>       sub    ax, cx
2375 000157E4 668B15[789C0100]    <1>       mov     dx, [NAMBAR]
2376 000157EB 6683C202            <1>       add     dx, CODEC_MASTER_VOL_REG  ; 02h ; Line Out
2377 000157EF 66EF                <1>       out     dx, ax
2378                              <1> ac97_vol_1:
2379 000157F1 C3                  <1>       retn
2380                              <1>
2381                              <1> ac97_int_handler:
2382                              <1>       ; 12/10/2017
2383                              <1>       ; 10/10/2017
2384                              <1>       ; 09/10/2017
2385                              <1>       ; 13/06/2017, 13/06/2017
2386                              <1>       ; 10/06/2017, 11/06/2017
2387                              <1>       ; Interrupt Handler for AC97 (ICH) Audio Controller
2388                              <1>       ; Note: called by 'dev_IRQ_service'
2389                              <1>       ; 28/05/2017
2390                              <1>
2391                              <1>       ;push  eax ; * must be saved !
2392                              <1>       ;push  edx
2393                              <1>       ;push  ecx
2394                              <1>       ;push  ebx ; * must be saved !
2395                              <1>       ;push  esi
2396                              <1>       ;push  edi
2397                              <1>
2398                              <1>       ;cmp   byte [audio_busy], 1
2399                              <1>       ;jnb   _ac97_ih2    ; busy !
2400                              <1>
2401 000157F2 66BA3000            <1>         mov     dx, GLOB_STS_REG
2402 000157F6 660315[7A9C0100]    <1>         add  dx, [NABMBAR]
2403 000157FD ED                  <1>       in     eax, dx
2404                              <1>
2405 000157FE 83F8FF              <1>         cmp     eax, 0FFFFFFFFh ; -1
2406 00015801 0F849A000000        <1>         je      _ac97_ih3 ; exit
2407                              <1>
2408 00015807 A940000000          <1>         test    eax, 40h ; PCM Out Interrupt
2409 0001580C 750E                <1>         jnz     short _ac97_ih0
2410                              <1>
2411 0001580E 85C0                <1>       test   eax, eax
2412 00015810 0F848B000000        <1>       jz     _ac97_ih3 ; exit
2413                              <1>
2414                              <1>       ;mov   dx, GLOB_STS_REG
2415                              <1>         ;add dx, [NABMBAR]
2416 00015816 EF                  <1>       out    dx, eax
2417                              <1>
2418 00015817 E985000000          <1>       jmp    _ac97_ih3 ; exit
2419                              <1>
2420                              <1> _ac97_ih0:
2421 0001581C 50                  <1>       push   eax
2422                              <1>       ; 09/10/2017
2423 0001581D 803D[749C0100]01    <1>       cmp    byte [audio_play_cmd], 1
2424 00015824 727C                <1>       jb     short _ac97_ih4 ; stop command !
2425                              <1>
2426                              <1>       ;mov   byte [audio_busy], 1
2427                              <1>
2428                              <1>       ;mov   al, 10h
```

```
2429                                    <1>       ;mov   dx, PO_CR_REG
2430                                    <1>         ;add dx, [NABMBAR]
2431                                    <1>       ;out   dx, al
2432                                    <1>
2433 00015826 66B81C00                 <1>       mov    ax, 1Ch ; FIFOE(=16)+BCIS(=8)+LVBCI(=4)
2434 0001582A 66BA1600                 <1>       mov    dx, PO_SR_REG
2435 0001582E 660315[7A9C0100]         <1>         add  dx, [NABMBAR]
2436 00015835 66EF                     <1>       out    dx, ax
2437                                    <1>
2438 00015837 66BA1400                 <1>       mov    dx, PO_CIV_REG
2439 0001583B 660315[7A9C0100]         <1>         add  dx, [NABMBAR]
2440 00015842 EC                       <1>       in     al, dx
2441                                    <1>
2442                                    <1>       ;cmp   al, [audio_civ] ; [audio_flag]
2443                                    <1>       ;je    short _ac97_ih2
2444                                    <1>
2445 00015843 A2[759C0100]             <1>       mov    [audio_civ], al
2446 00015848 FEC8                     <1>       dec    al
2447                                    <1>       ;inc   al ; 11/06/2017
2448 0001584A 241F                     <1>       and    al, 1Fh
2449                                    <1>
2450 0001584C 66BA1500                 <1>         mov     dx, PO_LVI_REG
2451 00015850 660315[7A9C0100]         <1>         add  dx, [NABMBAR]
2452 00015857 EE                       <1>         out  dx, al
2453                                    <1>
2454                                    <1>       ; 12/10/2017
2455 00015858 A0[759C0100]             <1>       mov    al, [audio_civ]
2456 0001585D FEC0                     <1>       inc    al
2457 0001585F 2401                     <1>       and    al, 1
2458 00015861 A2[689C0100]             <1>       mov    [audio_flag], al
2459                                    <1>       ;; [audio_flag] : 0 = Buffer 1, 1 = Buffer 2
2460                                    <1>       ;
2461 00015866 58                       <1>       pop    eax
2462                                    <1>       ;
2463 00015867 83E040                   <1>       and     eax, 40h
2464 0001586A 668B15[7A9C0100]         <1>         mov  dx, [NABMBAR]
2465 00015871 6683C230                 <1>       add    dx, GLOB_STS_REG
2466 00015875 EF                       <1>       out    dx, eax
2467                                    <1>
2468                                    <1>       ;; 13/06/2017
2469                                    <1>       ;mov   al, 11h ; IOCE + RPBM
2470                                    <1>       ;mov   dx, PO_CR_REG
2471                                    <1>         ;add dx, [NABMBAR]
2472                                    <1>       ;out   dx, al
2473                                    <1>
2474                                    <1> ac97_tuneloop:
2475                                    <1>       ; 09/10/2017
2476 00015876 8B3D[609C0100]           <1>       mov    edi, [audio_dma_buff]
2477 0001587C 8B0D[649C0100]           <1>       mov    ecx, [audio_dmabuff_size]
2478 00015882 D1E9                     <1>       shr    ecx, 1 ; dma buff size / 2 = half buffer size
2479                                    <1>
2480                                    <1>       ; 12/10/2017
2481 00015884 803D[689C0100]00         <1>       cmp    byte [audio_flag], 0
2482 0001588B 7702                     <1>       ja     short _ac97_ih1  ; Playing Half Buffer 2 (Current: FLAG)
2483                                    <1>       ; Playing Half Buffer 1 (Current: EOL)
2484 0001588D 01CF                     <1>       add    edi, ecx
2485                                    <1> _ac97_ih1:
2486                                    <1>       ; Update half buffer 2 while playing half buffer 1 (next: FLAG)
2487                                    <1>       ; Update half buffer 1 while playing half buffer 2 (next: EOL)
2488                                    <1>
2489 0001588F 8B35[589C0100]           <1>       mov    esi, [audio_p_buffer] ; phy addr of audio buff
2490 00015895 C1E902                   <1>       shr    ecx, 2 ; half buff size / 4
2491 00015898 F3A5                     <1>       rep    movsd
2492                                    <1>
2493                                    <1>       ; 10/10/2017
2494                                    <1>       ; switch flag value
2495 0001589A 8035[689C0100]01         <1>       xor    byte [audio_flag], 1
2496                                    <1>       ; 12/10/2017
2497                                    <1>       ; [audio_flag] = 0 : Playing dma half buffer 2 (even index value)
2498                                    <1>                          ; Next buffer (to update) is dma half buff 1
2499                                    <1>       ;             = 1 : Playing dma half buffer 1 (odd index value)
2500                                    <1>                          ; Next buffer (to update) is dma half buff 2
2501                                    <1>
2502                                    <1> _ac97_ih2:
2503                                    <1>       ;mov   byte [audio_busy], 0
2504                                    <1> _ac97_ih3:
2505                                    <1>       ;pop   edi
2506                                    <1>       ;pop   esi
2507                                    <1>       ;pop   ebx ; * must be restored !
2508                                    <1>       ;pop   ecx
2509                                    <1>       ;pop   edx
2510                                    <1>       ;pop   eax ; * must be restored !
2511                                    <1>
2512 000158A1 C3                       <1>       retn
2513                                    <1>
2514                                    <1> _ac97_ih4:
2515                                    <1>       ; 09/10/2017
2516 000158A2 E818000000               <1>       call   _ac97_stop
2517                                    <1>       ;
2518 000158A7 58                       <1>       pop    eax
2519                                    <1>       ;
2520 000158A8 83E040                   <1>       and     eax, 40h
2521 000158AB 668B15[7A9C0100]         <1>         mov  dx, [NABMBAR]
2522 000158B2 6683C230                 <1>       add    dx, GLOB_STS_REG
2523 000158B6 EF                       <1>       out    dx, eax
2524                                    <1>
2525                                    <1>       ;; 13/06/2017
2526                                    <1>       ;mov   al, 11h ; IOCE + RPBM
2527                                    <1>       ;dx, PO_CR_REG
2528                                    <1>         ;add dx, [NABMBAR]
2529                                    <1>       ;out   dx, al
2530                                    <1>
2531                                    <1>       ; 10/10/2017
2532                                    <1>       ;jmp   short _ac97_ih3  ; exit
2533 000158B7 C3                       <1>       retn
```

```
                                <1>
                                <1> ac97_stop:
                                <1>       ; 28/05/2017
000158B8 C605[749C0100]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
000158BF 30C0                   <1>       xor   al, al ; 0
000158C1 E813000000             <1>       call  ac97_po_cmd
                                <1>
                                <1>       ; (Ref: KolibriOS, intelac97.asm, 'stop:')
                                <1>       ; Clear FIFOE, BCIS, LVBCI (Ref: Intel ICH hub manual)
000158C6 66B81C00               <1>       mov     ax, 1Ch
000158CA 668B15[7A9C0100]       <1>       mov     dx, [NABMBAR]
000158D1 6683C216               <1>       add     dx, PO_SR_REG
000158D5 66EF                   <1>       out     dx, ax
                                <1>
                                <1>       ;retn
                                <1>
                                <1>       ; 11/06/2017
000158D7 B002                   <1>       mov     al, RR
                                <1> ac97_po_cmd:
                                <1>        ;11/06/2017
                                <1>       ; 29/05/2017
000158D9 668B15[7A9C0100]       <1>       mov     dx, [NABMBAR]
000158E0 6683C21B               <1>         add     dx, PO_CR_REG         ; PCM out control register
000158E4 EE                     <1>       out   dx, al
000158E5 C3                     <1>       retn
                                <1>
                                <1> ac97_pause:
                                <1>       ; 11/06/2017
                                <1>       ; 29/05/2017
000158E6 B010                   <1>       mov   al, IOCE
000158E8 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
000158EA 31C0                   <1>       xor     eax, eax
000158EC 66BA0B00               <1>         mov  dx, PI_CR_REG
000158F0 660315[7A9C0100]       <1>       add   dx, [NABMBAR]
000158F7 EE                     <1>       out     dx, al
                                <1>
000158F8 66BA1B00               <1>         mov     dx, PO_CR_REG
000158FC 660315[7A9C0100]       <1>       add   dx, [NABMBAR]
00015903 EE                     <1>       out     dx, al
                                <1>
00015904 66BA2B00               <1>         mov     dx, MC_CR_REG
00015908 660315[7A9C0100]       <1>       add   dx, [NABMBAR]
0001590F EE                     <1>       out     dx, al
                                <1>
00015910 B002                   <1>         mov     al, RR
00015912 66BA0B00               <1>         mov     dx, PI_CR_REG
00015916 660315[7A9C0100]       <1>       add   dx, [NABMBAR]
0001591D EE                     <1>       out     dx, al
                                <1>
0001591E 66BA1B00               <1>         mov     dx, PO_CR_REG
00015922 660315[7A9C0100]       <1>       add   dx, [NABMBAR]
00015929 EE                     <1>       out     dx, al
                                <1>
0001592A 66BA2B00               <1>         mov     dx, MC_CR_REG
0001592E 660315[7A9C0100]       <1>       add   dx, [NABMBAR]
00015935 EE                     <1>       out     dx, al
                                <1>
00015936 C3                     <1>       retn
                                <1>
                                <1> ac97_reset:
                                <1>       ; 10/06/2017
                                <1>       ; 29/05/2017
                                <1>       ; 28/05/2017
00015937 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)
0001593C 66BA2C00               <1>       mov   dx, GLOB_CNT_REG ; 2Ch
00015940 660315[7A9C0100]       <1>       add   dx, [NABMBAR]
00015947 ED                     <1>       in    eax, dx
                                <1>
00015948 A902000000             <1>       test  eax, 2
0001594D 7407                   <1>       jz    short _r_ac97codec_cold
                                <1>
0001594F E80F000000             <1>       call  warm_ac97codec_reset
00015954 7308                   <1>       jnc   short _r_ac97codec_ok
                                <1> _r_ac97codec_cold:
00015956 E83D000000             <1>         call    cold_ac97codec_reset
0001595B 7301                   <1>         jnc     short _r_ac97codec_ok
                                <1>
                                <1>       ; 16/04/2017
                                <1>         ;xor     eax, eax         ; timeout error
                                <1>             ;stc
0001595D C3                     <1>       retn
                                <1>
                                <1> _r_ac97codec_ok:
0001595E 31C0                   <1>         xor     eax, eax
                                <1>         ;mov al, VIA_ACLINK_C00_READY ; 1
00015960 FEC0                   <1>         inc  al
00015962 C3                     <1>       retn
```

```
2639                                    <1>
2640                                    <1> warm_ac97codec_reset:
2641                                    <1>         ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, intelac97.asm)
2642 00015963 B806000000                <1>           mov     eax, 6
2643 00015968 66BA2C00                  <1>         mov     dx, GLOB_CNT_REG ; 2Ch
2644 0001596C 660315[7A9C0100]          <1>         add     dx, [NABMBAR]
2645 00015973 EF                        <1>         out     dx, eax
2646                                    <1>
2647 00015974 B90A000000                <1>         mov     ecx, 10         ; total 1s
2648                                    <1> _warm_ac97c_rst_wait:
2649 00015979 51                        <1>         push    ecx
2650 0001597A E8ECF5FFFF                <1>         call    delay_100ms
2651 0001597F 59                        <1>         pop     ecx
2652                                    <1>
2653 00015980 66BA3000                  <1>         mov     dx, GLOB_STS_REG ; 30h
2654 00015984 660315[7A9C0100]          <1>         add     dx, [NABMBAR]
2655 0001598B ED                        <1>         in      eax, dx
2656                                    <1>
2657 0001598C A900030010                <1>         test    eax, CTRL_ST_CREADY
2658 00015991 7504                      <1>         jnz     short _warm_ac97c_rst_ok
2659                                    <1>
2660 00015993 49                        <1>           dec     ecx
2661 00015994 75E3                      <1>           jnz     short _warm_ac97c_rst_wait
2662                                    <1>
2663                                    <1> _warm_ac97c_rst_fail:
2664 00015996 F9                        <1>           stc
2665                                    <1> _warm_ac97c_rst_ok:
2666 00015997 C3                        <1>         retn
2667                                    <1>
2668                                    <1> cold_ac97codec_reset:
2669                                    <1>         ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, intelac97.asm)
2670 00015998 B802000000                <1>           mov     eax, 2
2671 0001599D 66BA2C00                  <1>         mov     dx, GLOB_CNT_REG ; 2Ch
2672 000159A1 660315[7A9C0100]          <1>         add     dx, [NABMBAR]
2673 000159A8 EF                        <1>         out     dx, eax
2674                                    <1>
2675 000159A9 E8BDF5FFFF                <1>         call    delay_100ms  ; wait 100 ms
2676 000159AE E8B8F5FFFF                <1>         call    delay_100ms  ; wait 100 ms
2677 000159B3 E8B3F5FFFF                <1>         call    delay_100ms  ; wait 100 ms
2678 000159B8 E8AEF5FFFF                <1>         call    delay_100ms  ; wait 100 ms
2679                                    <1>
2680 000159BD B910000000                <1>         mov     ecx, 16         ; total 20*100 ms = 2s
2681                                    <1> _cold_ac97c_rst_wait:
2682 000159C2 66BA3000                  <1>         mov     dx, GLOB_STS_REG ; 30h
2683 000159C6 660315[7A9C0100]          <1>         add     dx, [NABMBAR]
2684 000159CD ED                        <1>         in      eax, dx
2685                                    <1>
2686 000159CE A900030010                <1>         test    eax, CTRL_ST_CREADY
2687 000159D3 750B                      <1>         jnz     short _cold_ac97c_rst_ok
2688                                    <1>
2689 000159D5 51                        <1>         push    ecx
2690 000159D6 E890F5FFFF                <1>         call    delay_100ms
2691 000159DB 59                        <1>         pop     ecx
2692                                    <1>
2693 000159DC 49                        <1>           dec     ecx
2694 000159DD 75E3                      <1>           jnz     short _cold_ac97c_rst_wait
2695                                    <1>
2696                                    <1> _cold_ac97c_rst_fail:
2697 000159DF F9                        <1>           stc
2698                                    <1> _cold_ac97c_rst_ok:
2699 000159E0 C3                        <1>         retn
2700                                    <1>
2701                                    <1> sb16_current_sound_data:
2702                                    <1>         ; 20/08/2017
2703                                    <1>         ; 24/06/2017
2704                                    <1>         ; 22/06/2017
2705                                    <1>         ; get current sound (PCM out) data for graphics
2706                                    <1>         ; (for Sound Blaster 16)
2707                                    <1>         ; ebx = Physical address (on page boundary)
2708                                    <1>         ; ecx = Byte count
2709                                    <1>         ; [audio_buff_size]
2710                                    <1>
2711                                    <1>         ;;mov  edi, [audio_buff_size]
2712                                    <1>         ;mov   edi, [audio_dmabuff_size]
2713                                    <1>         ;mov   esi, [audio_dma_buff]
2714 000159E1 39CF                      <1>         cmp     edi, ecx
2715 000159E3 7302                      <1>         jnb     short sb16_gcd_0
2716 000159E5 89F9                      <1>         mov     ecx, edi
2717                                    <1> sb16_gcd_0:
2718                                    <1>         ; 20/08/2017
2719 000159E7 803D[709C0100]10          <1>         cmp     byte [audio_bps], 16
2720 000159EE 750F                      <1>         jne     short sb16_gcd_1 ; 8 bit DMA channel
2721 000159F0 E4C6                      <1>         in      al, 0C6h ; DMA channel 5 count register
2722 000159F2 88C2                      <1>         mov     dl, al
2723 000159F4 E4C6                      <1>         in      al, 0C6h
2724 000159F6 88C6                      <1>         mov      dh, al
2725 000159F8 0FB7C2                    <1>         movzx   eax, dx
2726 000159FB D1E0                      <1>         shl     eax, 1 ; word count -> byte count
2727 000159FD EB4E                      <1>         jmp     short sb16_gcd_2
2728                                    <1> sb16_gcd_1:
2729 000159FF E403                      <1>         in      al, 03h ; DMA channel 1 count register
2730 00015A01 88C2                      <1>         mov     dl, al
2731 00015A03 E403                      <1>         in      al, 03h
2732 00015A05 88C6                      <1>         mov      dh, al
2733 00015A07 0FB7C2                    <1>         movzx   eax, dx
2734 00015A0A EB41                      <1>         jmp     short sb16_gcd_2
2735                                    <1> ;sb16_gcd_2:
2736                                    <1> ;       cmp     eax, ecx
2737                                    <1> ;       jnb     short sb16_gcd_3
2738                                    <1> ;       ; remain count < graphics bytes
2739                                    <1> ;       mov     eax, ecx ; fix remain count to data size
2740                                    <1> ;sb16_gcd_3:
2741                                    <1> ;       sub     edi, eax
2742                                    <1> ;       jna     short sb16_gcd_4
2743                                    <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]
00015A0C 8B3D[649C0100]         <1>         mov     edi, [audio_dmabuff_size]
00015A12 8B35[609C0100]         <1>         mov     esi, [audio_dma_buff]
00015A18 803D[419C0100]02       <1>         cmp     byte [audio_device], 2
00015A1F 72C0                   <1>         jb      short sb16_current_sound_data ; = 1
00015A21 D1EF                   <1>         shr     edi, 1
00015A23 39CF                   <1>         cmp     edi, ecx
00015A25 7302                   <1>         jnb     short gcd_0
00015A27 89F9                   <1>         mov     ecx, edi
                                <1> gcd_0:
00015A29 803D[419C0100]03       <1>         cmp     byte [audio_device], 3
00015A30 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:
00015A32 BA0C000000             <1>         mov     edx, VIA_REG_OFFSET_CURR_COUNT
00015A37 E88FF5FFFF             <1>         call    ctrl_io_r32
00015A3C 89C2                   <1>         mov     edx, eax ; remain count (bits 23-0),
                                <1>                          ; SGD index (bits 31-24)
00015A3E 81E200000001           <1>         and     edx, 1000000h ; SGD index (0 = 1st half)
00015A44 7402                   <1>         jz      short vt8233_gcd_2
                                <1>         ; the second half of DMA buffer
00015A46 01FE                   <1>         add     esi, edi
                                <1> vt8233_gcd_2:
00015A48 25FFFFFF00             <1>         and     eax, 0FFFFFFh ; bits 23-0
                                <1> ac97_gcd_2:
                                <1> sb16_gcd_2:
00015A4D 39C8                   <1>         cmp     eax, ecx
00015A4F 7302                   <1>         jnb     short vt8233_gcd_3
                                <1>         ; remain count < graphics bytes
00015A51 89C8                   <1>         mov     eax, ecx ; fix remain count to data size
                                <1> vt8233_gcd_3:
00015A53 29C7                   <1>         sub     edi, eax
00015A55 7602                   <1>         jna     short vt8233_gcd_4
00015A57 01FE                   <1>         add     esi, edi ; dma buffer offset
                                <1> vt8233_gcd_4:
00015A59 89DF                   <1>         mov     edi, ebx ; buffer address (for graphics)
00015A5B 890D[64030300]         <1>         mov     [u.r0], ecx
00015A61 F3A4                   <1>         rep     movsb
                                <1> vt8233_gcd_5:
00015A63 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:
00015A64 66BA1400               <1>         mov     dx, PO_CIV_REG ; Position In Current Buff Reg
00015A68 660315[7A9C0100]       <1>         add     dx, [NABMBAR]
00015A6F EC                     <1>         in      al, dx ; current index value
00015A70 A801                   <1>         test    al, 1
00015A72 7402                   <1>         jz      short ac97_gcd_1
00015A74 01FE                   <1>         add     esi, edi
                                <1> ac97_gcd_1:
00015A76 31C0                   <1>         xor     eax, eax
00015A78 66BA1800               <1>         mov     dx, PO_PICB_REG ; Position In Current Buff Reg
00015A7C 660315[7A9C0100]       <1>         add     dx, [NABMBAR]
00015A83 66ED                   <1>         in      ax, dx ; remain dwords
00015A85 C1E002                 <1>         shl     eax, 2 ; remain bytes ; 23/06/2017
00015A88 EBC3                   <1>         jmp     short ac97_gcd_2
                                <1> ;       cmp     eax, ecx
                                <1> ;       jnb     short ac97_gcd_2
                                <1> ;       ; remain count < graphics bytes
```

```
2849                                <1> ;       mov     eax, ecx ; fix remain count to data size
2850                                <1> ;ac97_gcd_2:
2851                                <1> ;       sub     edi, eax
2852                                <1> ;       jna     short ac97_gcd_3
2853                                <1> ;       add     esi, edi ; dma buffer offset
2854                                <1> ;ac97_gcd_3:
2855                                <1> ;       mov     edi, ebx ; buffer address (for graphics)
2856                                <1> ;       mov     [u.r0], ecx
2857                                <1> ;       rep     movsb
2858                                <1> ;       retn
2859                                <1>
2860                                <1> sb16_get_dma_buff_off:
2861                                <1>         ; 28/10/2017
2862                                <1>         ; 24/06/2017
2863                                <1>         ; 22/06/2017
2864                                <1>         ; get current (PCM OUT DMA buffer) pointer
2865                                <1>         ; (for Sound Blaster 16)
2866                                <1>
2867                                <1>         ;mov   ecx, [audio_dmabuff_size]
2868                                <1>         ;xor   ebx, ebx
2869                                <1>         ;shr   ecx, 1
2870                                <1> sb16_gdmabo_0:
2871                                <1>         ; 28/10/2017
2872 00015A8A 803D[709C0100]10      <1>         cmp     byte [audio_bps], 16
2873 00015A91 750F                 <1>         jne     short sb16_gdmabo_1 ; 8 bit DMA channel
2874                                <1>         ; 16 bit DMA channel
2875 00015A93 E4C6                 <1>         in      al, 0C6h ; DMA channel 5 count register
2876 00015A95 88C2                 <1>         mov     dl, al
2877 00015A97 E4C6                 <1>         in      al, 0C6h
2878 00015A99 88C6                 <1>         mov     dh, al
2879 00015A9B 0FB7C2               <1>         movzx   eax, dx
2880 00015A9E D1E0                 <1>         shl     eax, 1 ; word count -> byte count
2881 00015AA0 EB3D                 <1>         jmp     short sb16_gdmabo_2
2882                                <1> sb16_gdmabo_1:
2883 00015AA2 E403                 <1>         in      al, 03h ; DMA channel 1 count register
2884 00015AA4 88C2                 <1>         mov     dl, al
2885 00015AA6 E403                 <1>         in      al, 03h
2886 00015AA8 88C6                 <1>         mov     dh, al
2887 00015AAA 0FB7C2               <1>         movzx   eax, dx
2888 00015AAD EB30                 <1>         jmp     short sb16_gdmabo_2
2889                                <1>
2890                                <1> get_dma_buffer_offset:
2891                                <1>         ; 24/06/2017
2892                                <1>         ; 22/06/2017
2893                                <1>         ; get current sound (PCM out) data for graphics
2894                                <1>         ;
2895                                <1>         ; ebx = Physical address (on page boundary)
2896                                <1>         ; ecx = Byte count
2897                                <1>         ; [audio_buff_size]
2898                                <1>
2899 00015AAF 8B0D[649C0100]       <1>         mov     ecx, [audio_dmabuff_size]
2900 00015AB5 31DB                 <1>         xor     ebx, ebx
2901                                <1> gdmabo_0:
2902 00015AB7 803D[419C0100]02     <1>         cmp     byte [audio_device], 2
2903 00015ABE 72CA                 <1>         jb      short sb16_get_dma_buff_off
2904 00015AC0 742A                 <1>         je      short ac97_get_dma_buff_off
2905                                <1>
2906                                <1> vt8233_get_dma_buff_off:
2907                                <1>         ; 24/06/2017
2908                                <1>         ; 22/06/2017
2909                                <1>         ; get current (PCM OUT DMA buffer) pointer
2910                                <1>         ; (for VT 8233, VT 8237R)
2911                                <1>
2912                                <1>         ;mov   ecx, [audio_dmabuff_size]
2913                                <1>         ;xor   ebx, ebx
2914 00015AC2 D1E9                 <1>         shr     ecx, 1
2915                                <1> vt8233_gdmabo_0:
2916 00015AC4 BA0C000000           <1>         mov     edx, VIA_REG_OFFSET_CURR_COUNT
2917 00015AC9 E8FDF4FFFF           <1>         call    ctrl_io_r32
2918 00015ACE 89C2                 <1>         mov     edx, eax ; remain count (bits 23-0),
2919                                <1>                          ; SGD index (bits 31-24)
2920 00015AD0 81E200000001         <1>         and     edx, 1000000h ; SGD index (0 = 1st half)
2921 00015AD6 7402                 <1>         jz      short vt8233_gdmabo_1
2922                                <1>         ; the second half of DMA buffer
2923 00015AD8 89CB                 <1>         mov     ebx, ecx
2924                                <1> vt8233_gdmabo_1:
2925 00015ADA 25FFFFFF00           <1>         and     eax, 0FFFFFFh ; bits 23-0
2926                                <1> sb16_gdmabo_2:
2927                                <1> ac97_gdmabo_2:
2928 00015ADF 29C1                 <1>         sub     ecx, eax
2929 00015AE1 7602                 <1>         jna     short vt8233_gdmabo_2
2930 00015AE3 01CB                 <1>         add     ebx, ecx ; dma buffer offset
2931                                <1> vt8233_gdmabo_2:
2932 00015AE5 891D[64030300]       <1>         mov     [u.r0], ebx
2933 00015AEB C3                   <1>         retn
2934                                <1>
2935                                <1> ac97_get_dma_buff_off:
2936                                <1>         ; 24/06/2017
2937                                <1>         ; 22/06/2017
2938                                <1>         ; get current (PCM OUT DMA buffer) pointer
2939                                <1>         ; (for AC'97, ICH)
2940                                <1>         ; ebx = Physical address (on page boundary)
2941                                <1>         ; ecx = Byte count
2942                                <1>         ; [audio_buff_size]
2943                                <1>
2944                                <1>         ;mov   ecx, [audio_dmabuff_size]
2945                                <1>         ;xor   ebx, ebx
2946 00015AEC D1E9                 <1>         shr     ecx, 1
2947                                <1> ac97_gdmabo_0:
2948 00015AEE 66BA1400             <1>         mov     dx, PO_CIV_REG ; Position In Current Buff Reg
2949 00015AF2 660315[7A9C0100]     <1>         add     dx, [NABMBAR]
2950 00015AF9 EC                   <1>         in      al, dx ; current index value
2951 00015AFA A801                 <1>         test    al, 1
2952 00015AFC 7402                 <1>         jz      short ac97_gdmabo_1
2953 00015AFE 89CB                 <1>         mov     ebx, ecx
```

```
2954                                <1> ac97_gdmabo_1:
2955 00015B00 31C0                 <1>         xor    eax, eax
2956 00015B02 66BA1800             <1>         mov    dx, PO_PICB_REG ; Position In Current Buff Reg
2957 00015B06 660315[7A9C0100]     <1>         add    dx, [NABMBAR]
2958 00015B0D 66ED                 <1>         in     ax, dx ; remain dwords
2959 00015B0F EBCE                 <1>         jmp    short ac97_gdmabo_2
3425
3426 00015B11 90<rept>                  align 4
3427
3428                                    %include 'vgadata.s' ; 04/07/2016
   1                                <1> ; ********************************************************************************************************
   2                                <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.3 - vgadata.s  (palette and font data)
   3                                <1> ; ---------------------------------------------------------------------------
   4                                <1> ; Last Update: 01/01/2021
   5                                <1> ; ---------------------------------------------------------------------------
   6                                <1> ; Beginning: 16/01/2016
   7                                <1> ; ---------------------------------------------------------------------------
   8                                <1> ; Assembler: NASM version 2.15 (trdos386.s)
   9                                <1> ; ---------------------------------------------------------------------------
  10                                <1> ; Turkish Rational DOS
  11                                <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
  12                                <1> ;
  13                                <1> ; Derived from 'Plex86/Bochs VGABios' source code, vgabios-0.7a (2011)
  14                                <1> ; by the LGPL VGABios Developers Team (2001-2008), 'vgatables.h'
  15                                <1> ;
  16                                <1> ; Oracle VirtualBox 5.0.24 VGABios Source Code
  17                                <1> ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
  18                                <1> ;
  19                                <1> ; Palette and font data in assembly language format:
  20                                <1> ; 'VBoxVgaBiosAlternative.asm'
  21                                <1>
  22                                <1> ; ********************************************************************************************************
  23                                <1>
  24                                <1> ; 25/11/2020 (TRDOS 386 v2.0.3)
  25                                <1> ; ('vgatables.h' - 30/12/2019 - vruppert)
  26                                <1>
  27                                <1> ; 04/07/2016
  28                                <1> ; COLOR DATA
  29                                <1>
  30                                <1> palette0: ; (63+1)*3
  31 00015B14 000000000000000000-   <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
  31 00015B1D 00000000000000        <1>
  32 00015B24 00000000000000002A-   <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
  32 00015B2D 2A2A2A2A2A2A2A        <1>
  33 00015B34 2A2A2A2A2A2A2A2A2A-   <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
  33 00015B3D 2A2A2A2A2A2A2A        <1>
  34 00015B44 2A2A2A2A2A2A2A2A2A-   <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
  34 00015B4D 2A2A2A2A2A2A2A        <1>
  35 00015B54 2A2A2A2A2A2A2A2A3F-   <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh
  35 00015B5D 3F3F3F3F3F3F3F        <1>
  36 00015B64 3F3F3F3F3F3F3F3F3F-   <1>     db  03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh
  36 00015B6D 3F3F3F3F3F3F3F        <1>
  37 00015B74 000000000000000000-   <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
  37 00015B7D 00000000000000        <1>
  38 00015B84 00000000000000002A-   <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
  38 00015B8D 2A2A2A2A2A2A2A        <1>
  39 00015B94 2A2A2A2A2A2A2A2A2A-   <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
  39 00015B9D 2A2A2A2A2A2A2A        <1>
  40 00015BA4 2A2A2A2A2A2A2A2A2A-   <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
  40 00015BAD 2A2A2A2A2A2A2A        <1>
  41 00015BB4 2A2A2A2A2A2A2A2A3F-   <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh
  41 00015BBD 3F3F3F3F3F3F3F        <1>
  42 00015BC4 3F3F3F3F3F3F3F3F3F-   <1>     db  03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh
  42 00015BCD 3F3F3F3F3F3F3F        <1>
  43                                <1> palette1: ; (63+1)*3
  44 00015BD4 00000000002A002A00-   <1>     db  000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah
  44 00015BDD 002A2A2A00002A        <1>
  45 00015BE4 002A2A15002A2A2A00-   <1>     db  000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah
  45 00015BED 000000002A002A        <1>
  46 00015BF4 00002A2A2A00002A00-   <1>     db  000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah, 000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah
  46 00015BFD 2A2A15002A2A2A        <1>
  47 00015C04 15151515153F153F15-   <1>     db  015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh, 015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh
  47 00015C0D 153F3F3F15153F        <1>
  48 00015C14 153F3F3F153F3F3F15-   <1>     db  015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh
  48 00015C1D 151515153F153F        <1>
  49 00015C24 15153F3F3F15153F15-   <1>     db  015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh
  49 00015C2D 3F3F3F153F3F3F        <1>
  50 00015C34 00000000002A002A00-   <1>     db  000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah
  50 00015C3D 002A2A2A00002A        <1>
  51 00015C44 002A2A15002A2A2A00-   <1>     db  000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah
  51 00015C4D 000000002A002A        <1>
  52 00015C54 00002A2A2A00002A00-   <1>     db  000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah, 000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah
  52 00015C5D 2A2A15002A2A2A        <1>
  53 00015C64 15151515153F153F15-   <1>     db  015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh, 015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh
  53 00015C6D 153F3F3F15153F        <1>
  54 00015C74 153F3F3F153F3F3F15-   <1>     db  015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh
  54 00015C7D 151515153F153F        <1>
  55 00015C84 15153F3F3F15153F15-   <1>     db  015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh
  55 00015C8D 3F3F3F153F3F3F        <1>
  56                                <1> palette2: ; (63+1)*3
  57 00015C94 00000000002A002A00-   <1>     db  000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah
  57 00015C9D 002A2A2A00002A        <1>
  58 00015CA4 002A2A2A002A2A2A00-   <1>     db  000h, 02ah, 02ah, 02ah, 000h, 02ah, 02ah, 02ah, 000h, 000h, 015h, 000h, 000h, 03fh, 000h, 02ah
  58 00015CAD 001500003F002A        <1>
  59 00015CB4 15002A3F2A00152A00-   <1>     db  015h, 000h, 02ah, 03fh, 02ah, 000h, 015h, 02ah, 000h, 03fh, 02ah, 02ah, 015h, 02ah, 02ah, 03fh
  59 00015CBD 3F2A2A152A2A3F        <1>
  60 00015CC4 00150000152A003F00-   <1>     db  000h, 015h, 000h, 000h, 015h, 02ah, 000h, 03fh, 000h, 000h, 03fh, 02ah, 02ah, 015h, 000h, 02ah
  60 00015CCD 003F2A2A15002A        <1>
  61 00015CD4 152A2A3F002A3F2A00-   <1>     db  015h, 02ah, 02ah, 03fh, 000h, 02ah, 03fh, 02ah, 000h, 015h, 015h, 000h, 015h, 03fh, 000h, 03fh
  61 00015CDD 151500153F003F        <1>
  62 00015CE4 15003F3F2A15152A15-   <1>     db  015h, 000h, 03fh, 03fh, 02ah, 015h, 015h, 02ah, 015h, 03fh, 02ah, 03fh, 015h, 02ah, 03fh, 03fh
  62 00015CED 3F2A3F152A3F3F        <1>
  63 00015CF4 15000015002A152A00-   <1>     db  015h, 000h, 000h, 015h, 000h, 02ah, 015h, 02ah, 000h, 015h, 02ah, 02ah, 03fh, 000h, 000h, 03fh
  63 00015CFD 152A2A3F00003F        <1>
  64 00015D04 002A3F2A003F2A2A15-   <1>     db  000h, 02ah, 03fh, 02ah, 000h, 03fh, 02ah, 02ah, 015h, 000h, 015h, 015h, 000h, 03fh, 015h, 02ah
  64 00015D0D 001515003F152A        <1>
  65 00015D14 15152A3F3F00153F00-   <1>     db  015h, 015h, 02ah, 03fh, 03fh, 000h, 015h, 03fh, 000h, 03fh, 03fh, 02ah, 015h, 03fh, 02ah, 03fh
  65 00015D1D 3F3F2A153F2A3F        <1>
  66 00015D24 15150015152A153F00-   <1>     db  015h, 015h, 000h, 015h, 015h, 02ah, 015h, 03fh, 000h, 015h, 03fh, 02ah, 03fh, 015h, 000h, 03fh
  66 00015D2D 153F2A3F15003F        <1>
  67 00015D34 152A3F3F003F3F2A15-   <1>     db  015h, 02ah, 03fh, 03fh, 000h, 03fh, 03fh, 02ah, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh
  67 00015D3D 151515153F153F        <1>
  68 00015D44 15153F3F3F15153F15-   <1>     db  015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh
  68 00015D4D 3F3F3F153F3F3F        <1>
  69                                <1> palette3: ; 256*3
  70 00015D54 00000000002A002A00-   <1>     db  000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah
```

```
 70 00015D5D 002A2A2A00002A      <1>
 71 00015D64 002A2A15002A2A2A15- <1>     db  000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh
 71 00015D6D 151515153F153F      <1>
 72 00015D74 15153F3F3F15153F15- <1>     db  015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh
 72 00015D7D 3F3F3F153F3F3F      <1>
 73 00015D84 000000050505080808- <1>     db  000h, 000h, 000h, 005h, 005h, 005h, 008h, 008h, 008h, 00bh, 00bh, 00bh, 00eh, 00eh, 00eh, 011h
 73 00015D8D 0B0B0B0E0E0E11      <1>
 74 00015D94 11111414141818181C- <1>     db  011h, 011h, 014h, 014h, 014h, 018h, 018h, 018h, 01ch, 01ch, 01ch, 020h, 020h, 020h, 024h, 024h
 74 00015D9D 1C1C2020202424      <1>
 75 00015DA4 242828282D2D2D3232- <1>     db  024h, 028h, 028h, 028h, 02dh, 02dh, 02dh, 032h, 032h, 032h, 038h, 038h, 038h, 03fh, 03fh, 03fh
 75 00015DAD 323838383F3F3F      <1>
 76 00015DB4 00003F10003F1F003F- <1>     db  000h, 000h, 03fh, 010h, 000h, 03fh, 01fh, 000h, 03fh, 02fh, 000h, 03fh, 03fh, 000h, 03fh, 03fh
 76 00015DBD 2F003F3F003F3F      <1>
 77 00015DC4 002F3F001F3F00103F- <1>     db  000h, 02fh, 03fh, 000h, 01fh, 03fh, 000h, 010h, 03fh, 000h, 000h, 03fh, 010h, 000h, 03fh, 01fh
 77 00015DCD 00003F10003F1F      <1>
 78 00015DD4 003F2F003F3F002F3F- <1>     db  000h, 03fh, 02fh, 000h, 03fh, 03fh, 000h, 02fh, 03fh, 000h, 01fh, 03fh, 000h, 010h, 03fh, 000h
 78 00015DDD 001F3F00103F00      <1>
 79 00015DE4 003F00003F10003F1F- <1>     db  000h, 03fh, 000h, 000h, 03fh, 010h, 000h, 03fh, 01fh, 000h, 03fh, 02fh, 000h, 03fh, 03fh, 000h
 79 00015DED 003F2F003F3F00      <1>
 80 00015DF4 2F3F001F3F00103F1F- <1>     db  02fh, 03fh, 000h, 01fh, 03fh, 000h, 010h, 03fh, 01fh, 01fh, 03fh, 027h, 01fh, 03fh, 02fh, 01fh
 80 00015DFD 1F3F271F3F2F1F      <1>
 81 00015E04 3F371F3F3F1F3F3F1F- <1>     db  03fh, 037h, 01fh, 03fh, 03fh, 01fh, 03fh, 03fh, 01fh, 037h, 03fh, 01fh, 02fh, 03fh, 01fh, 027h
 81 00015E0D 373F1F2F3F1F27      <1>
 82 00015E14 3F1F1F3F271F3F2F1F- <1>     db  03fh, 01fh, 01fh, 03fh, 027h, 01fh, 03fh, 02fh, 01fh, 03fh, 037h, 01fh, 03fh, 03fh, 01fh, 037h
 82 00015E1D 3F371F3F3F1F37      <1>
 83 00015E24 3F1F2F3F1F273F1F1F- <1>     db  03fh, 01fh, 02fh, 03fh, 01fh, 027h, 03fh, 01fh, 01fh, 03fh, 01fh, 01fh, 03fh, 027h, 01fh, 03fh
 83 00015E2D 3F1F1F3F271F3F      <1>
 84 00015E34 2F1F3F371F3F3F1F37- <1>     db  02fh, 01fh, 03fh, 037h, 01fh, 03fh, 03fh, 01fh, 037h, 03fh, 01fh, 02fh, 03fh, 01fh, 027h, 03fh
 84 00015E3D 3F1F2F3F1F273F      <1>
 85 00015E44 2D2D3F312D3F362D3F- <1>     db  02dh, 02dh, 03fh, 031h, 02dh, 03fh, 036h, 02dh, 03fh, 03ah, 02dh, 03fh, 03fh, 02dh, 03fh, 03fh
 85 00015E4D 3A2D3F3F2D3F3F      <1>
 86 00015E54 2D3A3F2D363F2D313F- <1>     db  02dh, 03ah, 03fh, 02dh, 036h, 03fh, 02dh, 031h, 03fh, 02dh, 02dh, 03fh, 031h, 02dh, 03fh, 036h
 86 00015E5D 2D2D3F312D3F36      <1>
 87 00015E64 2D3F3A2D3F3F2D3A3F- <1>     db  02dh, 03fh, 03ah, 02dh, 03fh, 03fh, 02dh, 03ah, 03fh, 02dh, 036h, 03fh, 02dh, 031h, 03fh, 02dh
 87 00015E6D 2D363F2D313F2D      <1>
 88 00015E74 2D3F2D2D3F312D3F36- <1>     db  02dh, 03fh, 02dh, 02dh, 03fh, 031h, 02dh, 03fh, 036h, 02dh, 03fh, 03ah, 02dh, 03fh, 03fh, 02dh
 88 00015E7D 2D3F3A2D3F3F2D      <1>
 89 00015E84 3A3F2D363F2D313F00- <1>     db  03ah, 03fh, 02dh, 036h, 03fh, 02dh, 031h, 03fh, 000h, 000h, 01ch, 007h, 000h, 01ch, 00eh, 000h
 89 00015E8D 001C07001C0E00      <1>
 90 00015E94 1C15001C1C001C1C00- <1>     db  01ch, 015h, 000h, 01ch, 01ch, 000h, 01ch, 01ch, 000h, 015h, 01ch, 000h, 00eh, 01ch, 000h, 007h
 90 00015E9D 151C000E1C0007      <1>
 91 00015EA4 1C00001C07001C0E00- <1>     db  01ch, 000h, 000h, 01ch, 007h, 000h, 01ch, 00eh, 000h, 01ch, 015h, 000h, 01ch, 01ch, 000h, 015h
 91 00015EAD 1C15001C1C0015      <1>
 92 00015EB4 1C000E1C00071C0000- <1>     db  01ch, 000h, 00eh, 01ch, 000h, 007h, 01ch, 000h, 000h, 01ch, 000h, 000h, 01ch, 007h, 000h, 01ch
 92 00015EBD 1C00001C07001C      <1>
 93 00015EC4 0E001C15001C1C0015- <1>     db  00eh, 000h, 01ch, 015h, 000h, 01ch, 01ch, 000h, 015h, 01ch, 000h, 00eh, 01ch, 000h, 007h, 01ch
 93 00015ECD 1C000E1C00071C      <1>
 94 00015ED4 0E0E1C110E1C150E1C- <1>     db  00eh, 00eh, 01ch, 011h, 00eh, 01ch, 015h, 00eh, 01ch, 018h, 00eh, 01ch, 01ch, 00eh, 01ch, 01ch
 94 00015EDD 180E1C1C0E1C1C      <1>
 95 00015EE4 0E181C0E151C0E111C- <1>     db  00eh, 018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 00eh, 00eh, 01ch, 011h, 00eh, 01ch, 015h
 95 00015EED 0E0E1C110E1C15      <1>
 96 00015EF4 0E1C180E1C1C0E181C- <1>     db  00eh, 01ch, 018h, 00eh, 01ch, 01ch, 00eh, 018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 00eh
 96 00015EFD 0E151C0E111C0E      <1>
 97 00015F04 0E1C0E0E1C110E1C15- <1>     db  00eh, 01ch, 00eh, 00eh, 01ch, 011h, 00eh, 01ch, 015h, 00eh, 01ch, 018h, 00eh, 01ch, 01ch, 00eh
 97 00015F0D 0E1C180E1C1C0E      <1>
 98 00015F14 181C0E151C0E111C14- <1>     db  018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 014h, 014h, 01ch, 016h, 014h, 01ch, 018h, 014h
 98 00015F1D 141C16141C1814      <1>
 99 00015F24 1C1A141C1C141C1C14- <1>     db  01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ch, 01ch, 014h, 01ah, 01ch, 014h, 018h, 01ch, 014h, 016h
 99 00015F2D 1A1C14181C1416      <1>
100 00015F34 1C14141C16141C1814- <1>     db  01ch, 014h, 014h, 01ch, 016h, 014h, 01ch, 018h, 014h, 01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ah
100 00015F3D 1C1A141C1C141A      <1>
101 00015F44 1C14181C14161C1414- <1>     db  01ch, 014h, 018h, 01ch, 014h, 016h, 01ch, 014h, 014h, 01ch, 014h, 014h, 01ch, 016h, 014h, 01ch
101 00015F4D 1C14141C16141C      <1>
102 00015F54 18141C1A141C1C141A- <1>     db  018h, 014h, 01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ah, 01ch, 014h, 018h, 01ch, 014h, 016h, 01ch
102 00015F5D 1C14181C14161C      <1>
103 00015F64 000010040010080010- <1>     db  000h, 000h, 010h, 004h, 000h, 010h, 008h, 000h, 010h, 00ch, 000h, 010h, 010h, 000h, 010h, 010h
103 00015F6D 0C001010001010      <1>
104 00015F74 000C10000810000410- <1>     db  000h, 00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 000h, 000h, 010h, 004h, 000h, 010h, 008h
104 00015F7D 00001004001008      <1>
105 00015F84 00100C001010000C10- <1>     db  000h, 010h, 00ch, 000h, 010h, 010h, 000h, 00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 000h
105 00015F8D 00081000041000      <1>
106 00015F94 001000001004001008- <1>     db  000h, 010h, 000h, 000h, 010h, 004h, 000h, 010h, 008h, 000h, 010h, 00ch, 000h, 010h, 010h, 000h
106 00015F9D 00100C00101000      <1>
107 00015FA4 0C1000081000041008- <1>     db  00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 008h, 008h, 010h, 00ah, 008h, 010h, 00ch, 008h
107 00015FAD 08100A08100C08      <1>
108 00015FB4 100E08101008101008- <1>     db  010h, 00eh, 008h, 010h, 010h, 008h, 010h, 010h, 008h, 00eh, 010h, 008h, 00ch, 010h, 008h, 00ah
108 00015FBD 0E10080C10080A      <1>
109 00015FC4 100808100A08100C08- <1>     db  010h, 008h, 008h, 010h, 00ah, 008h, 010h, 00ch, 008h, 010h, 00eh, 008h, 010h, 010h, 008h, 00eh
109 00015FCD 100E081010080E      <1>
110 00015FD4 10080C10080A100808- <1>     db  010h, 008h, 00ch, 010h, 008h, 00ah, 010h, 008h, 008h, 010h, 008h, 008h, 010h, 00ah, 008h, 010h
110 00015FDD 100808100A0810      <1>
111 00015FE4 0C08100E081010080E- <1>     db  00ch, 008h, 010h, 00eh, 008h, 010h, 010h, 008h, 00eh, 010h, 008h, 00ch, 010h, 008h, 00ah, 010h
111 00015FED 10080C10080A10      <1>
112 00015FF4 0B0B100C0B100D0B10- <1>     db  00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh, 00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh, 010h, 010h
112 00015FFD 0F0B10100B1010      <1>
113 00016004 0B0F100B0D100B0C10- <1>     db  00bh, 00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh
113 0001600D 0B0B100C0B100D      <1>
114 00016014 0B100F0B10100B0F10- <1>     db  00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh, 00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 00bh
114 0001601D 0B0D100B0C100B      <1>
115 00016024 0B100B0B100C0B100D- <1>     db  00bh, 010h, 00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh, 00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh
115 0001602D 0B100F0B10100B      <1>
116 00016034 0F100B0D100B0C1000- <1>     db  00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
116 0001603D 00000000000000      <1>
117 00016044 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
117 0001604D 00000000000000      <1>
118                              <1>
119                              <1>
120                              <1> ; 04/07/2016
121                              <1> ; FONT DATA
122                              <1>
123                              <1> CRT_CHAR_GEN:
124                              <1> vgafont8:
125 00016054 00000000000000007E- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 081h, 0a5h, 081h, 0bdh, 099h, 081h, 07eh
125 0001605D 81A581BD99817E      <1>
126 00016064 7EFFDBFFC3E7FF7E6C- <1>     db  07eh, 0ffh, 0dbh, 0ffh, 0c3h, 0e7h, 0ffh, 07eh, 06ch, 0feh, 0feh, 0feh, 07ch, 038h, 010h, 000h
126 0001606D FEFEFE7C381000      <1>
127 00016074 10387CFE7C38100038- <1>     db  010h, 038h, 07ch, 0feh, 07ch, 038h, 010h, 000h, 038h, 07ch, 038h, 0feh, 0feh, 07ch, 038h, 07ch
127 0001607D 7C38FEFE7C387C      <1>
128 00016084 1010387CFE7C387C00- <1>     db  010h, 010h, 038h, 07ch, 0feh, 07ch, 038h, 07ch, 000h, 000h, 018h, 03ch, 03ch, 018h, 000h, 000h
128 0001608D 00183C3C180000      <1>
129 00016094 FFFFE7C3C3E7FFFF00- <1>     db  0ffh, 0ffh, 0e7h, 0c3h, 0c3h, 0e7h, 0ffh, 0ffh, 000h, 03ch, 066h, 042h, 042h, 066h, 03ch, 000h
129 0001609D 3C664242663C00      <1>
130 000160A4 FFC399BDBD99C3FF0F- <1>     db  0ffh, 0c3h, 099h, 0bdh, 0bdh, 099h, 0c3h, 0ffh, 00fh, 007h, 00fh, 07dh, 0cch, 0cch, 0cch, 078h
130 000160AD 070F7DCCCCCC78      <1>
131 000160B4 3C6666663C187E183F- <1>     db  03ch, 066h, 066h, 066h, 03ch, 018h, 07eh, 018h, 03fh, 033h, 03fh, 030h, 030h, 070h, 0f0h, 0e0h
131 000160BD 333F303070F0E0      <1>
132 000160C4 7F637F636367E6C099- <1>     db  07fh, 063h, 07fh, 063h, 063h, 067h, 0e6h, 0c0h, 099h, 05ah, 03ch, 0e7h, 0e7h, 03ch, 05ah, 099h
132 000160CD 5A3CE7E73C5A99      <1>
```

```
133 000160D4 80E0F8FEF8E0800002- <1>     db  080h, 0e0h, 0f8h, 0feh, 0f8h, 0e0h, 080h, 000h, 002h, 00eh, 03eh, 0feh, 03eh, 00eh, 002h, 000h
133 000160DD 0E3EFE3E0E0200      <1>
134 000160E4 183C7E18187E3C1866- <1>     db  018h, 03ch, 07eh, 018h, 018h, 07eh, 03ch, 018h, 066h, 066h, 066h, 066h, 066h, 000h, 066h, 000h
134 000160ED 66666666006600      <1>
135 000160F4 7FDBDB7B1B1B1B003E- <1>     db  07fh, 0dbh, 0dbh, 07bh, 01bh, 01bh, 01bh, 000h, 03eh, 063h, 038h, 06ch, 06ch, 038h, 0cch, 078h
135 000160FD 63386C6C38CC78      <1>
136 00016104 000000007E7E7E0018- <1>     db  000h, 000h, 000h, 000h, 07eh, 07eh, 07eh, 000h, 018h, 03ch, 07eh, 018h, 07eh, 03ch, 018h, 0ffh
136 0001610D 3C7E187E3C18FF      <1>
137 00016114 183C7E181818180018- <1>     db  018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h
137 0001611D 1818187E3C1800      <1>
138 00016124 00180CFE0C18000000- <1>     db  000h, 018h, 00ch, 0feh, 00ch, 018h, 000h, 000h, 000h, 030h, 060h, 0feh, 060h, 030h, 000h, 000h
138 0001612D 3060FE60300000      <1>
139 00016134 0000C0C0C0FE000000- <1>     db  000h, 000h, 0c0h, 0c0h, 0c0h, 0feh, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h
139 0001613D 2466FF66240000      <1>
140 00016144 00183C7EFFFF000000- <1>     db  000h, 018h, 03ch, 07eh, 0ffh, 0ffh, 000h, 000h, 000h, 0ffh, 0ffh, 07eh, 03ch, 018h, 000h, 000h
140 0001614D FFFF7E3C180000      <1>
141 00016154 000000000000000030- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 030h, 078h, 078h, 030h, 030h, 000h, 030h, 000h
141 0001615D 78783030003000      <1>
142 00016164 6C6C6C00000000006C- <1>     db  06ch, 06ch, 06ch, 000h, 000h, 000h, 000h, 000h, 06ch, 06ch, 0feh, 06ch, 0feh, 06ch, 06ch, 000h
142 0001616D 6CFE6CFE6C6C00      <1>
143 00016174 307CC0780CF8300000- <1>     db  030h, 07ch, 0c0h, 078h, 00ch, 0f8h, 030h, 000h, 000h, 0c6h, 0cch, 018h, 030h, 066h, 0c6h, 000h
143 0001617D C6CC183066C600      <1>
144 00016184 386C3876DCCC760060- <1>     db  038h, 06ch, 038h, 076h, 0dch, 0cch, 076h, 000h, 060h, 060h, 0c0h, 000h, 000h, 000h, 000h, 000h
144 0001618D 60C000000000000     <1>
145 00016194 183060606030180060- <1>     db  018h, 030h, 060h, 060h, 060h, 030h, 018h, 000h, 060h, 030h, 018h, 018h, 018h, 030h, 060h, 000h
145 0001619D 30181818306000      <1>
146 000161A4 00663CFF3C66000000- <1>     db  000h, 066h, 03ch, 0ffh, 03ch, 066h, 000h, 000h, 000h, 030h, 030h, 0fch, 030h, 030h, 000h, 000h
146 000161AD 3030FC30300000      <1>
147 000161B4 000000000030306000- <1>     db  000h, 000h, 000h, 000h, 000h, 030h, 030h, 060h, 000h, 000h, 000h, 0fch, 000h, 000h, 000h, 000h
147 000161BD 0000FC00000000      <1>
148 000161C4 000000000030300006- <1>     db  000h, 000h, 000h, 000h, 000h, 030h, 030h, 000h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 080h, 000h
148 000161CD 0C183060C08000      <1>
149 000161D4 7CC6CEDEF6E67C0030- <1>     db  07ch, 0c6h, 0ceh, 0deh, 0f6h, 0e6h, 07ch, 000h, 030h, 070h, 030h, 030h, 030h, 030h, 0fch, 000h
149 000161DD 7030303030FC00      <1>
150 000161E4 78CC0C3860CCFC0078- <1>     db  078h, 0cch, 00ch, 038h, 060h, 0cch, 0fch, 000h, 078h, 0cch, 00ch, 038h, 00ch, 0cch, 078h, 000h
150 000161ED CC0C380CCC7800      <1>
151 000161F4 1C3C6CCCFE0C1E00FC- <1>     db  01ch, 03ch, 06ch, 0cch, 0feh, 00ch, 01eh, 000h, 0fch, 0c0h, 0f8h, 00ch, 00ch, 0cch, 078h, 000h
151 000161FD C0F80C0CCC7800      <1>
152 00016204 3860C0F8CCCC7800FC- <1>     db  038h, 060h, 0c0h, 0f8h, 0cch, 0cch, 078h, 000h, 0fch, 0cch, 00ch, 018h, 030h, 030h, 030h, 000h
152 0001620D CC0C1830303000      <1>
153 00016214 78CCCC78CCCC780078- <1>     db  078h, 0cch, 0cch, 078h, 0cch, 0cch, 078h, 000h, 078h, 0cch, 0cch, 07ch, 00ch, 018h, 070h, 000h
153 0001621D CCCC7C0C187000      <1>
154 00016224 0030300000303000000- <1>     db  000h, 030h, 030h, 000h, 000h, 030h, 030h, 000h, 000h, 030h, 030h, 000h, 000h, 030h, 030h, 060h
154 0001622D 30300000303060      <1>
155 00016234 183060C06030180000- <1>     db  018h, 030h, 060h, 0c0h, 060h, 030h, 018h, 000h, 000h, 000h, 0fch, 000h, 000h, 0fch, 000h, 000h
155 0001623D 00FC0000FC0000      <1>
156 00016244 6030180C1830600078- <1>     db  060h, 030h, 018h, 00ch, 018h, 030h, 060h, 000h, 078h, 0cch, 00ch, 018h, 030h, 000h, 030h, 000h
156 0001624D CC0C1830003000      <1>
157 00016254 7CC6DEDEDEC0780030- <1>     db  07ch, 0c6h, 0deh, 0deh, 0deh, 0c0h, 078h, 000h, 030h, 078h, 0cch, 0cch, 0fch, 0cch, 0cch, 000h
157 0001625D 78CCCCFCCCCC00      <1>
158 00016264 FC66667C6666FC003C- <1>     db  0fch, 066h, 066h, 07ch, 066h, 066h, 0fch, 000h, 03ch, 066h, 0c0h, 0c0h, 0c0h, 066h, 03ch, 000h
158 0001626D 66C0C0C0663C00      <1>
159 00016274 F86C6666666CF800FE- <1>     db  0f8h, 06ch, 066h, 066h, 066h, 06ch, 0f8h, 000h, 0feh, 062h, 068h, 078h, 068h, 062h, 0feh, 000h
159 0001627D 6268786862FE00      <1>
160 00016284 FE6268786860F0003C- <1>     db  0feh, 062h, 068h, 078h, 068h, 060h, 0f0h, 000h, 03ch, 066h, 0c0h, 0c0h, 0ceh, 066h, 03eh, 000h
160 0001628D 66C0C0CE663E00      <1>
161 00016294 CCCCCCFCCCCCCC0078- <1>     db  0cch, 0cch, 0cch, 0fch, 0cch, 0cch, 0cch, 000h, 078h, 030h, 030h, 030h, 030h, 030h, 078h, 000h
161 0001629D 30303030307800      <1>
162 000162A4 1E0C0C0CCCCC7800E6- <1>     db  01eh, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 000h, 0e6h, 066h, 06ch, 078h, 06ch, 066h, 0e6h, 000h
162 000162AD 666C786C66E600      <1>
163 000162B4 F06060606266FE00C6- <1>     db  0f0h, 060h, 060h, 060h, 062h, 066h, 0feh, 000h, 0c6h, 0eeh, 0feh, 0feh, 0d6h, 0c6h, 0c6h, 000h
163 000162BD EEFEFED6C6C600      <1>
164 000162C4 C6E6F6DECEC6C60038- <1>     db  0c6h, 0e6h, 0f6h, 0deh, 0ceh, 0c6h, 0c6h, 000h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h
164 000162CD 6CC6C6C66C3800      <1>
165 000162D4 FC66667C6060F00078- <1>     db  0fch, 066h, 066h, 07ch, 060h, 060h, 0f0h, 000h, 078h, 0cch, 0cch, 0cch, 0dch, 078h, 01ch, 000h
165 000162DD CCCCCCDC781C00      <1>
166 000162E4 FC66667C6C66E60078- <1>     db  0fch, 066h, 066h, 07ch, 06ch, 066h, 0e6h, 000h, 078h, 0cch, 0e0h, 070h, 01ch, 0cch, 078h, 000h
166 000162ED CCE0701CCC7800      <1>
167 000162F4 FCB4303030307800CC- <1>     db  0fch, 0b4h, 030h, 030h, 030h, 030h, 078h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 0fch, 000h
167 000162FD CCCCCCCCCCFC00      <1>
168 00016304 CCCCCCCCCC783000C6- <1>     db  0cch, 0cch, 0cch, 0cch, 0cch, 078h, 030h, 000h, 0c6h, 0c6h, 0c6h, 0d6h, 0feh, 0eeh, 0c6h, 000h
168 0001630D C6C6D6FEEEC600      <1>
169 00016314 C6C66C38386CC600CC- <1>     db  0c6h, 0c6h, 06ch, 038h, 038h, 06ch, 0c6h, 000h, 0cch, 0cch, 0cch, 078h, 030h, 030h, 078h, 000h
169 0001631D CCCC7830307800      <1>
170 00016324 FEC68C183266FE0078- <1>     db  0feh, 0c6h, 08ch, 018h, 032h, 066h, 0feh, 000h, 078h, 060h, 060h, 060h, 060h, 060h, 078h, 000h
170 0001632D 60606060607800      <1>
171 00016334 C06030180C06020078- <1>     db  0c0h, 060h, 030h, 018h, 00ch, 006h, 002h, 000h, 078h, 018h, 018h, 018h, 018h, 018h, 078h, 000h
171 0001633D 18181818187800      <1>
172 00016344 10386CC600000000000- <1>     db  010h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh
172 0001634D 000000000000FF      <1>
173 00016354 3030180000000000000- <1>     db  030h, 030h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 078h, 00ch, 07ch, 0cch, 076h, 000h
173 0001635D 00780C7CCC7600      <1>
174 00016364 E060607C6666DC0000- <1>     db  0e0h, 060h, 060h, 07ch, 066h, 066h, 0dch, 000h, 000h, 000h, 078h, 0cch, 0c0h, 0cch, 078h, 000h
174 0001636D 0078CCC0CC7800      <1>
175 00016374 1C0C0C7CCCCC760000- <1>     db  01ch, 00ch, 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h
175 0001637D 0078CCFCC07800      <1>
176 00016384 386C60F06060F00000- <1>     db  038h, 06ch, 060h, 0f0h, 060h, 060h, 0f0h, 000h, 000h, 000h, 076h, 0cch, 0cch, 07ch, 00ch, 0f8h
176 0001638D 0076CCCC7C0CF8      <1>
177 00016394 E0606C766666E60030- <1>     db  0e0h, 060h, 06ch, 076h, 066h, 066h, 0e6h, 000h, 030h, 000h, 070h, 030h, 030h, 030h, 078h, 000h
177 0001639D 00703030307800      <1>
178 000163A4 0C000C0C0CCCCC78E0- <1>     db  00ch, 000h, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 0e0h, 060h, 066h, 06ch, 078h, 06ch, 0e6h, 000h
178 000163AD 60666C786CE600      <1>
179 000163B4 7030303030307800000- <1>     db  070h, 030h, 030h, 030h, 030h, 030h, 078h, 000h, 000h, 000h, 0cch, 0feh, 0feh, 0d6h, 0c6h, 000h
179 000163BD 00CCFEFED6C600      <1>
180 000163C4 0000F8CCCCCCCC0000- <1>     db  000h, 000h, 0f8h, 0cch, 0cch, 0cch, 0cch, 000h, 000h, 000h, 078h, 0cch, 0cch, 0cch, 078h, 000h
180 000163CD 0078CCCCCC7800      <1>
181 000163D4 0000DC66667C60F000- <1>     db  000h, 000h, 0dch, 066h, 066h, 07ch, 060h, 0f0h, 000h, 000h, 076h, 0cch, 0cch, 07ch, 00ch, 01eh
181 000163DD 0076CCCC7C0C1E      <1>
182 000163E4 0000DC766660F00000- <1>     db  000h, 000h, 0dch, 076h, 066h, 060h, 0f0h, 000h, 000h, 000h, 07ch, 0c0h, 078h, 00ch, 0f8h, 000h
182 000163ED 007CC0780CF800      <1>
183 000163F4 10307C303034180000- <1>     db  010h, 030h, 07ch, 030h, 030h, 034h, 018h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 076h, 000h
183 000163FD 00CCCCCCCC7600      <1>
184 00016404 0000CCCCCC78300000- <1>     db  000h, 000h, 0cch, 0cch, 0cch, 078h, 030h, 000h, 000h, 000h, 0c6h, 0d6h, 0feh, 0feh, 06ch, 000h
184 0001640D 00C6D6FEFE6C00      <1>
185 00016414 0000C66C386CC60000- <1>     db  000h, 000h, 0c6h, 06ch, 038h, 06ch, 0c6h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 07ch, 00ch, 0f8h
185 0001641D 00CCCCCC7C0CF8      <1>
186 00016424 0000FC983064FC001C- <1>     db  000h, 000h, 0fch, 098h, 030h, 064h, 0fch, 000h, 01ch, 030h, 030h, 0e0h, 030h, 030h, 01ch, 000h
186 0001642D 3030E030301C00      <1>
187 00016434 18181800181818000E0- <1>     db  018h, 018h, 018h, 000h, 018h, 018h, 018h, 000h, 0e0h, 030h, 030h, 01ch, 030h, 030h, 0e0h, 000h
187 0001643D 30301C3030E000      <1>
188 00016444 76DC00000000000000- <1>     db  076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 000h
188 0001644D 10386CC6C6FE00      <1>
189 00016454 78CCC0CC78180C7800- <1>     db  078h, 0cch, 0c0h, 0cch, 078h, 018h, 00ch, 078h, 000h, 0cch, 000h, 0cch, 0cch, 0cch, 07eh, 000h
189 0001645D CC00CCCCCC7E00      <1>
190 00016464 1C0078CCFCC078007E- <1>     db  01ch, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h, 07eh, 0c3h, 03ch, 006h, 03eh, 066h, 03fh, 000h
190 0001646D C33C063E663F00      <1>
191 00016474 CC00780C7CCC7E00E0- <1>     db  0cch, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 0e0h, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h
191 0001647D 00780C7CCC7E00      <1>
```

```
192 00016484 3030780C7CCC7E0000- <1>     db  030h, 030h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 000h, 000h, 078h, 0c0h, 0c0h, 078h, 00ch, 038h
192 0001648D 0078C0C0780C38      <1>
193 00016494 7EC33C667E603C00CC- <1>     db  07eh, 0c3h, 03ch, 066h, 07eh, 060h, 03ch, 000h, 0cch, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h
193 0001649D 0078CCFCC07800      <1>
194 000164A4 E00078CCFCC07800CC- <1>     db  0e0h, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h, 0cch, 000h, 070h, 030h, 030h, 030h, 078h, 000h
194 000164AD 00703030307800      <1>
195 000164B4 7CC6381818183C00E0- <1>     db  07ch, 0c6h, 038h, 018h, 018h, 018h, 03ch, 000h, 0e0h, 000h, 070h, 030h, 030h, 030h, 078h, 000h
195 000164BD 00703030307800      <1>
196 000164C4 C6386CC6FEC6C60030- <1>     db  0c6h, 038h, 06ch, 0c6h, 0feh, 0c6h, 0c6h, 000h, 030h, 030h, 000h, 078h, 0cch, 0fch, 0cch, 000h
196 000164CD 300078CCFCCC00      <1>
197 000164D4 1C00FC607860FC0000- <1>     db  01ch, 000h, 0fch, 060h, 078h, 060h, 0fch, 000h, 000h, 000h, 07fh, 00ch, 07fh, 0cch, 07fh, 000h
197 000164DD 007F0C7FCC7F00      <1>
198 000164E4 3E6CCCFECCCCCE0078- <1>     db  03eh, 06ch, 0cch, 0feh, 0cch, 0cch, 0ceh, 000h, 078h, 0cch, 000h, 078h, 0cch, 0cch, 078h, 000h
198 000164ED CC0078CCCC7800      <1>
199 000164F4 00CC0078CCCC780000- <1>     db  000h, 0cch, 000h, 078h, 0cch, 0cch, 078h, 000h, 000h, 0e0h, 000h, 078h, 0cch, 0cch, 078h, 000h
199 000164FD E00078CCCC7800      <1>
200 00016504 78CC00CCCCCC7E0000- <1>     db  078h, 0cch, 000h, 0cch, 0cch, 0cch, 07eh, 000h, 000h, 0e0h, 000h, 0cch, 0cch, 0cch, 07eh, 000h
200 0001650D E000CCCCCC7E00      <1>
201 00016514 00CC00CCCC7C0CF8C3- <1>     db  000h, 0cch, 000h, 0cch, 0cch, 07ch, 00ch, 0f8h, 0c3h, 018h, 03ch, 066h, 066h, 03ch, 018h, 000h
201 0001651D 183C66663C1800      <1>
202 00016524 CC00CCCCCCCC780018- <1>     db  0cch, 000h, 0cch, 0cch, 0cch, 0cch, 078h, 000h, 018h, 018h, 07eh, 0c0h, 0c0h, 07eh, 018h, 018h
202 0001652D 187EC0C07E1818      <1>
203 00016534 386C64F060E6FC00CC- <1>     db  038h, 06ch, 064h, 0f0h, 060h, 0e6h, 0fch, 000h, 0cch, 0cch, 078h, 0fch, 030h, 0fch, 030h, 030h
203 0001653D CC78FC30FC3030      <1>
204 00016544 F8CCCCFAC6CFC6C70E- <1>     db  0f8h, 0cch, 0cch, 0fah, 0c6h, 0cfh, 0c6h, 0c7h, 00eh, 01bh, 018h, 03ch, 018h, 018h, 0d8h, 070h
204 0001654D 1B183C1818D870      <1>
205 00016554 1C00780C7CCC7E0038- <1>     db  01ch, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 038h, 000h, 070h, 030h, 030h, 030h, 078h, 000h
205 0001655D 00703030307800      <1>
206 00016564 001C0078CCCC780000- <1>     db  000h, 01ch, 000h, 078h, 0cch, 0cch, 078h, 000h, 000h, 01ch, 000h, 0cch, 0cch, 0cch, 07eh, 000h
206 0001656D 1C00CCCCCC7E00      <1>
207 00016574 00F800F8CCCCCC00FC- <1>     db  000h, 0f8h, 000h, 0f8h, 0cch, 0cch, 0cch, 000h, 0fch, 000h, 0cch, 0ech, 0fch, 0dch, 0cch, 000h
207 0001657D 00CCECFCDCCC00      <1>
208 00016584 3C6C6C3E007E000038- <1>     db  03ch, 06ch, 06ch, 03eh, 000h, 07eh, 000h, 000h, 038h, 06ch, 06ch, 038h, 000h, 07ch, 000h, 000h
208 0001658D 6C6C38007C0000      <1>
209 00016594 30003060C0CC780000- <1>     db  030h, 000h, 030h, 060h, 0c0h, 0cch, 078h, 000h, 000h, 000h, 000h, 0fch, 0c0h, 0c0h, 000h, 000h
209 0001659D 0000FCC0C00000      <1>
210 000165A4 000000FC0C0C0000C3- <1>     db  000h, 000h, 000h, 0fch, 00ch, 00ch, 000h, 000h, 0c3h, 0c6h, 0cch, 0deh, 033h, 066h, 0cch, 00fh
210 000165AD C6CCDE3366CC0F      <1>
211 000165B4 C3C6CCDB376FCF0318- <1>     db  0c3h, 0c6h, 0cch, 0dbh, 037h, 06fh, 0cfh, 003h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 000h
211 000165BD 18001818181800      <1>
212 000165C4 003366CC6633000000- <1>     db  000h, 033h, 066h, 0cch, 066h, 033h, 000h, 000h, 000h, 0cch, 066h, 033h, 066h, 0cch, 000h, 000h
212 000165CD CC663366CC0000      <1>
213 000165D4 228822882288228855- <1>     db  022h, 088h, 022h, 088h, 022h, 088h, 022h, 088h, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah
213 000165DD AA55AA55AA55AA      <1>
214 000165E4 DB77DBEEDB77DBEE18- <1>     db  0dbh, 077h, 0dbh, 0eeh, 0dbh, 077h, 0dbh, 0eeh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
214 000165ED 18181818181818      <1>
215 000165F4 18181818F818181818- <1>     db  018h, 018h, 018h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h, 018h
215 000165FD 18F818F8181818      <1>
216 00016604 36363636F636363600- <1>     db  036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h
216 0001660D 000000FE363636      <1>
217 00016614 0000F818F818181836- <1>     db  000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h, 036h
217 0001661D 36F606F6363636      <1>
218 00016624 363636363636363600- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 000h, 000h, 0feh, 006h, 0f6h, 036h, 036h, 036h
218 0001662D 00FE06F6363636      <1>
219 00016634 3636F606FE00000036- <1>     db  036h, 036h, 0f6h, 006h, 0feh, 000h, 000h, 000h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h, 000h
219 0001663D 363636FE000000      <1>
220 00016644 1818F818F800000000- <1>     db  018h, 018h, 0f8h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 018h, 018h
220 0001664D 000000F8181818      <1>
221 00016654 181818181F00000018- <1>     db  018h, 018h, 018h, 018h, 01fh, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 0ffh, 000h, 000h, 000h
221 0001665D 181818FF000000      <1>
222 00016664 00000000FF18181818- <1>     db  000h, 000h, 000h, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 018h, 018h
222 0001666D 1818181F181818      <1>
223 00016674 00000000FF00000018- <1>     db  000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h
223 0001667D 181818FF181818      <1>
224 00016684 18181F181F18181836- <1>     db  018h, 018h, 01fh, 018h, 01fh, 018h, 018h, 018h, 036h, 036h, 036h, 036h, 037h, 036h, 036h, 036h
224 0001668D 36363637363636      <1>
225 00016694 363637303F00000000- <1>     db  036h, 036h, 037h, 030h, 03fh, 000h, 000h, 000h, 000h, 000h, 03fh, 030h, 037h, 036h, 036h, 036h
225 0001669D 003F3037363636      <1>
226 000166A4 3636F700FF00000000- <1>     db  036h, 036h, 0f7h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0f7h, 036h, 036h, 036h
226 000166AD 00FF00F7363636      <1>
227 000166B4 363637303736363600- <1>     db  036h, 036h, 037h, 030h, 037h, 036h, 036h, 036h, 000h, 000h, 0ffh, 000h, 0ffh, 000h, 000h, 000h
227 000166BD 00FF00FF000000      <1>
228 000166C4 3636F700F736363618- <1>     db  036h, 036h, 0f7h, 000h, 0f7h, 036h, 036h, 036h, 018h, 018h, 0ffh, 000h, 0ffh, 000h, 000h, 000h
228 000166CD 18FF00FF000000      <1>
229 000166D4 36363636FF00000000- <1>     db  036h, 036h, 036h, 036h, 0ffh, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 018h, 018h, 018h
229 000166DD 00FF00FF181818      <1>
230 000166E4 00000000FF36363636- <1>     db  000h, 000h, 000h, 000h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 03fh, 000h, 000h, 000h
230 000166ED 3636363F000000      <1>
231 000166F4 18181F181F00000000- <1>     db  018h, 018h, 01fh, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h, 018h
231 000166FD 001F181F181818      <1>
232 00016704 000000003F36363636- <1>     db  000h, 000h, 000h, 000h, 03fh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h
232 0001670D 363636FF363636      <1>
233 00016714 1818FF18FF18181818- <1>     db  018h, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 000h, 000h, 000h
233 0001671D 181818F8000000      <1>
234 00016724 000000001F181818FF- <1>     db  000h, 000h, 000h, 000h, 01fh, 018h, 018h, 018h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
234 0001672D FFFFFFFFFFFFFF      <1>
235 00016734 00000000FFFFFFFFF0- <1>     db  000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h
235 0001673D F0F0F0F0F0F0F0      <1>
236 00016744 0F0F0F0F0F0F0F0FFF- <1>     db  00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h
236 0001674D FFFFFF00000000      <1>
237 00016754 000076DCC8DC760000- <1>     db  000h, 000h, 076h, 0dch, 0c8h, 0dch, 076h, 000h, 000h, 078h, 0cch, 0f8h, 0cch, 0f8h, 0c0h, 0c0h
237 0001675D 78CCF8CCF8C0C0      <1>
238 00016764 00FCCCC0C0C0C00000- <1>     db  000h, 0fch, 0cch, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 0feh, 06ch, 06ch, 06ch, 06ch, 06ch, 000h
238 0001676D FE6C6C6C6C6C00      <1>
239 00016774 FCCC603060CCFC0000- <1>     db  0fch, 0cch, 060h, 030h, 060h, 0cch, 0fch, 000h, 000h, 000h, 07eh, 0d8h, 0d8h, 0d8h, 070h, 000h
239 0001677D 007ED8D8D87000      <1>
240 00016784 006666666667C60C000- <1>     db  000h, 066h, 066h, 066h, 066h, 07ch, 060h, 0c0h, 000h, 076h, 0dch, 018h, 018h, 018h, 018h, 000h
240 0001678D 76DC1818181800      <1>
241 00016794 FC3078CCCC7830FC38- <1>     db  0fch, 030h, 078h, 0cch, 0cch, 078h, 030h, 0fch, 038h, 06ch, 0c6h, 0feh, 0c6h, 06ch, 038h, 000h
241 0001679D 6CC6FEC66C3800      <1>
242 000167A4 386CC6C66C6CEE001C- <1>     db  038h, 06ch, 0c6h, 0c6h, 06ch, 06ch, 0eeh, 000h, 01ch, 030h, 018h, 07ch, 0cch, 0cch, 078h, 000h
242 000167AD 30187CCCCC7800      <1>
243 000167B4 00007EDBDB7E000006- <1>     db  000h, 000h, 07eh, 0dbh, 0dbh, 07eh, 000h, 000h, 006h, 00ch, 07eh, 0dbh, 0dbh, 07eh, 060h, 0c0h
243 000167BD 0C7EDBDB7E60C0      <1>
244 000167C4 3860C0F8C060380078- <1>     db  038h, 060h, 0c0h, 0f8h, 0c0h, 060h, 038h, 000h, 078h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 000h
244 000167CD CCCCCCCCCCCC00      <1>
245 000167D4 00FC00FC00FC000030- <1>     db  000h, 0fch, 000h, 0fch, 000h, 0fch, 000h, 000h, 030h, 030h, 0fch, 030h, 030h, 000h, 0fch, 000h
245 000167DD 30FC303000FC00      <1>
246 000167E4 603018306000FC0018- <1>     db  060h, 030h, 018h, 030h, 060h, 000h, 0fch, 000h, 018h, 030h, 060h, 030h, 018h, 000h, 0fch, 000h
246 000167ED 3060301800FC00      <1>
247 000167F4 0E1B1B181818181818- <1>     db  00eh, 01bh, 01bh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0d8h, 0d8h, 070h
247 000167FD 18181818D8D870      <1>
248 00016804 303000FC00303000000- <1>     db  030h, 030h, 000h, 0fch, 000h, 030h, 030h, 000h, 000h, 076h, 0dch, 000h, 076h, 0dch, 000h, 000h
248 0001680D 76DC0076DC0000      <1>
249 00016814 386C6C380000000000- <1>     db  038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h
249 0001681D 00001818000000      <1>
250 00016824 000000001800000000F- <1>     db  000h, 000h, 000h, 000h, 018h, 000h, 000h, 000h, 00fh, 00ch, 00ch, 00ch, 0ech, 06ch, 03ch, 01ch
250 0001682D 0C0C0CEC6C3C1C      <1>
```

```
251 00016834 786C6C6C6C00000070- <1>     db  078h, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 070h, 018h, 030h, 060h, 078h, 000h, 000h, 000h
251 0001683D 18306078000000      <1>
252 00016844 00003C3C3C3C000000- <1>     db  000h, 000h, 03ch, 03ch, 03ch, 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
252 0001684D 00000000000000      <1>
253                              <1> vgafont14:
254 00016854 00000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
254 0001685D 00000000000000      <1>
255 00016864 7E81A58181BD99817E- <1>     db  07eh, 081h, 0a5h, 081h, 081h, 0bdh, 099h, 081h, 07eh, 000h, 000h, 000h, 000h, 000h, 07eh, 0ffh
255 0001686D 00000000007EFF      <1>
256 00016874 DBFFFFC3E7FF7E0000- <1>     db  0dbh, 0ffh, 0ffh, 0c3h, 0e7h, 0ffh, 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 06ch, 0feh, 0feh
256 0001687D 000000006CFEFE      <1>
257 00016884 FEFE7C381000000000- <1>     db  0feh, 0feh, 07ch, 038h, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 07ch, 0feh, 07ch
257 0001688D 000010387CFE7C      <1>
258 00016894 3810000000000000018- <1>     db  038h, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 03ch, 0e7h, 0e7h, 0e7h, 018h, 018h
258 0001689D 3C3CE7E7E71818      <1>
259 000168A4 3C0000000000183C7E- <1>     db  03ch, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 07eh, 0ffh, 0ffh, 07eh, 018h, 018h, 03ch, 000h
259 000168AD FFFF7E18183C00      <1>
260 000168B4 00000000000000183C- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 03ch, 018h, 000h, 000h, 000h, 000h, 000h
260 000168BD 3C180000000000      <1>
261 000168C4 FFFFFFFFFFE7C3C3E7- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0e7h, 0c3h, 0c3h, 0e7h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h
261 000168CD FFFFFFFFFF0000      <1>
262 000168D4 00003C664242663C00- <1>     db  000h, 000h, 03ch, 066h, 042h, 042h, 066h, 03ch, 000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh
262 000168DD 000000FFFFFFFF      <1>
263 000168E4 C399BDBD99C3FFFFFF- <1>     db  0c3h, 099h, 0bdh, 0bdh, 099h, 0c3h, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 01eh, 00eh, 01ah, 032h
263 000168ED FF00001E0E1A32      <1>
264 000168F4 78CCCCCC7800000000- <1>     db  078h, 0cch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 066h, 066h, 03ch, 018h
264 000168FD 003C6666663C18      <1>
265 00016904 7E1818000000000003F- <1>     db  07eh, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 03fh, 033h, 03fh, 030h, 030h, 030h, 070h, 0f0h
265 0001690D 333F30303070F0      <1>
266 00016914 E000000000007F637F- <1>     db  0e0h, 000h, 000h, 000h, 000h, 000h, 07fh, 063h, 07fh, 063h, 063h, 063h, 067h, 0e7h, 0e6h, 0c0h
266 0001691D 63636367E7E6C0      <1>
267 00016924 000000001818DB3CE7- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 0dbh, 03ch, 0e7h, 03ch, 0dbh, 018h, 018h, 000h, 000h, 000h
267 0001692D 3CDB1818000000      <1>
268 00016934 000080C0E0F8FEF8E0- <1>     db  000h, 000h, 080h, 0c0h, 0e0h, 0f8h, 0feh, 0f8h, 0e0h, 0c0h, 080h, 000h, 000h, 000h, 000h, 000h
268 0001693D C0800000000000      <1>
269 00016944 02060E3EFE3E0E0602- <1>     db  002h, 006h, 00eh, 03eh, 0feh, 03eh, 00eh, 006h, 002h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch
269 0001694D 0000000000183C      <1>
270 00016954 7E1818187E3C180000- <1>     db  07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h
270 0001695D 00000066666666      <1>
271 00016964 6666006666000000000- <1>     db  066h, 066h, 000h, 066h, 066h, 000h, 000h, 000h, 000h, 000h, 07fh, 0dbh, 0dbh, 0dbh, 07bh, 01bh
271 0001696D 007FDBDBDB7B1B      <1>
272 00016974 1B1B1B000000007CC6- <1>     db  01bh, 01bh, 01bh, 000h, 000h, 000h, 000h, 07ch, 0c6h, 060h, 038h, 06ch, 0c6h, 0c6h, 06ch, 038h
272 0001697D 60386CC6C66C38      <1>
273 00016984 0CC67C000000000000- <1>     db  00ch, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 0feh, 0feh, 000h
273 0001698D 000000FEFEFE00      <1>
274 00016994 00000000183C7E1818- <1>     db  000h, 000h, 000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 07eh, 000h, 000h
274 0001699D 187E3C187E0000      <1>
275 000169A4 0000183C7E18181818- <1>     db  000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h
275 000169AD 18180000000000      <1>
276 000169B4 1818181818187E3C18- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
276 000169BD 00000000000000      <1>
277 000169C4 180CFE0C1800000000- <1>     db  018h, 00ch, 0feh, 00ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 030h, 060h
277 000169CD 00000000003060      <1>
278 000169D4 FE60300000000000000- <1>     db  0feh, 060h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0c0h, 0c0h, 0c0h
278 000169DD 00000000C0C0C0      <1>
279 000169E4 FE00000000000000000- <1>     db  0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 028h, 06ch, 0feh, 06ch, 028h, 000h
279 000169ED 00286CFE6C2800      <1>
280 000169F4 000000000000001038- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 038h, 07ch, 07ch, 0feh, 0feh, 000h, 000h
280 000169FD 387C7CFEFE0000      <1>
281 00016A04 0000000000FEFE7C7C- <1>     db  000h, 000h, 000h, 000h, 000h, 0feh, 0feh, 07ch, 07ch, 038h, 038h, 010h, 000h, 000h, 000h, 000h
281 00016A0D 38381000000000      <1>
282 00016A14 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
282 00016A1D 00000000000000      <1>
283 00016A24 183C3C3C1818001818- <1>     db  018h, 03ch, 03ch, 03ch, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 066h, 066h, 066h
283 00016A2D 00000000666666      <1>
284 00016A34 2400000000000000000- <1>     db  024h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 06ch, 06ch, 0feh, 06ch
284 00016A3D 0000006C6CFE6C      <1>
285 00016A44 6C6CFE6C6C00000018- <1>     db  06ch, 06ch, 0feh, 06ch, 06ch, 000h, 000h, 000h, 018h, 018h, 07ch, 0c6h, 0c2h, 0c0h, 07ch, 006h
285 00016A4D 187CC6C2C07C06      <1>
286 00016A54 86C67C1818000000000- <1>     db  086h, 0c6h, 07ch, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 0c2h, 0c6h, 00ch, 018h, 030h, 066h
286 00016A5D 00C2C60C183066      <1>
287 00016A64 C600000000000386C6C- <1>     db  0c6h, 000h, 000h, 000h, 000h, 000h, 038h, 06ch, 06ch, 038h, 076h, 0dch, 0cch, 0cch, 076h, 000h
287 00016A6D 3876DCCCCC7600      <1>
288 00016A74 000000303030600000- <1>     db  000h, 000h, 000h, 030h, 030h, 030h, 060h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
288 00016A7D 00000000000000      <1>
289 00016A84 00000C183030303030- <1>     db  000h, 000h, 00ch, 018h, 030h, 030h, 030h, 030h, 030h, 018h, 00ch, 000h, 000h, 000h, 000h, 000h
289 00016A8D 180C0000000000      <1>
290 00016A94 30180C0C0C0C0C1830- <1>     db  030h, 018h, 00ch, 00ch, 00ch, 00ch, 00ch, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
290 00016A9D 00000000000000      <1>
291 00016AA4 663CFF3C6600000000- <1>     db  066h, 03ch, 0ffh, 03ch, 066h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h
291 00016AAD 00000000001818      <1>
292 00016AB4 7E1818000000000000- <1>     db  07eh, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
292 00016ABD 00000000000000      <1>
293 00016AC4 1818183000000000000- <1>     db  018h, 018h, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 000h, 000h, 000h
293 00016ACD 000000FE000000      <1>
294 00016AD4 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h
294 00016ADD 00000000181800      <1>
295 00016AE4 0000000002060C1830- <1>     db  000h, 000h, 000h, 000h, 002h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 080h, 000h, 000h, 000h, 000h
295 00016AED 60C08000000000      <1>
296 00016AF4 00007CC6CEDEF6E6C6- <1>     db  000h, 000h, 07ch, 0c6h, 0ceh, 0deh, 0f6h, 0e6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
296 00016AFD C67C0000000000      <1>
297 00016B04 183878181818187E- <1>     db  018h, 038h, 078h, 018h, 018h, 018h, 018h, 018h, 07eh, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h
297 00016B0D 00000000007CC6      <1>
298 00016B14 060C183060C6FE0000- <1>     db  006h, 00ch, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 006h, 006h
298 00016B1D 0000007CC60606      <1>
299 00016B24 3C0606C67C00000000- <1>     db  03ch, 006h, 006h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 00ch, 01ch, 03ch, 06ch, 0cch, 0feh
299 00016B2D 000C1C3C6CCCFE      <1>
300 00016B34 0C0C1E0000000000FE- <1>     db  00ch, 00ch, 01eh, 000h, 000h, 000h, 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 0fch, 006h, 006h, 0c6h
300 00016B3D C0C0C0FC0606C6      <1>
301 00016B44 7C00000000003860C0- <1>     db  07ch, 000h, 000h, 000h, 000h, 000h, 038h, 060h, 0c0h, 0c0h, 0fch, 0c6h, 0c6h, 0c6h, 07ch, 000h
301 00016B4D C0FCC6C6C67C00      <1>
302 00016B54 00000000FEC6060C18- <1>     db  000h, 000h, 000h, 000h, 0feh, 0c6h, 006h, 00ch, 018h, 030h, 030h, 030h, 030h, 000h, 000h, 000h
302 00016B5D 30303030000000      <1>
303 00016B64 00007CC6C6C67CC6C6- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 07ch, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
303 00016B6D C67C0000000000      <1>
304 00016B74 7CC6C6C67E06060C78- <1>     db  07ch, 0c6h, 0c6h, 0c6h, 07eh, 006h, 006h, 00ch, 078h, 000h, 000h, 000h, 000h, 000h, 000h, 018h
304 00016B7D 00000000000018      <1>
305 00016B84 180000001818000000- <1>     db  018h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h
305 00016B8D 00000000181800      <1>
306 00016B94 000018183000000000- <1>     db  000h, 000h, 018h, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 006h, 00ch, 018h, 030h, 060h, 030h
306 00016B9D 00060C18306030      <1>
307 00016BA4 180C06000000000000- <1>     db  018h, 00ch, 006h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 000h, 000h, 07eh, 000h
307 00016BAD 00007E00007E00      <1>
308 00016BB4 000000000000603018- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 00ch, 006h, 00ch, 018h, 030h, 060h, 000h
308 00016BBD 0C060C18306000      <1>
309 00016BC4 000000007CC6C60C18- <1>     db  000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 00ch, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h
309 00016BCD 18001818000000      <1>
310 00016BD4 00007CC6C6DEDEDEDC- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 0deh, 0deh, 0deh, 0dch, 0c0h, 07ch, 000h, 000h, 000h, 000h, 000h
```

```
310 00016BDD C07C0000000000      <1>
311 00016BE4 10386CC6C6FEC6C6C6- <1>     db  010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h, 0fch, 066h
311 00016BED 0000000000FC66      <1>
312 00016BF4 66667C666666FC0000- <1>     db  066h, 066h, 07ch, 066h, 066h, 066h, 0fch, 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 0c2h, 0c0h
312 00016BFD 0000003C66C2C0      <1>
313 00016C04 C0C0C2663C00000000- <1>     db  0c0h, 0c0h, 0c2h, 066h, 03ch, 000h, 000h, 000h, 000h, 000h, 0f8h, 06ch, 066h, 066h, 066h, 066h
313 00016C0D 00F86C66666666      <1>
314 00016C14 666CF800000000000FE- <1>     db  066h, 06ch, 0f8h, 000h, 000h, 000h, 000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 062h, 066h
314 00016C1D 66626878686266      <1>
315 00016C24 FE0000000000FE6662- <1>     db  0feh, 000h, 000h, 000h, 000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 060h, 060h, 0f0h, 000h
315 00016C2D 6878686060F000      <1>
316 00016C34 000000003C66C2C0C0- <1>     db  000h, 000h, 000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0deh, 0c6h, 066h, 03ah, 000h, 000h, 000h
316 00016C3D DEC6663A000000      <1>
317 00016C44 0000C6C6C6C6FEC6C6- <1>     db  000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h
317 00016C4D C6C600000000000     <1>
318 00016C54 3C1818181818183C- <1>     db  03ch, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 01eh, 00ch
318 00016C5D 00000000001E0C      <1>
319 00016C64 0C0C0C0CCCCC780000- <1>     db  00ch, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h, 000h, 0e6h, 066h, 06ch, 06ch
319 00016C6D 000000E6666C6C      <1>
320 00016C74 786C6C66E600000000- <1>     db  078h, 06ch, 06ch, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 0f0h, 060h, 060h, 060h, 060h, 060h
320 00016C7D 00F06060606060      <1>
321 00016C84 6266FE0000000000C6- <1>     db  062h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h, 0c6h, 0eeh, 0feh, 0feh, 0d6h, 0c6h, 0c6h, 0c6h
321 00016C8D EEFEFED6C6C6C6      <1>
322 00016C94 C600000000006E6F6- <1>     db  0c6h, 000h, 000h, 000h, 000h, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h, 0c6h, 0c6h, 000h
322 00016C9D FEDECEC6C6C600      <1>
323 00016CA4 00000000386CC6C6C6- <1>     db  000h, 000h, 000h, 000h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h
323 00016CAD C6C66C38000000      <1>
324 00016CB4 0000FC6666667C6060- <1>     db  000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h, 000h
324 00016CBD 60F00000000000      <1>
325 00016CC4 7CC6C6C6C6D6DE7C0C- <1>     db  07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0d6h, 0deh, 07ch, 00ch, 00eh, 000h, 000h, 000h, 000h, 0fch, 066h
325 00016CCD 0E00000000FC66      <1>
326 00016CD4 66667C6C6666E60000- <1>     db  066h, 066h, 07ch, 06ch, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 060h
326 00016CDD 0000007CC6C660      <1>
327 00016CE4 380CC6C67C00000000- <1>     db  038h, 00ch, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 07eh, 07eh, 05ah, 018h, 018h, 018h
327 00016CED 007E7E5A181818      <1>
328 00016CF4 18183C0000000000C6- <1>     db  018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h
328 00016CFD C6C6C6C6C6C6C6      <1>
329 00016D04 7C0000000000C6C6C6- <1>     db  07ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 010h, 000h
329 00016D0D C6C6C66C381000      <1>
330 00016D14 00000000C6C6C6C6D6- <1>     db  000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0d6h, 0d6h, 0feh, 07ch, 06ch, 000h, 000h, 000h
330 00016D1D D6FE7C6C000000      <1>
331 00016D24 0000C6C66C3838386C- <1>     db  000h, 000h, 0c6h, 0c6h, 06ch, 038h, 038h, 038h, 06ch, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h
331 00016D2D C6C600000000000     <1>
332 00016D34 666666663C1818183C- <1>     db  066h, 066h, 066h, 066h, 03ch, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 0feh, 0c6h
332 00016D3D 0000000000FEC6      <1>
333 00016D44 8C183060C2C6FE0000- <1>     db  08ch, 018h, 030h, 060h, 0c2h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h, 03ch, 030h, 030h, 030h
333 00016D4D 0000003C303030      <1>
334 00016D54 303030303C00000000- <1>     db  030h, 030h, 030h, 030h, 03ch, 000h, 000h, 000h, 000h, 000h, 080h, 0c0h, 0e0h, 070h, 038h, 01ch
334 00016D5D 0080C0E070381C      <1>
335 00016D64 0E060200000000003C- <1>     db  00eh, 006h, 002h, 000h, 000h, 000h, 000h, 000h, 03ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch
335 00016D6D 0C0C0C0C0C0C0C      <1>
336 00016D74 3C00000010386CC600- <1>     db  03ch, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
336 00016D7D 00000000000000      <1>
337 00016D84 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h
337 00016D8D 0000000000FF00      <1>
338 00016D94 303018000000000000- <1>     db  030h, 030h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
338 00016D9D 00000000000000      <1>
339 00016DA4 000000780C7CCCCC76- <1>     db  000h, 000h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 0e0h, 060h
339 00016DAD 0000000000E060      <1>
340 00016DB4 60786C6666667C0000- <1>     db  060h, 078h, 06ch, 066h, 066h, 066h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch
340 00016DBD 0000000000007C      <1>
341 00016DC4 C6C0C0C67C00000000- <1>     db  0c6h, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 01ch, 00ch, 00ch, 03ch, 06ch, 0cch
341 00016DCD 001C0C0C3C6CCC      <1>
342 00016DD4 CCCC760000000000000- <1>     db  0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h
342 00016DDD 00007CC6FEC0C6      <1>
343 00016DE4 7C0000000000386C64- <1>     db  07ch, 000h, 000h, 000h, 000h, 000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h, 060h, 0f0h, 000h
343 00016DED 60F0606060F000      <1>
344 00016DF4 0000000000000076CC- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 07ch, 00ch, 0cch, 078h, 000h
344 00016DFD CCCC7C0CCC7800      <1>
345 00016E04 0000E060606C766666- <1>     db  000h, 000h, 0e0h, 060h, 060h, 06ch, 076h, 066h, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h
345 00016E0D 66E60000000000      <1>
346 00016E14 18180038181818183C- <1>     db  018h, 018h, 000h, 038h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 006h, 006h
346 00016E1D 00000000000606      <1>
347 00016E24 000E0606060666663C- <1>     db  000h, 00eh, 006h, 006h, 006h, 006h, 066h, 066h, 03ch, 000h, 000h, 000h, 0e0h, 060h, 060h, 066h
347 00016E2D 000000E0606066      <1>
348 00016E34 6C786C66E600000000- <1>     db  06ch, 078h, 06ch, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 038h, 018h, 018h, 018h, 018h, 018h
348 00016E3D 00381818181818      <1>
349 00016E44 18183C0000000000000- <1>     db  018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ech, 0feh, 0d6h, 0d6h, 0d6h
349 00016E4D 0000ECFED6D6D6      <1>
350 00016E54 C600000000000000000- <1>     db  0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 000h
350 00016E5D DC666666666600      <1>
351 00016E64 000000000000007CC6- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h
351 00016E6D C6C6C67C000000      <1>
352 00016E74 0000000000DC666666- <1>     db  000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 07ch, 060h, 060h, 0f0h, 000h, 000h, 000h
352 00016E7D 7C6060F0000000      <1>
353 00016E84 00000076CCCCCC7C0C- <1>     db  000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 07ch, 00ch, 00ch, 01eh, 000h, 000h, 000h, 000h, 000h
353 00016E8D 0C1E0000000000      <1>
354 00016E94 00DC76666060F00000- <1>     db  000h, 0dch, 076h, 066h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch
354 00016E9D 0000000000007C      <1>
355 00016EA4 C6701CC67C00000000- <1>     db  0c6h, 070h, 01ch, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 010h, 030h, 030h, 0fch, 030h, 030h
355 00016EAD 00103030FC3030      <1>
356 00016EB4 30361C000000000000- <1>     db  030h, 036h, 01ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch
356 00016EBD 0000CCCCCCCCCC      <1>
357 00016EC4 760000000000000000- <1>     db  076h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h, 03ch, 018h, 000h
357 00016ECD 666666663C1800      <1>
358 00016ED4 00000000000000C6C6- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0d6h, 0d6h, 0feh, 06ch, 000h, 000h, 000h
358 00016EDD D6D6FE6C000000      <1>
359 00016EE4 0000000000C66C3838- <1>     db  000h, 000h, 000h, 000h, 000h, 0c6h, 06ch, 038h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h
359 00016EED 6CC600000000000     <1>
360 00016EF4 000000C6C6C6C67E06- <1>     db  000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 0f8h, 000h, 000h, 000h, 000h, 000h
360 00016EFD 0CF80000000000      <1>
361 00016F04 00FECC183066FE0000- <1>     db  000h, 0feh, 0cch, 018h, 030h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h, 00eh, 018h, 018h, 018h
361 00016F0D 0000000E181818      <1>
362 00016F14 701818180E00000000- <1>     db  070h, 018h, 018h, 018h, 00eh, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 000h, 018h
362 00016F1D 00181818180018      <1>
363 00016F24 1818180000000000070- <1>     db  018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 070h, 018h, 018h, 018h, 00eh, 018h, 018h, 018h
363 00016F2D 1818180E181818      <1>
364 00016F34 70000000000076DC00- <1>     db  070h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
364 00016F3D 00000000000000      <1>
365 00016F44 000000000000010386C- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 000h, 000h, 000h, 000h
365 00016F4D C6C6FE00000000      <1>
366 00016F54 00003C66C2C0C0C266- <1>     db  000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0c2h, 066h, 03ch, 00ch, 006h, 07ch, 000h, 000h, 000h
366 00016F5D 3C0C067C000000      <1>
367 00016F64 CCCC00CCCCCCCCCC76- <1>     db  0cch, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 00ch, 018h, 030h
367 00016F6D 000000000C1830      <1>
368 00016F74 007CC6FEC0C67C0000- <1>     db  000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 000h, 078h
368 00016F7D 000010386C0078      <1>
369 00016F84 0C7CCCCC7600000000- <1>     db  00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 0cch, 0cch, 000h, 078h, 00ch, 07ch
```

```
369 00016F8D 00CCCC00780C7C      <1>
370 00016F94 CCCC76000000006030- <1>     db  0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 078h, 00ch, 07ch, 0cch, 0cch
370 00016F9D 1800780C7CCCCC      <1>
371 00016FA4 7600000000386C3800- <1>     db  076h, 000h, 000h, 000h, 000h, 038h, 06ch, 038h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h
371 00016FAD 780C7CCCCC7600      <1>
372 00016FB4 0000000000003C6660- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 060h, 066h, 03ch, 00ch, 006h, 03ch, 000h, 000h
372 00016FBD 663C0C063C0000      <1>
373 00016FC4 0010386C007CC6FEC0- <1>     db  000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
373 00016FCD C67C0000000000      <1>
374 00016FD4 CCCC007CC6FEC0C67C- <1>     db  0cch, 0cch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h
374 00016FDD 00000000603018      <1>
375 00016FE4 007CC6FEC0C67C0000- <1>     db  000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 000h, 038h
375 00016FED 00000066660038      <1>
376 00016FF4 181818183C00000000- <1>     db  018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 018h, 03ch, 066h, 000h, 038h, 018h, 018h
376 00016FFD 183C6600381818      <1>
377 00017004 18183C000000006030- <1>     db  018h, 018h, 03ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 038h, 018h, 018h, 018h, 018h
377 0001700D 18003818181818      <1>
378 00017014 3C00000000C6C61038- <1>     db  03ch, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 000h
378 0001701D 6CC6C6FEC6C600      <1>
379 00017024 0000386C3800386CC6- <1>     db  000h, 000h, 038h, 06ch, 038h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 000h, 000h, 000h
379 0001702D C6FEC6C6000000      <1>
380 00017034 18306000FE66607C60- <1>     db  018h, 030h, 060h, 000h, 0feh, 066h, 060h, 07ch, 060h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h
380 0001703D 66FE0000000000      <1>
381 00017044 0000CC76367ED8D86E- <1>     db  000h, 000h, 0cch, 076h, 036h, 07eh, 0d8h, 0d8h, 06eh, 000h, 000h, 000h, 000h, 000h, 03eh, 06ch
381 0001704D 000000000003E6C     <1>
382 00017054 CCCCFECCCCCCCE0000- <1>     db  0cch, 0cch, 0feh, 0cch, 0cch, 0cch, 0ceh, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 000h, 07ch
382 0001705D 000010386C007C      <1>
383 00017064 C6C6C6C67C00000000- <1>     db  0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 000h, 07ch, 0c6h, 0c6h
383 0001706D 00C6C6007CC6C6      <1>
384 00017074 C6C67C000000006030- <1>     db  0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h
384 0001707D 18007CC6C6C6C6      <1>
385 00017084 7C0000000003078CC00- <1>     db  07ch, 000h, 000h, 000h, 000h, 030h, 078h, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h
385 0001708D CCCCCCCCCC7600      <1>
386 00017094 00000060301800CCCC- <1>     db  000h, 000h, 000h, 060h, 030h, 018h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h
386 0001709D CCCCCC76000000      <1>
387 000170A4 0000C6C600C6C6C6C6- <1>     db  000h, 000h, 0c6h, 0c6h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 078h, 000h, 000h, 0c6h
387 000170AD 7E060C780000C6      <1>
388 000170B4 C6386CC6C6C6C66C38- <1>     db  0c6h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 000h
388 000170BD 00000000C6C600      <1>
389 000170C4 C6C6C6C6C6C67C0000- <1>     db  0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 018h, 018h, 03ch, 066h, 060h
389 000170CD 000018183C6660      <1>
390 000170D4 60663C181800000000- <1>     db  060h, 066h, 03ch, 018h, 018h, 000h, 000h, 000h, 000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h
390 000170DD 386C6460F06060      <1>
391 000170E4 60E6FC000000000066- <1>     db  060h, 0e6h, 0fch, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 03ch, 018h, 07eh, 018h, 07eh, 018h
391 000170ED 663C187E187E18      <1>
392 000170F4 1800000000F8CCCCF8- <1>     db  018h, 000h, 000h, 000h, 000h, 0f8h, 0cch, 0cch, 0f8h, 0c4h, 0cch, 0deh, 0cch, 0cch, 0c6h, 000h
392 000170FD C4CCDECCCCC600      <1>
393 00017104 0000000E1B1818187E- <1>     db  000h, 000h, 000h, 00eh, 01bh, 018h, 018h, 018h, 07eh, 018h, 018h, 018h, 018h, 0d8h, 070h, 000h
393 0001710D 18181818D87000      <1>
394 00017114 0018306000780C7CCC- <1>     db  000h, 018h, 030h, 060h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 00ch
394 0001711D CC76000000000C      <1>
395 00017124 18300038181818183C- <1>     db  018h, 030h, 000h, 038h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 018h, 030h, 060h
395 0001712D 00000000183060      <1>
396 00017134 007CC6C6C6C67C0000- <1>     db  000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 018h, 030h, 060h, 000h, 0cch
396 0001713D 000018306000CC      <1>
397 00017144 CCCCCCCC7600000000- <1>     db  0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 0dch, 066h, 066h
397 0001714D 0076DC00DC6666      <1>
398 00017154 66666600000076DC00- <1>     db  066h, 066h, 066h, 000h, 000h, 000h, 076h, 0dch, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h
398 0001715D C6E6F6FEDECEC6      <1>
399 00017164 C6000000003C6C6C3E- <1>     db  0c6h, 000h, 000h, 000h, 000h, 03ch, 06ch, 06ch, 03eh, 000h, 07eh, 000h, 000h, 000h, 000h, 000h
399 0001716D 007E0000000000      <1>
400 00017174 000000386C6C38007C- <1>     db  000h, 000h, 000h, 038h, 06ch, 06ch, 038h, 000h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h
400 0001717D 00000000000000      <1>
401 00017184 0000303000303060C6- <1>     db  000h, 000h, 030h, 030h, 000h, 030h, 030h, 060h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
401 0001718D C67C0000000000      <1>
402 00017194 00000000FEC0C0C000- <1>     db  000h, 000h, 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
402 0001719D 00000000000000      <1>
403 000171A4 0000FE060606000000- <1>     db  000h, 000h, 0feh, 006h, 006h, 006h, 000h, 000h, 000h, 000h, 000h, 0c0h, 0c0h, 0c6h, 0cch, 0d8h
403 000171AD 0000C0C0C6CCD8      <1>
404 000171B4 3060DC860C183E0000- <1>     db  030h, 060h, 0dch, 086h, 00ch, 018h, 03eh, 000h, 000h, 0c0h, 0c0h, 0c6h, 0cch, 0d8h, 030h, 066h
404 000171BD C0C0C6CCD83066      <1>
405 000171C4 CE9E3E060600000018- <1>     db  0ceh, 09eh, 03eh, 006h, 006h, 000h, 000h, 000h, 018h, 018h, 000h, 018h, 018h, 03ch, 03ch, 03ch
405 000171CD 180018183C3C3C      <1>
406 000171D4 180000000000000036- <1>     db  018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 036h, 06ch, 0d8h, 06ch, 036h, 000h, 000h, 000h
406 000171DD 6CD86C36000000      <1>
407 000171E4 000000000000D86C36- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 0d8h, 06ch, 036h, 06ch, 0d8h, 000h, 000h, 000h, 000h, 000h
407 000171ED 6CD800000000000     <1>
408 000171F4 114414411441144411- <1>     db  011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 055h, 0aah
408 000171FD 441144114455AA      <1>
409 00017204 55AA55AA55AA55AA55- <1>     db  055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 0ddh, 077h, 0ddh, 077h
409 0001720D AA55AADD77DD77      <1>
410 00017214 DD77DD77DD77DD77DD- <1>     db  0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 018h, 018h, 018h, 018h, 018h, 018h
410 0001721D 77181818181818      <1>
411 00017224 181818181818181818- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h
411 0001722D 181818181818F8      <1>
412 00017234 181818181818181818- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h
412 0001723D 1818F818F81818      <1>
413 00017244 181818183636363636- <1>     db  018h, 018h, 018h, 018h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 036h
413 0001724D 3636F636363636      <1>
414 00017254 363600000000000000- <1>     db  036h, 036h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h, 036h, 036h, 036h
414 0001725D FE363636363636      <1>
415 00017264 0000000000F818F818- <1>     db  000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 036h, 036h
415 0001726D 18181818183636      <1>
416 00017274 363636F606F6363636- <1>     db  036h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
416 0001727D 36363636363636      <1>
417 00017284 3636363636363636- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 000h, 0feh
417 0001728D 360000000000FE      <1>
418 00017294 06F6363636363636- <1>     db  006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0f6h, 006h, 0feh
418 0001729D 36363636F606FE      <1>
419 000172A4 0000000000000363636- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h
419 000172AD 36363636FE0000      <1>
420 000172B4 0000000001818181818- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 000h, 000h, 000h, 000h
420 000172BD F818F800000000      <1>
421 000172C4 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h
421 000172CD F8181818181818      <1>
422 000172D4 1818181818181F00- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h
422 000172DD 00000000001818      <1>
423 000172E4 1818181818FF000000- <1>     db  018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
423 000172ED 00000000000000      <1>
424 000172F4 000000FF1818181818- <1>     db  000h, 000h, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
424 000172FD 18181818181818      <1>
425 00017304 181F18181818181800- <1>     db  018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh
425 0001730D 000000000000FF      <1>
426 00017314 0000000000000181818- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h
426 0001731D 18181818FF1818      <1>
427 00017324 181818181818181818- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh, 018h, 018h, 018h, 018h
427 0001732D 1F181F18181818      <1>
428 00017334 1818363636363636- <1>     db  018h, 018h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 037h, 036h, 036h, 036h, 036h, 036h, 036h
```

```
428 0001733D 37363636363636      <1>
429 00017344 3636363636373037F00- <1>     db  036h, 036h, 036h, 036h, 036h, 037h, 030h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
429 0001734D 00000000000000      <1>
430 00017354 0000003F3037363636- <1>     db  000h, 000h, 000h, 03fh, 030h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
430 0001735D 36363636363636      <1>
431 00017364 36F700FF0000000000- <1>     db  036h, 0f7h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh
431 0001736D 000000000000FF      <1>
432 00017374 00F736363636363636- <1>     db  000h, 0f7h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 037h, 030h, 037h
432 0001737D 36363636373037      <1>
433 00017384 3636363636000000- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 000h, 000h
433 0001738D 0000FF00FF0000      <1>
434 00017394 000000003636363636- <1>     db  000h, 000h, 000h, 000h, 036h, 036h, 036h, 036h, 036h, 0f7h, 000h, 0f7h, 036h, 036h, 036h, 036h
434 0001739D F700F736363636      <1>
435 000173A4 36361818181818FF00- <1>     db  036h, 036h, 018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h
435 000173AD FF000000000000      <1>
436 000173B4 3636363636636FF00- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
436 000173BD 00000000000000      <1>
437 000173C4 000000FF00FF181818- <1>     db  000h, 000h, 000h, 0ffh, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h
437 000173CD 18181800000000      <1>
438 000173D4 000000FF3636363636- <1>     db  000h, 000h, 000h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
438 000173DD 36363636363636      <1>
439 000173E4 363F00000000000018- <1>     db  036h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh
439 000173ED 181818181F181F      <1>
440 000173F4 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h
440 000173FD 00001F181F1818      <1>
441 00017404 181818180000000000- <1>     db  018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 03fh, 036h, 036h, 036h, 036h
441 0001740D 00003F36363636      <1>
442 00017414 3636363636363636- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h
442 0001741D FF363636363636      <1>
443 00017424 1818181818FF18FF18- <1>     db  018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
443 0001742D 18181818181818      <1>
444 00017434 1818181818F8000000- <1>     db  018h, 018h, 018h, 018h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
444 0001743D 00000000000000      <1>
445 00017444 0000001F1818181818- <1>     db  000h, 000h, 000h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
445 0001744D 18FFFFFFFFFFFF      <1>
446 00017454 FFFFFFFFFFFFFFFF00- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh
446 0001745D 000000000000FF      <1>
447 00017464 FFFFFFFFFFFF0F0F0- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h
447 0001746D F0F0F0F0F0F0F0      <1>
448 00017474 F0F0F0F00F0F0F0F0F- <1>     db  0f0h, 0f0h, 0f0h, 0f0h, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh
448 0001747D 0F0F0F0F0F0F0F      <1>
449 00017484 0F0FFFFFFFFFFFFFFF- <1>     db  00fh, 00fh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h
449 0001748D 00000000000000      <1>
450 00017494 000000000076DCD8D8- <1>     db  000h, 000h, 000h, 000h, 000h, 076h, 0dch, 0d8h, 0d8h, 0dch, 076h, 000h, 000h, 000h, 000h, 000h
450 0001749D DC760000000000      <1>
451 000174A4 00007CC6FCC6C6FCC0- <1>     db  000h, 000h, 07ch, 0c6h, 0fch, 0c6h, 0c6h, 0fch, 0c0h, 0c0h, 040h, 000h, 000h, 000h, 0feh, 0c6h
451 000174AD C040000000FEC6      <1>
452 000174B4 C6C0C0C0C0C0C00000- <1>     db  0c6h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 06ch
452 000174BD 0000000000FE6C      <1>
453 000174C4 6C6C6C6C6C00000000- <1>     db  06ch, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h, 000h, 0feh, 0c6h, 060h, 030h, 018h, 030h
453 000174CD 00FEC660301830      <1>
454 000174D4 60C6FE000000000000- <1>     db  060h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 0d8h, 0d8h, 0d8h, 0d8h
454 000174DD 00007ED8D8D8D8      <1>
455 000174E4 7000000000000000066- <1>     db  070h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h, 07ch, 060h, 060h, 0c0h
455 000174ED 6666667C6060C0      <1>
456 000174F4 00000000000076DC18- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h
456 000174FD 18181818000000      <1>
457 00017504 00007E183C6666663C- <1>     db  000h, 000h, 07eh, 018h, 03ch, 066h, 066h, 066h, 03ch, 018h, 07eh, 000h, 000h, 000h, 000h, 000h
457 0001750D 187E0000000000      <1>
458 00017514 386CC6C6FEC6C66C38- <1>     db  038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 038h, 06ch
458 0001751D 0000000000386C      <1>
459 00017524 C6C6C66C6C6CEE0000- <1>     db  0c6h, 0c6h, 0c6h, 06ch, 06ch, 06ch, 0eeh, 000h, 000h, 000h, 000h, 000h, 01eh, 030h, 018h, 00ch
459 0001752D 0000001E30180C      <1>
460 00017534 3E6666663C00000000- <1>     db  03eh, 066h, 066h, 066h, 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 0dbh, 0dbh
460 0001753D 000000007EDBDB      <1>
461 00017544 7E00000000000000003- <1>     db  07eh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 003h, 006h, 07eh, 0dbh, 0dbh, 0f3h, 07eh, 060h
461 0001754D 067EDBDBF37E60      <1>
462 00017554 C000000000001C3060- <1>     db  0c0h, 000h, 000h, 000h, 000h, 000h, 01ch, 030h, 060h, 060h, 07ch, 060h, 060h, 030h, 01ch, 000h
462 0001755D 607C6060301C00      <1>
463 00017564 000000000007CC6C6C6- <1>     db  000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h
463 0001756D C6C6C6C6000000      <1>
464 00017574 000000FE0000FE0000- <1>     db  000h, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h
464 0001757D FE000000000000      <1>
465 00017584 0018187E18180000FF- <1>     db  000h, 018h, 018h, 07eh, 018h, 018h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 030h, 018h
465 0001758D 00000000003018      <1>
466 00017594 0C060C1830007E0000- <1>     db  00ch, 006h, 00ch, 018h, 030h, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 00ch, 018h, 030h, 060h
466 0001759D 0000000C183060      <1>
467 000175A4 30180C007E00000000- <1>     db  030h, 018h, 00ch, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 00eh, 01bh, 01bh, 018h, 018h, 018h
467 000175AD 000E1B1B181818      <1>
468 000175B4 181818181818181818- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0d8h, 0d8h
468 000175BD 1818181818D8D8      <1>
469 000175C4 7000000000000001818- <1>     db  070h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 07eh, 000h, 018h, 018h, 000h, 000h
469 000175CD 007E0018180000      <1>
470 000175D4 00000000000076DC00- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h
470 000175DD 76DC0000000000      <1>
471 000175E4 00386C6C3800000000- <1>     db  000h, 038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
471 000175ED 00000000000000      <1>
472 000175F4 0000000018180000000- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
472 000175FD 00000000000000      <1>
473 00017604 0000001800000000000- <1>     db  000h, 000h, 000h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 00fh, 00ch, 00ch, 00ch, 00ch
473 0001760D 00000F0C0C0C0C      <1>
474 00017614 0CEC6C3C1C00000000- <1>     db  00ch, 0ech, 06ch, 03ch, 01ch, 000h, 000h, 000h, 000h, 0d8h, 06ch, 06ch, 06ch, 06ch, 06ch, 000h
474 0001761D D86C6C6C6C6C00      <1>
475 00017624 0000000000000070D8- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 070h, 0d8h, 030h, 060h, 0c8h, 0f8h, 000h, 000h, 000h
475 0001762D 3060C8F8000000      <1>
476 00017634 000000000000000007C- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 000h, 000h
476 0001763D 7C7C7C7C7C0000      <1>
477 00017644 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
477 0001764D 00000000000000      <1>
478                              <1> vgafont16:
479 00017654 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
479 0001765D 00000000000000      <1>
480 00017664 00007E81A58181BD99- <1>     db  000h, 000h, 07eh, 081h, 0a5h, 081h, 081h, 0bdh, 099h, 081h, 081h, 07eh, 000h, 000h, 000h, 000h
480 0001766D 81817E00000000      <1>
481 00017674 00007EFFDBFFFFC3E7- <1>     db  000h, 000h, 07eh, 0ffh, 0dbh, 0ffh, 0ffh, 0c3h, 0e7h, 0ffh, 0ffh, 07eh, 000h, 000h, 000h, 000h
481 0001767D FFFF7E00000000      <1>
482 00017684 000000006CFEFEFEFE- <1>     db  000h, 000h, 000h, 000h, 06ch, 0feh, 0feh, 0feh, 0feh, 07ch, 038h, 010h, 000h, 000h, 000h, 000h
482 0001768D 7C381000000000      <1>
483 00017694 0000000010387CFE7C- <1>     db  000h, 000h, 000h, 000h, 010h, 038h, 07ch, 0feh, 07ch, 038h, 010h, 000h, 000h, 000h, 000h, 000h
483 0001769D 38100000000000      <1>
484 000176A4 000000183C3CE7E7E7- <1>     db  000h, 000h, 000h, 018h, 03ch, 03ch, 0e7h, 0e7h, 0e7h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
484 000176AD 18183C00000000      <1>
485 000176B4 000000183C7EFFFF7E- <1>     db  000h, 000h, 000h, 018h, 03ch, 07eh, 0ffh, 0ffh, 07eh, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
485 000176BD 18183C00000000      <1>
486 000176C4 000000000000183C3C- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 03ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h
486 000176CD 18000000000000      <1>
487 000176D4 FFFFFFFFFFFFE7C3C3- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0e7h, 0c3h, 0c3h, 0e7h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
487 000176DD E7FFFFFFFFFFFF      <1>
```

```
488 000176E4 00000000003C664242- <1>     db  000h, 000h, 000h, 000h, 000h, 03ch, 066h, 042h, 042h, 066h, 03ch, 000h, 000h, 000h, 000h, 000h
488 000176ED 663C0000000000      <1>
489 000176F4 FFFFFFFFFFC399BDBD- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0c3h, 099h, 0bdh, 0bdh, 099h, 0c3h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
489 000176FD 99C3FFFFFFFFFF      <1>
490 00017704 00001E0E1A3278CCCC- <1>     db  000h, 000h, 01eh, 00eh, 01ah, 032h, 078h, 0cch, 0cch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h
490 0001770D CCCC7800000000      <1>
491 00017714 00003C666666663C18- <1>     db  000h, 000h, 03ch, 066h, 066h, 066h, 066h, 03ch, 018h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h
491 0001771D 7E181800000000      <1>
492 00017724 00003F333F30303030- <1>     db  000h, 000h, 03fh, 033h, 03fh, 030h, 030h, 030h, 030h, 070h, 0f0h, 0e0h, 000h, 000h, 000h, 000h
492 0001772D 70F0E000000000      <1>
493 00017734 00007F637F63636363- <1>     db  000h, 000h, 07fh, 063h, 07fh, 063h, 063h, 063h, 063h, 067h, 0e7h, 0e6h, 0c0h, 000h, 000h, 000h
493 0001773D 67E7E6C0000000      <1>
494 00017744 0000001818DB3CE73C- <1>     db  000h, 000h, 000h, 018h, 018h, 0dbh, 03ch, 0e7h, 03ch, 0dbh, 018h, 018h, 000h, 000h, 000h, 000h
494 0001774D DB181800000000      <1>
495 00017754 0080C0E0F0F8FEF8F0- <1>     db  000h, 080h, 0c0h, 0e0h, 0f0h, 0f8h, 0feh, 0f8h, 0f0h, 0e0h, 0c0h, 080h, 000h, 000h, 000h, 000h
495 0001775D E0C08000000000      <1>
496 00017764 0002060E1E3EFE3E1E- <1>     db  000h, 002h, 006h, 00eh, 01eh, 03eh, 0feh, 03eh, 01eh, 00eh, 006h, 002h, 000h, 000h, 000h, 000h
496 0001776D 0E060200000000      <1>
497 00017774 0000183C7E1818187E- <1>     db  000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h
497 0001777D 3C180000000000      <1>
498 00017784 000066666666666666- <1>     db  000h, 000h, 066h, 066h, 066h, 066h, 066h, 066h, 066h, 000h, 066h, 066h, 000h, 000h, 000h, 000h
498 0001778D 00666600000000      <1>
499 00017794 00007FDBDBDB7B1B1B- <1>     db  000h, 000h, 07fh, 0dbh, 0dbh, 0dbh, 07bh, 01bh, 01bh, 01bh, 01bh, 01bh, 000h, 000h, 000h, 000h
499 0001779D 1B1B1B00000000      <1>
500 000177A4 007CC660386CC6C66C- <1>     db  000h, 07ch, 0c6h, 060h, 038h, 06ch, 0c6h, 0c6h, 06ch, 038h, 00ch, 0c6h, 07ch, 000h, 000h, 000h
500 000177AD 380CC67C000000      <1>
501 000177B4 00000000000000FE- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 0feh, 0feh, 0feh, 000h, 000h, 000h, 000h
501 000177BD FEFEFE00000000      <1>
502 000177C4 0000183C7E1818187E- <1>     db  000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 07eh, 000h, 000h, 000h, 000h
502 000177CD 3C187E00000000      <1>
503 000177D4 0000183C7E18181818- <1>     db  000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h
503 000177DD 18181800000000      <1>
504 000177E4 0000181818181818- <1>     db  000h, 000h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h
504 000177ED 7E3C1800000000      <1>
505 000177F4 0000000000180CFE0C- <1>     db  000h, 000h, 000h, 000h, 000h, 018h, 00ch, 0feh, 00ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h
505 000177FD 18000000000000      <1>
506 00017804 00000000003060FE60- <1>     db  000h, 000h, 000h, 000h, 000h, 030h, 060h, 0feh, 060h, 030h, 000h, 000h, 000h, 000h, 000h, 000h
506 0001780D 30000000000000      <1>
507 00017814 000000000000C0C0C0- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 0c0h, 0c0h, 0c0h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h
507 0001781D FE000000000000      <1>
508 00017824 00000000002466FF66- <1>     db  000h, 000h, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h, 000h, 000h, 000h, 000h
508 0001782D 24000000000000      <1>
509 00017834 0000000001038387C7C- <1>     db  000h, 000h, 000h, 000h, 010h, 038h, 038h, 07ch, 07ch, 0feh, 0feh, 000h, 000h, 000h, 000h, 000h
509 0001783D FEFE0000000000      <1>
510 00017844 00000000FEFE7C7C38- <1>     db  000h, 000h, 000h, 000h, 0feh, 0feh, 07ch, 07ch, 038h, 038h, 010h, 000h, 000h, 000h, 000h, 000h
510 0001784D 38100000000000      <1>
511 00017854 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
511 0001785D 00000000000000      <1>
512 00017864 0000183C3C3C181818- <1>     db  000h, 000h, 018h, 03ch, 03ch, 03ch, 018h, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h, 000h
512 0001786D 00181800000000      <1>
513 00017874 006666662400000000- <1>     db  000h, 066h, 066h, 066h, 024h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
513 0001787D 00000000000000      <1>
514 00017884 0000006C6CFE6C6C6C- <1>     db  000h, 000h, 000h, 06ch, 06ch, 0feh, 06ch, 06ch, 06ch, 0feh, 06ch, 06ch, 000h, 000h, 000h, 000h
514 0001788D FE6C6C00000000      <1>
515 00017894 18187CC6C2C07C0606- <1>     db  018h, 018h, 07ch, 0c6h, 0c2h, 0c0h, 07ch, 006h, 006h, 086h, 0c6h, 07ch, 018h, 018h, 000h, 000h
515 0001789D 86C67C18180000      <1>
516 000178A4 00000000C2C60C1830- <1>     db  000h, 000h, 000h, 000h, 0c2h, 0c6h, 00ch, 018h, 030h, 060h, 0c6h, 086h, 000h, 000h, 000h, 000h
516 000178AD 60C68600000000      <1>
517 000178B4 0000386C6C3876DCCC- <1>     db  000h, 000h, 038h, 06ch, 06ch, 038h, 076h, 0dch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
517 000178BD CCCC7600000000      <1>
518 000178C4 003030306000000000- <1>     db  000h, 030h, 030h, 030h, 060h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
518 000178CD 00000000000000      <1>
519 000178D4 00000C183030303030- <1>     db  000h, 000h, 00ch, 018h, 030h, 030h, 030h, 030h, 030h, 030h, 018h, 00ch, 000h, 000h, 000h, 000h
519 000178DD 30180C00000000      <1>
520 000178E4 000030180C0C0C0C0C- <1>     db  000h, 000h, 030h, 018h, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 018h, 030h, 000h, 000h, 000h, 000h
520 000178ED 0C183000000000      <1>
521 000178F4 0000000000663CFF3C- <1>     db  000h, 000h, 000h, 000h, 000h, 066h, 03ch, 0ffh, 03ch, 066h, 000h, 000h, 000h, 000h, 000h, 000h
521 000178FD 66000000000000      <1>
522 00017904 0000000000018187E18- <1>     db  000h, 000h, 000h, 000h, 000h, 018h, 018h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h
522 0001790D 18000000000000      <1>
523 00017914 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 030h, 000h, 000h, 000h
523 0001791D 18181830000000      <1>
524 00017924 000000000000000FE00- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
524 0001792D 00000000000000      <1>
525 00017934 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h
525 0001793D 00181800000000      <1>
526 00017944 0000000002060C1830- <1>     db  000h, 000h, 000h, 000h, 002h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 080h, 000h, 000h, 000h, 000h
526 0001794D 60C08000000000      <1>
527 00017954 00003C66C3C3DBDBC3- <1>     db  000h, 000h, 03ch, 066h, 0c3h, 0c3h, 0dbh, 0dbh, 0c3h, 0c3h, 066h, 03ch, 000h, 000h, 000h, 000h
527 0001795D C3663C00000000      <1>
528 00017964 0000183878181818- <1>     db  000h, 000h, 018h, 038h, 078h, 018h, 018h, 018h, 018h, 018h, 018h, 07eh, 000h, 000h, 000h, 000h
528 0001796D 18187E00000000      <1>
529 00017974 00007CC6060C183060- <1>     db  000h, 000h, 07ch, 0c6h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 0c6h, 0feh, 000h, 000h, 000h, 000h
529 0001797D C0C6FE00000000      <1>
530 00017984 00007CC606063C0606- <1>     db  000h, 000h, 07ch, 0c6h, 006h, 006h, 03ch, 006h, 006h, 006h, 0c6h, 07ch, 000h, 000h, 000h, 000h
530 0001798D 06C67C00000000      <1>
531 00017994 00000C1C3C6CCCFE0C- <1>     db  000h, 000h, 00ch, 01ch, 03ch, 06ch, 0cch, 0feh, 00ch, 00ch, 00ch, 01eh, 000h, 000h, 000h, 000h
531 0001799D 0C0C1E00000000      <1>
532 000179A4 0000FEC0C0C0FC0606- <1>     db  000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 0fch, 006h, 006h, 006h, 0c6h, 07ch, 000h, 000h, 000h, 000h
532 000179AD 06C67C00000000      <1>
533 000179B4 00003860C0C0FCC6C6- <1>     db  000h, 000h, 038h, 060h, 0c0h, 0c0h, 0fch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
533 000179BD C6C67C00000000      <1>
534 000179C4 0000FEC606060C1830- <1>     db  000h, 000h, 0feh, 0c6h, 006h, 006h, 00ch, 018h, 030h, 030h, 030h, 030h, 000h, 000h, 000h, 000h
534 000179CD 30303000000000      <1>
535 000179D4 00007CC6C6C67CC6C6- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
535 000179DD C6C67C00000000      <1>
536 000179E4 00007CC6C6C67E0606- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 07eh, 006h, 006h, 006h, 00ch, 078h, 000h, 000h, 000h, 000h
536 000179ED 060C7800000000      <1>
537 000179F4 0000000001818000000- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h
537 000179FD 18180000000000      <1>
538 00017A04 0000000001818000000- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 018h, 018h, 030h, 000h, 000h, 000h, 000h
538 00017A0D 18183000000000      <1>
539 00017A14 000000060C18306030- <1>     db  000h, 000h, 000h, 006h, 00ch, 018h, 030h, 060h, 030h, 018h, 00ch, 006h, 000h, 000h, 000h, 000h
539 00017A1D 180C0600000000      <1>
540 00017A24 00000000007E00007E- <1>     db  000h, 000h, 000h, 000h, 000h, 07eh, 000h, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 000h
540 00017A2D 00000000000000      <1>
541 00017A34 0000006030180C060C- <1>     db  000h, 000h, 000h, 060h, 030h, 018h, 00ch, 006h, 00ch, 018h, 030h, 060h, 000h, 000h, 000h, 000h
541 00017A3D 18306000000000      <1>
542 00017A44 00007CC6C60C181818- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 00ch, 018h, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h, 000h
542 00017A4D 00181800000000      <1>
543 00017A54 0000007CC6C6DEDEDE- <1>     db  000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0deh, 0deh, 0deh, 0dch, 0c0h, 07ch, 000h, 000h, 000h, 000h
543 00017A5D DCC07C00000000      <1>
544 00017A64 000010386CC6C6FEC6- <1>     db  000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
544 00017A6D C6C6C600000000      <1>
545 00017A74 0000FC6666667C6666- <1>     db  000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 066h, 066h, 066h, 066h, 0fch, 000h, 000h, 000h, 000h
545 00017A7D 6666FC00000000      <1>
546 00017A84 00003C66C2C0C0C0C0- <1>     db  000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0c0h, 0c0h, 0c2h, 066h, 03ch, 000h, 000h, 000h, 000h
546 00017A8D C2663C00000000      <1>
```

```
547 00017A94 0000F86C6666666666- <1>     db  000h, 000h, 0f8h, 06ch, 066h, 066h, 066h, 066h, 066h, 066h, 06ch, 0f8h, 000h, 000h, 000h, 000h
547 00017A9D 666CF800000000      <1>
548 00017AA4 0000FE666268786860- <1>     db  000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 060h, 062h, 066h, 0feh, 000h, 000h, 000h, 000h
548 00017AAD 6266FE00000000      <1>
549 00017AB4 0000FE666268786860- <1>     db  000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h
549 00017ABD 6060F000000000      <1>
550 00017AC4 00003C66C2C0C0DEC6- <1>     db  000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0deh, 0c6h, 0c6h, 066h, 03ah, 000h, 000h, 000h, 000h
550 00017ACD C6663A00000000      <1>
551 00017AD4 0000C6C6C6C6FEC6C6- <1>     db  000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
551 00017ADD C6C6C600000000      <1>
552 00017AE4 00003C181818181818- <1>     db  000h, 000h, 03ch, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
552 00017AED 18183C00000000      <1>
553 00017AF4 00001E0C0C0C0C0CCC- <1>     db  000h, 000h, 01eh, 00ch, 00ch, 00ch, 00ch, 00ch, 0cch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h
553 00017AFD CCCC7800000000      <1>
554 00017B04 0000E666666C78786C- <1>     db  000h, 000h, 0e6h, 066h, 066h, 06ch, 078h, 078h, 06ch, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h
554 00017B0D 6666E600000000      <1>
555 00017B14 0000F06060606060C- <1>     db  000h, 000h, 0f0h, 060h, 060h, 060h, 060h, 060h, 060h, 062h, 066h, 0feh, 000h, 000h, 000h, 000h
555 00017B1D 6266FE00000000      <1>
556 00017B24 0000C3E7FFFFDBC3C3- <1>     db  000h, 000h, 0c3h, 0e7h, 0ffh, 0ffh, 0dbh, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 000h, 000h, 000h, 000h
556 00017B2D C3C3C300000000      <1>
557 00017B34 0000C6E6F6FEDECEC6- <1>     db  000h, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
557 00017B3D C6C6C600000000      <1>
558 00017B44 00007CC6C6C6C6C6C6- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
558 00017B4D C6C67C00000000      <1>
559 00017B54 0000FC6666667C6060- <1>     db  000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 060h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h
559 00017B5D 6060F000000000      <1>
560 00017B64 00007CC6C6C6C6C6C6- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0d6h, 0deh, 07ch, 00ch, 00eh, 000h, 000h
560 00017B6D D6DE7C0C0E0000      <1>
561 00017B74 0000FC6666667C6C66- <1>     db  000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 06ch, 066h, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h
561 00017B7D 6666E600000000      <1>
562 00017B84 00007CC6C660380C06- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 060h, 038h, 00ch, 006h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
562 00017B8D C6C67C00000000      <1>
563 00017B94 0000FFDB9918181818- <1>     db  000h, 000h, 0ffh, 0dbh, 099h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
563 00017B9D 18183C00000000      <1>
564 00017BA4 0000C6C6C6C6C6C6C6- <1>     db  000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
564 00017BAD C6C67C00000000      <1>
565 00017BB4 0000C3C3C3C3C3C3C3- <1>     db  000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 000h
565 00017BBD 663C1800000000      <1>
566 00017BC4 0000C3C3C3C3C3DBDB- <1>     db  000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 066h, 000h, 000h, 000h, 000h
566 00017BCD FF666600000000      <1>
567 00017BD4 0000C3C3663C18183C- <1>     db  000h, 000h, 0c3h, 0c3h, 066h, 03ch, 018h, 018h, 03ch, 066h, 0c3h, 0c3h, 000h, 000h, 000h, 000h
567 00017BDD 66C3C300000000      <1>
568 00017BE4 0000C3C3C3663C1818- <1>     db  000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
568 00017BED 18183C00000000      <1>
569 00017BF4 0000FFC3860C183060- <1>     db  000h, 000h, 0ffh, 0c3h, 086h, 00ch, 018h, 030h, 060h, 0c1h, 0c3h, 0ffh, 000h, 000h, 000h, 000h
569 00017BFD C1C3FF00000000      <1>
570 00017C04 00003C303030303030- <1>     db  000h, 000h, 03ch, 030h, 030h, 030h, 030h, 030h, 030h, 030h, 030h, 03ch, 000h, 000h, 000h, 000h
570 00017C0D 30303C00000000      <1>
571 00017C14 00000080C0E070381C- <1>     db  000h, 000h, 000h, 080h, 0c0h, 0e0h, 070h, 038h, 01ch, 00eh, 006h, 002h, 000h, 000h, 000h, 000h
571 00017C1D 0E060200000000      <1>
572 00017C24 00003C0C0C0C0C0C0C- <1>     db  000h, 000h, 03ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 03ch, 000h, 000h, 000h, 000h
572 00017C2D 0C0C3C00000000      <1>
573 00017C34 10386CC600000000000- <1>     db  010h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
573 00017C3D 00000000000000      <1>
574 00017C44 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 000h
574 00017C4D 00000000FF0000      <1>
575 00017C54 3030180000000000000- <1>     db  030h, 030h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
575 00017C5D 00000000000000      <1>
576 00017C64 0000000000780C7CCC- <1>     db  000h, 000h, 000h, 000h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
576 00017C6D CCCC7600000000      <1>
577 00017C74 0000E06060786C6666- <1>     db  000h, 000h, 0e0h, 060h, 060h, 078h, 06ch, 066h, 066h, 066h, 066h, 07ch, 000h, 000h, 000h, 000h
577 00017C7D 66667C00000000      <1>
578 00017C84 000000000007CC6C0C0- <1>     db  000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c0h, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
578 00017C8D C0C67C00000000      <1>
579 00017C94 00001C0C0C3C6CCCCC- <1>     db  000h, 000h, 01ch, 00ch, 00ch, 03ch, 06ch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
579 00017C9D CCCC7600000000      <1>
580 00017CA4 000000000007CC6FEC0- <1>     db  000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
580 00017CAD C0C67C00000000      <1>
581 00017CB4 0000386C6460F06060- <1>     db  000h, 000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h
581 00017CBD 6060F000000000      <1>
582 00017CC4 000000000076CCCCCC- <1>     db  000h, 000h, 000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 0cch, 0cch, 07ch, 00ch, 0cch, 078h, 000h
582 00017CCD CCCC7C0CCC7800      <1>
583 00017CD4 0000E060606C766666- <1>     db  000h, 000h, 0e0h, 060h, 060h, 06ch, 076h, 066h, 066h, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h
583 00017CDD 6666E600000000      <1>
584 00017CE4 000018180038181818- <1>     db  000h, 000h, 018h, 018h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
584 00017CED 18183C00000000      <1>
585 00017CF4 00006060000E060606- <1>     db  000h, 000h, 006h, 006h, 000h, 00eh, 006h, 006h, 006h, 006h, 006h, 006h, 066h, 066h, 03ch, 000h
585 00017CFD 06060666663C00      <1>
586 00017D04 0000E06060666C7878- <1>     db  000h, 000h, 0e0h, 060h, 060h, 066h, 06ch, 078h, 078h, 06ch, 066h, 0e6h, 000h, 000h, 000h, 000h
586 00017D0D 6C66E600000000      <1>
587 00017D14 000038181818181818- <1>     db  000h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
587 00017D1D 18183C00000000      <1>
588 00017D24 0000000000E6FFDBDB- <1>     db  000h, 000h, 000h, 000h, 000h, 0e6h, 0ffh, 0dbh, 0dbh, 0dbh, 0dbh, 0dbh, 000h, 000h, 000h, 000h
588 00017D2D DBDBDB00000000      <1>
589 00017D34 0000000000DC666666- <1>     db  000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 066h, 000h, 000h, 000h, 000h
589 00017D3D 66666600000000      <1>
590 00017D44 000000000007CC6C6C6- <1>     db  000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
590 00017D4D C6C67C00000000      <1>
591 00017D54 0000000000DC666666- <1>     db  000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 07ch, 060h, 060h, 0f0h, 000h
591 00017D5D 66667C6060F000      <1>
592 00017D64 000000000076CCCCCC- <1>     db  000h, 000h, 000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 0cch, 0cch, 07ch, 00ch, 00ch, 01eh, 000h
592 00017D6D CCCC7C0C0C1E00      <1>
593 00017D74 0000000000DC766660- <1>     db  000h, 000h, 000h, 000h, 000h, 0dch, 076h, 066h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h
593 00017D7D 6060F000000000      <1>
594 00017D84 000000000007CC66038- <1>     db  000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 060h, 038h, 00ch, 0c6h, 07ch, 000h, 000h, 000h, 000h
594 00017D8D 0CC67C00000000      <1>
595 00017D94 0000103030FC303030- <1>     db  000h, 000h, 010h, 030h, 030h, 0fch, 030h, 030h, 030h, 030h, 036h, 01ch, 000h, 000h, 000h, 000h
595 00017D9D 30361C00000000      <1>
596 00017DA4 000000000CCCCCCCC- <1>     db  000h, 000h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
596 00017DAD CCCC7600000000      <1>
597 00017DB4 0000000000C3C3C3C3- <1>     db  000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 000h
597 00017DBD 663C1800000000      <1>
598 00017DC4 0000000000C3C3C3DB- <1>     db  000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h, 000h, 000h, 000h
598 00017DCD DBFF6600000000      <1>
599 00017DD4 0000000000C3663C18- <1>     db  000h, 000h, 000h, 000h, 000h, 0c3h, 066h, 03ch, 018h, 03ch, 066h, 0c3h, 000h, 000h, 000h, 000h
599 00017DDD 3C66C300000000      <1>
600 00017DE4 0000000000C6C6C6C6- <1>     db  000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 0f8h, 000h
600 00017DED C6C67E060CF800      <1>
601 00017DF4 0000000000FECC1830- <1>     db  000h, 000h, 000h, 000h, 000h, 0feh, 0cch, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h
601 00017DFD 60C6FE00000000      <1>
602 00017E04 00000E181818701818- <1>     db  000h, 000h, 00eh, 018h, 018h, 018h, 070h, 018h, 018h, 018h, 018h, 00eh, 000h, 000h, 000h, 000h
602 00017E0D 18180E00000000      <1>
603 00017E14 0000181818001818- <1>     db  000h, 000h, 018h, 018h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h
603 00017E1D 18181800000000      <1>
604 00017E24 0000701818180E1818- <1>     db  000h, 000h, 070h, 018h, 018h, 018h, 00eh, 018h, 018h, 018h, 018h, 070h, 000h, 000h, 000h, 000h
604 00017E2D 18187000000000      <1>
605 00017E34 000076DC0000000000- <1>     db  000h, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
605 00017E3D 00000000000000      <1>
```

```
606 00017E44 0000000010386CC6C6- <1>     db  000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h
606 00017E4D C6FE0000000000      <1>
607 00017E54 00003C66C2C0C0C0C2- <1>     db  000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0c0h, 0c2h, 066h, 03ch, 00ch, 006h, 07ch, 000h, 000h
607 00017E5D 663C0C067C0000      <1>
608 00017E64 0000CC0000CCCCCCCC- <1>     db  000h, 000h, 0cch, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
608 00017E6D CCCC7600000000      <1>
609 00017E74 000C1830007CC6FEC0- <1>     db  000h, 00ch, 018h, 030h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
609 00017E7D C0C67C00000000      <1>
610 00017E84 0010386C00780C7CCC- <1>     db  000h, 010h, 038h, 06ch, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
610 00017E8D CCCC7600000000      <1>
611 00017E94 0000CC0000780C7CCC- <1>     db  000h, 000h, 0cch, 000h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
611 00017E9D CCCC7600000000      <1>
612 00017EA4 0060301800780C7CCC- <1>     db  000h, 060h, 030h, 018h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
612 00017EAD CCCC7600000000      <1>
613 00017EB4 00386C3800780C7CCC- <1>     db  000h, 038h, 06ch, 038h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
613 00017EBD CCCC7600000000      <1>
614 00017EC4 000000003C66606066- <1>     db  000h, 000h, 000h, 000h, 03ch, 066h, 060h, 060h, 066h, 03ch, 00ch, 006h, 03ch, 000h, 000h, 000h
614 00017ECD 3C0C063C000000      <1>
615 00017ED4 0010386C007CC6FEC0- <1>     db  000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
615 00017EDD C0C67C00000000      <1>
616 00017EE4 0000C600007CC6FEC0- <1>     db  000h, 000h, 0c6h, 000h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
616 00017EED C0C67C00000000      <1>
617 00017EF4 00603018007CC6FEC0- <1>     db  000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
617 00017EFD C0C67C00000000      <1>
618 00017F04 000066000038181818- <1>     db  000h, 000h, 066h, 000h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
618 00017F0D 18183C00000000      <1>
619 00017F14 00183C660038181818- <1>     db  000h, 018h, 03ch, 066h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
619 00017F1D 18183C00000000      <1>
620 00017F24 006030180038181818- <1>     db  000h, 060h, 030h, 018h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
620 00017F2D 18183C00000000      <1>
621 00017F34 00C60010386CC6C6FE- <1>     db  000h, 0c6h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
621 00017F3D C6C6C600000000      <1>
622 00017F44 386C3800386CC6C6FE- <1>     db  038h, 06ch, 038h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
622 00017F4D C6C6C600000000      <1>
623 00017F54 18306000FE66607C60- <1>     db  018h, 030h, 060h, 000h, 0feh, 066h, 060h, 07ch, 060h, 060h, 066h, 0feh, 000h, 000h, 000h, 000h
623 00017F5D 6066FE00000000      <1>
624 00017F64 00000000006E3B1B7E- <1>     db  000h, 000h, 000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h, 000h, 000h
624 00017F6D D8DC7700000000      <1>
625 00017F74 00003E6CCCCCFECCCC- <1>     db  000h, 000h, 03eh, 06ch, 0cch, 0cch, 0feh, 0cch, 0cch, 0cch, 0cch, 0ceh, 000h, 000h, 000h, 000h
625 00017F7D CCCCCE00000000      <1>
626 00017F84 0010386C007CC6C6C6- <1>     db  000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
626 00017F8D C6C67C00000000      <1>
627 00017F94 0000C600007CC6C6C6- <1>     db  000h, 000h, 0c6h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
627 00017F9D C6C67C00000000      <1>
628 00017FA4 00603018007CC6C6C6- <1>     db  000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
628 00017FAD C6C67C00000000      <1>
629 00017FB4 003078CC00CCCCCCCC- <1>     db  000h, 030h, 078h, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
629 00017FBD CCCC7600000000      <1>
630 00017FC4 0060301800CCCCCCCC- <1>     db  000h, 060h, 030h, 018h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
630 00017FCD CCCC7600000000      <1>
631 00017FD4 0000C60000C6C6C6C6- <1>     db  000h, 000h, 0c6h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 078h, 000h
631 00017FDD C6C67E060C7800      <1>
632 00017FE4 00C6007CC6C6C6C6C6- <1>     db  000h, 0c6h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
632 00017FED C6C67C00000000      <1>
633 00017FF4 00C600C6C6C6C6C6C6- <1>     db  000h, 0c6h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
633 00017FFD C6C67C00000000      <1>
634 00018004 0018187EC3C0C0C0C3- <1>     db  000h, 018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h
634 0001800D 7E181800000000      <1>
635 00018014 00386C6460F0606060- <1>     db  000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h, 060h, 060h, 0e6h, 0fch, 000h, 000h, 000h, 000h
635 0001801D 60E6FC00000000      <1>
636 00018024 0000C3663C18FF18FF- <1>     db  000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 000h, 000h, 000h, 000h
636 0001802D 18181800000000      <1>
637 00018034 00FC66667C62666F66- <1>     db  000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 066h, 0f3h, 000h, 000h, 000h, 000h
637 0001803D 6666F300000000      <1>
638 00018044 000E1B1818187E1818- <1>     db  000h, 00eh, 01bh, 018h, 018h, 018h, 07eh, 018h, 018h, 018h, 018h, 018h, 0d8h, 070h, 000h, 000h
638 0001804D 181818D8700000      <1>
639 00018054 0018306000780C7CCC- <1>     db  000h, 018h, 030h, 060h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
639 0001805D CCCC7600000000      <1>
640 00018064 000C18300038181818- <1>     db  000h, 00ch, 018h, 030h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
640 0001806D 18183C00000000      <1>
641 00018074 00183060007CC6C6C6- <1>     db  000h, 018h, 030h, 060h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
641 0001807D C6C67C00000000      <1>
642 00018084 0018306000CCCCCCCC- <1>     db  000h, 018h, 030h, 060h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
642 0001808D CCCC7600000000      <1>
643 00018094 000076DC00DC666666- <1>     db  000h, 000h, 076h, 0dch, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 066h, 000h, 000h, 000h, 000h
643 0001809D 66666600000000      <1>
644 000180A4 76DC00C6E6F6FEDECE- <1>     db  076h, 0dch, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
644 000180AD C6C6C600000000      <1>
645 000180B4 003C6C6C3E007E0000- <1>     db  000h, 03ch, 06ch, 06ch, 03eh, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
645 000180BD 00000000000000      <1>
646 000180C4 00386C6C38007C0000- <1>     db  000h, 038h, 06ch, 06ch, 038h, 000h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
646 000180CD 00000000000000      <1>
647 000180D4 0000303000303060C0- <1>     db  000h, 000h, 030h, 030h, 000h, 030h, 030h, 060h, 0c0h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
647 000180DD C6C67C00000000      <1>
648 000180E4 000000000000FEC0C0- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h, 000h
648 000180ED C0C00000000000      <1>
649 000180F4 000000000000FE0606- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 0feh, 006h, 006h, 006h, 006h, 000h, 000h, 000h, 000h, 000h
649 000180FD 06060000000000      <1>
650 00018104 00C0C0C2C6CC183060- <1>     db  000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 060h, 0ceh, 09bh, 006h, 00ch, 01fh, 000h, 000h
650 0001810D CE9B060C1F0000      <1>
651 00018114 00C0C0C2C6CC183066- <1>     db  000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 066h, 0ceh, 096h, 03eh, 006h, 006h, 000h, 000h
651 0001811D CE963E06060000      <1>
652 00018124 0000181800181818 3C- <1>     db  000h, 000h, 018h, 018h, 000h, 018h, 018h, 018h, 03ch, 03ch, 03ch, 018h, 000h, 000h, 000h, 000h
652 0001812D 3C3C1800000000      <1>
653 00018134 0000000000366CD86C- <1>     db  000h, 000h, 000h, 000h, 000h, 036h, 06ch, 0d8h, 06ch, 036h, 000h, 000h, 000h, 000h, 000h, 000h
653 0001813D 36000000000000      <1>
654 00018144 0000000000D86C366C- <1>     db  000h, 000h, 000h, 000h, 000h, 0d8h, 06ch, 036h, 06ch, 0d8h, 000h, 000h, 000h, 000h, 000h, 000h
654 0001814D D8000000000000      <1>
655 00018154 114411441144114411- <1>     db  011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h
655 0001815D 44114411441144      <1>
656 00018164 55AA55AA55AA55AA55- <1>     db  055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah
656 0001816D AA55AA55AA55AA      <1>
657 00018174 DD77DD77DD77DD77DD- <1>     db  0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h
657 0001817D 77DD77DD77DD77      <1>
658 00018184 1818181818181818- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
658 0001818D 18181818181818      <1>
659 00018194 1818181818F818F8- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
659 0001819D 18181818181818      <1>
660 000181A4 1818181818F818F8- <1>     db  018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
660 000181AD 18181818181818      <1>
661 000181B4 36363636363636F6- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
661 000181BD 36363636363636      <1>
662 000181C4 000000000000FE36- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
662 000181CD 36363636363636      <1>
663 000181D4 0000000000F818F8- <1>     db  000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
663 000181DD 18181818181818      <1>
664 000181E4 3636363636F606F6- <1>     db  036h, 036h, 036h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
664 000181ED 36363636363636      <1>
```

```
665 000181F4 363636363636363636- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
665 000181FD 36363636363636      <1>
666 00018204 0000000000FE06F636- <1>     db  000h, 000h, 000h, 000h, 000h, 0feh, 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
666 0001820D 36363636363636      <1>
667 00018214 3636363636F606FE00- <1>     db  036h, 036h, 036h, 036h, 036h, 0f6h, 006h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
667 0001821D 00000000000000      <1>
668 00018224 363636363636FE00- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
668 0001822D 00000000000000      <1>
669 00018234 1818181818F818F800- <1>     db  018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
669 0001823D 00000000000000      <1>
670 00018244 00000000000000F818- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
670 0001824D 18181818181818      <1>
671 00018254 181818181818181F00- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
671 0001825D 00000000000000      <1>
672 00018264 18181818181818FF00- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
672 0001826D 00000000000000      <1>
673 00018274 00000000000000FF18- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
673 0001827D 18181818181818      <1>
674 00018284 181818181818181F18- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
674 0001828D 18181818181818      <1>
675 00018294 00000000000000FF00- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
675 0001829D 00000000000000      <1>
676 000182A4 18181818181818FF18- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
676 000182AD 18181818181818      <1>
677 000182B4 18181818181F181F18- <1>     db  018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
677 000182BD 18181818181818      <1>
678 000182C4 363636363636363736- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
678 000182CD 36363636363636      <1>
679 000182D4 363636363637303F00- <1>     db  036h, 036h, 036h, 036h, 036h, 037h, 030h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
679 000182DD 00000000000000      <1>
680 000182E4 00000000003F303736- <1>     db  000h, 000h, 000h, 000h, 000h, 03fh, 030h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
680 000182ED 36363636363636      <1>
681 000182F4 3636363636F700FF00- <1>     db  036h, 036h, 036h, 036h, 036h, 0f7h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
681 000182FD 00000000000000      <1>
682 00018304 0000000000FF00F736- <1>     db  000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0f7h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
682 0001830D 36363636363636      <1>
683 00018314 363636363637303736- <1>     db  036h, 036h, 036h, 036h, 036h, 037h, 030h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
683 0001831D 36363636363636      <1>
684 00018324 0000000000FF00FF00- <1>     db  000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
684 0001832D 00000000000000      <1>
685 00018334 3636363636F700F736- <1>     db  036h, 036h, 036h, 036h, 036h, 0f7h, 000h, 0f7h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
685 0001833D 36363636363636      <1>
686 00018344 1818181818FF00FF00- <1>     db  018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
686 0001834D 00000000000000      <1>
687 00018354 363636363636FF00- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
687 0001835D 00000000000000      <1>
688 00018364 0000000000FF00FF18- <1>     db  000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
688 0001836D 18181818181818      <1>
689 00018374 00000000000000FF36- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
689 0001837D 36363636363636      <1>
690 00018384 363636363636363F00- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
690 0001838D 00000000000000      <1>
691 00018394 18181818181F181F00- <1>     db  018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
691 0001839D 00000000000000      <1>
692 000183A4 00000000001F181F18- <1>     db  000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
692 000183AD 18181818181818      <1>
693 000183B4 000000000000003F36- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 03fh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
693 000183BD 36363636363636      <1>
694 000183C4 363636363636FF36- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
694 000183CD 36363636363636      <1>
695 000183D4 1818181818FF18FF18- <1>     db  018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
695 000183DD 18181818181818      <1>
696 000183E4 181818181818F800- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
696 000183ED 00000000000000      <1>
697 000183F4 000000000000001F18- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
697 000183FD 18181818181818      <1>
698 00018404 FFFFFFFFFFFFFFFFFF- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
698 0001840D FFFFFFFFFFFFFF      <1>
699 00018414 000000000000FFFF- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
699 0001841D FFFFFFFFFFFFFF      <1>
700 00018424 F0F0F0F0F0F0F0F0F0- <1>     db  0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h
700 0001842D F0F0F0F0F0F0F0      <1>
701 00018434 0F0F0F0F0F0F0F0F0F- <1>     db  00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh
701 0001843D 0F0F0F0F0F0F0F      <1>
702 00018444 FFFFFFFFFFFFFF0000- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
702 0001844D 00000000000000      <1>
703 00018454 0000000000076DCD8D8- <1>     db  000h, 000h, 000h, 000h, 000h, 076h, 0dch, 0d8h, 0d8h, 0d8h, 0dch, 076h, 000h, 000h, 000h, 000h
703 0001845D D8DC7600000000      <1>
704 00018464 000078CCCCCCD8CCC6- <1>     db  000h, 000h, 078h, 0cch, 0cch, 0cch, 0d8h, 0cch, 0c6h, 0c6h, 0c6h, 0cch, 000h, 000h, 000h, 000h
704 0001846D C6C6CC00000000      <1>
705 00018474 0000FEC6C6C0C0C0C0- <1>     db  000h, 000h, 0feh, 0c6h, 0c6h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h
705 0001847D C0C0C000000000      <1>
706 00018484 00000000FE6C6C6C6C- <1>     db  000h, 000h, 000h, 000h, 0feh, 06ch, 06ch, 06ch, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h
706 0001848D 6C6C6C00000000      <1>
707 00018494 000000FEC660301830- <1>     db  000h, 000h, 000h, 0feh, 0c6h, 060h, 030h, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h
707 0001849D 60C6FE00000000      <1>
708 000184A4 00000000007ED8D8D8- <1>     db  000h, 000h, 000h, 000h, 000h, 07eh, 0d8h, 0d8h, 0d8h, 0d8h, 0d8h, 070h, 000h, 000h, 000h, 000h
708 000184AD D8D87000000000      <1>
709 000184B4 0000000066666666666- <1>     db  000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h, 066h, 07ch, 060h, 060h, 0c0h, 000h, 000h, 000h
709 000184BD 7C6060C0000000      <1>
710 000184C4 0000000076DC181818- <1>     db  000h, 000h, 000h, 000h, 076h, 0dch, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h
710 000184CD 18181800000000      <1>
711 000184D4 0000007E183C666666- <1>     db  000h, 000h, 000h, 07eh, 018h, 03ch, 066h, 066h, 066h, 03ch, 018h, 07eh, 000h, 000h, 000h, 000h
711 000184DD 3C187E00000000      <1>
712 000184E4 000000386CC6C6FEC6- <1>     db  000h, 000h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h
712 000184ED C66C3800000000      <1>
713 000184F4 0000386CC6C6C66C6C- <1>     db  000h, 000h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 06ch, 06ch, 06ch, 06ch, 0eeh, 000h, 000h, 000h, 000h
713 000184FD 6C6CEE00000000      <1>
714 00018504 00001E30180C3E6666- <1>     db  000h, 000h, 01eh, 030h, 018h, 00ch, 03eh, 066h, 066h, 066h, 066h, 03ch, 000h, 000h, 000h, 000h
714 0001850D 66663C00000000      <1>
715 00018514 00000000007EDBDBDB- <1>     db  000h, 000h, 000h, 000h, 000h, 07eh, 0dbh, 0dbh, 0dbh, 07eh, 000h, 000h, 000h, 000h, 000h, 000h
715 0001851D 7E000000000000      <1>
716 00018524 00000003067EDBDBF3- <1>     db  000h, 000h, 000h, 003h, 006h, 07eh, 0dbh, 0dbh, 0f3h, 07eh, 060h, 0c0h, 000h, 000h, 000h, 000h
716 0001852D 7E60C000000000      <1>
717 00018534 00001C3060607C6060- <1>     db  000h, 000h, 01ch, 030h, 060h, 060h, 07ch, 060h, 060h, 060h, 030h, 01ch, 000h, 000h, 000h, 000h
717 0001853D 60301C00000000      <1>
718 00018544 0000007CC6C6C6C6C6- <1>     db  000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
718 0001854D C6C6C600000000      <1>
719 00018554 00000000FE0000FE00- <1>     db  000h, 000h, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 000h, 000h, 000h
719 0001855D 00FE0000000000      <1>
720 00018564 0000000018187E1818- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 07eh, 018h, 018h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h
720 0001856D 0000FF00000000      <1>
721 00018574 00000030180C060C18- <1>     db  000h, 000h, 000h, 030h, 018h, 00ch, 006h, 00ch, 018h, 030h, 000h, 07eh, 000h, 000h, 000h, 000h
721 0001857D 30007E00000000      <1>
722 00018584 0000000C1830603018- <1>     db  000h, 000h, 000h, 00ch, 018h, 030h, 060h, 030h, 018h, 00ch, 000h, 07eh, 000h, 000h, 000h, 000h
722 0001858D 0C007E00000000      <1>
723 00018594 00000E1B1B18181818- <1>     db  000h, 000h, 00eh, 01bh, 01bh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
723 0001859D 18181818181818      <1>
```

```
724 000185A4 1818181818181818D8- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0d8h, 0d8h, 0d8h, 070h, 000h, 000h, 000h, 000h
724 000185AD D8D87000000000      <1>
725 000185B4 000000001818007E00- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 000h, 07eh, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h
725 000185BD 18180000000000      <1>
726 000185C4 000000000076DC0076- <1>     db  000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h
726 000185CD DC000000000000      <1>
727 000185D4 00386C6C3800000000- <1>     db  000h, 038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
727 000185DD 00000000000000      <1>
728 000185E4 000000000000001818- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
728 000185ED 00000000000000      <1>
729 000185F4 000000000000000018- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
729 000185FD 00000000000000      <1>
730 00018604 000F0C0C0C0C0CEC6C- <1>     db  000h, 00fh, 00ch, 00ch, 00ch, 00ch, 00ch, 0ech, 06ch, 06ch, 03ch, 01ch, 000h, 000h, 000h, 000h
730 0001860D 6C3C1C00000000      <1>
731 00018614 00D86C6C6C6C6C0000- <1>     db  000h, 0d8h, 06ch, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
731 0001861D 00000000000000      <1>
732 00018624 0070D83060C8F80000- <1>     db  000h, 070h, 0d8h, 030h, 060h, 0c8h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
732 0001862D 00000000000000      <1>
733 00018634 000000007C7C7C7C7C- <1>     db  000h, 000h, 000h, 000h, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 000h, 000h, 000h, 000h, 000h
733 0001863D 7C7C0000000000      <1>
734 00018644 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
734 0001864D 00000000000000      <1>
735                              <1>
736                              <1> ; 01/01/2021 (TRDOS 386 v2.0.3)
737                              <1>
738                              <1> %if 0
739                              <1>
740                              <1> vgafont14alt:
741                              <1>     db  01dh, 000h, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h, 000h, 000h, 000h, 022h
742                              <1>     db  000h, 063h, 063h, 063h, 022h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 02bh, 000h
743                              <1>     db  000h, 000h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 02dh, 000h, 000h
744                              <1>     db  000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 04dh, 000h, 000h, 0c3h
745                              <1>     db  0e7h, 0ffh, 0dbh, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 000h, 000h, 000h, 054h, 000h, 000h, 0ffh, 0dbh
746                              <1>     db  099h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 056h, 000h, 000h, 0c3h, 0c3h, 0c3h
747                              <1>     db  0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 057h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h
748                              <1>     db  0dbh, 0dbh, 0ffh, 066h, 066h, 000h, 000h, 000h, 058h, 000h, 000h, 0c3h, 0c3h, 066h, 03ch, 018h
749                              <1>     db  03ch, 066h, 0c3h, 0c3h, 000h, 000h, 000h, 059h, 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h
750                              <1>     db  018h, 018h, 03ch, 000h, 000h, 000h, 05ah, 000h, 000h, 0ffh, 0c3h, 086h, 00ch, 018h, 030h, 061h
751                              <1>     db  0c3h, 0ffh, 000h, 000h, 000h, 06dh, 000h, 000h, 000h, 000h, 000h, 0e6h, 0ffh, 0dbh, 0dbh, 0dbh
752                              <1>     db  0dbh, 000h, 000h, 000h, 076h, 000h, 000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h
753                              <1>     db  000h, 000h, 000h, 077h, 000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h
754                              <1>     db  000h, 000h, 091h, 000h, 000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h
755                              <1>     db  000h, 09bh, 000h, 018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h
756                              <1>     db  09dh, 000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 000h, 000h, 000h, 09eh
757                              <1>     db  000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 0f3h, 000h, 000h, 000h, 0f1h, 000h
758                              <1>     db  000h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 000h, 0ffh, 000h, 000h, 000h, 0f6h, 000h, 000h
759                              <1>     db  018h, 018h, 000h, 000h, 0ffh, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h
760                              <1> vgafont16alt:
761                              <1>     db  01dh, 000h, 000h, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h, 000h, 000h, 000h
762                              <1>     db  000h, 030h, 000h, 000h, 03ch, 066h, 0c3h, 0c3h, 0dbh, 0dbh, 0c3h, 0c3h, 066h, 03ch, 000h, 000h
763                              <1>     db  000h, 000h, 04dh, 000h, 000h, 0c3h, 0e7h, 0ffh, 0ffh, 0dbh, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 000h
764                              <1>     db  000h, 000h, 000h, 054h, 000h, 000h, 0ffh, 0dbh, 099h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch
765                              <1>     db  000h, 000h, 000h, 000h, 056h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch
766                              <1>     db  018h, 000h, 000h, 000h, 000h, 057h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh
767                              <1>     db  066h, 066h, 000h, 000h, 000h, 000h, 058h, 000h, 000h, 0c3h, 0c3h, 066h, 03ch, 018h, 018h, 03ch
768                              <1>     db  066h, 0c3h, 0c3h, 000h, 000h, 000h, 000h, 059h, 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h
769                              <1>     db  018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 05ah, 000h, 000h, 0ffh, 0c3h, 086h, 00ch, 018h
770                              <1>     db  030h, 060h, 0c1h, 0c3h, 0ffh, 000h, 000h, 000h, 000h, 06dh, 000h, 000h, 000h, 000h, 000h, 0e6h
771                              <1>     db  0ffh, 0dbh, 0dbh, 0dbh, 0dbh, 0dbh, 000h, 000h, 000h, 000h, 076h, 000h, 000h, 000h, 000h, 000h
772                              <1>     db  0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 000h, 077h, 000h, 000h, 000h, 000h
773                              <1>     db  000h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h, 000h, 000h, 000h, 078h, 000h, 000h, 000h
774                              <1>     db  000h, 000h, 0c3h, 066h, 03ch, 018h, 03ch, 066h, 0c3h, 000h, 000h, 000h, 091h, 000h, 000h, 000h
775                              <1>     db  000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h, 000h, 09bh, 000h, 000h
776                              <1>     db  018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h, 09dh
777                              <1>     db  000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 000h, 000h, 000h, 000h
778                              <1>     db  09eh, 000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 066h, 0f3h, 000h, 000h, 000h
779                              <1>     db  000h, 0abh, 000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 060h, 0ceh, 09bh, 006h, 00ch, 01fh
780                              <1>     db  000h, 000h, 0ach, 000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 066h, 0ceh, 096h, 03eh, 006h
781                              <1>     db  006h, 000h, 000h, 000h
782                              <1>
783                              <1> %endif
3429
3430                                     ; 20/11/2020
3431                                     vbe2_bochs_vbios:
3432                                     ;     db "BOCHS/QEMU"
3433 00018654 424F4348532F51454D-           db "BOCHS/QEMU/VIRTUALBOX"  ; 26/11/2020
3433 0001865D 552F5649525455414C-
3433 00018666 424F58
3434                                     ;vbe_vnumber equ vbe2_bochs_vbios + 28 ; "3" or "2"
3435                                     ; 26/11/2020
3436                                     vbe_vnumber equ vbe2_bochs_vbios + 30 ; "3" or "2"
3437                                     ; 15/11/2020
3438                                     vesa_vbe3_bios_msg:
3439                                           ;db " VESA VBE version 3 Video BIOS ..."
3440 00018669 205645534120564245-           db " VESA VBE3 Video BIOS ..." ; 26/11/2020
3440 00018672 3320566964656F2042-
3440 0001867B 494F53202E2E2E
3441 00018682 0D0A0D0A00                    db 0Dh, 0Ah, 0Dh, 0Ah, 0
3442
3443 00018687 90                     align 2
3444
3445                                 ; EPOCH Variables
3446                                 ; 13/04/2015 - Retro UNIX 386 v1 Beginning
3447                                 ; 09/04/2013 epoch variables
3448                                 ; Retro UNIX 8086 v1 Prototype: UNIXCOPY.ASM, 10/03/2013
3449                                 ;
3450 00018688 B207                  year:       dw 1970
3451 0001868A 0100                  month:      dw 1
3452 0001868C 0100                  day:  dw 1
3453 0001868E 0000                  hour:       dw 0
3454 00018690 0000                  minute: dw 0
3455 00018692 0000                  second: dw 0
3456
3457                                 DMonth:
3458 00018694 0000                         dw 0
3459 00018696 1F00                         dw 31
3460 00018698 3B00                         dw 59
3461 0001869A 5A00                         dw 90
3462 0001869C 7800                         dw 120
3463 0001869E 9700                         dw 151
3464 000186A0 B500                         dw 181
3465 000186A2 D400                         dw 212
```

```
3466 000186A4 F300                      dw 243
3467 000186A6 1101                      dw 273
3468 000186A8 3001                      dw 304
3469 000186AA 4E01                      dw 334
3470
3471                                    ; 15/11/2020
3472 000186AC 00                        db    0
3473                                    kernel_version_msg: ; 20/11/2020
3474 000186AD 5452444F5320283338-       db    "TRDOS (386) Kernel v2.0.3 by Erdogan Tan"
3474 000186B6 3629204B65726E656C-
3474 000186BF 2076322E302E332062-
3474 000186C8 79204572646F67616E-
3474 000186D1 2054616E
3475 000186D5 00                        db    0
3476
3477                                    ; 20/02/2017
3478                                    KERNELFSIZE  equ $  ; 04/07/2016
3479
3480                                    bss_start:
3481
3482                                    ABSOLUTE bss_start
3483
3484 000186D6 <res 00000002>            alignb 8 ; 25/12/2016
3485
3486                                         ; 15/04/2016
3487                                         ; TRDOS 386 (TRDOS v2.0)
3488                                         ;     80 interrupts
3489                                         ; 11/03/2015
3490                                         ; Interrupt Descriptor Table (20/08/2014)
3491                                    idt:
3492                                         ;resb  64*8 ; INT 0 to INT 3Fh
3493                                         ; 15/04/2016
3494 000186D8 <res 00000280>                 resb   80*8 ; INT 0 to INT 4Fh
3495
3496                                    idt_end:
3497
3498                                    ;alignb 4
3499
3500                                    task_state_segment:
3501                                         ; 24/03/2015
3502 00018958 <res 00000002>            tss.link:   resw 1
3503 0001895A <res 00000002>                        resw 1
3504                                    ; tss offset 4
3505 0001895C <res 00000004>            tss.esp0:   resd 1
3506 00018960 <res 00000002>            tss.ss0:    resw 1
3507 00018962 <res 00000002>                        resw 1
3508 00018964 <res 00000004>            tss.esp1:   resd 1
3509 00018968 <res 00000002>            tss.ss1:    resw 1
3510 0001896A <res 00000002>                        resw 1
3511 0001896C <res 00000004>            tss.esp2:   resd 1
3512 00018970 <res 00000002>            tss.ss2:    resw 1
3513 00018972 <res 00000002>                        resw 1
3514                                    ; tss offset 28
3515 00018974 <res 00000004>            tss.CR3:    resd 1
3516 00018978 <res 00000004>            tss.eip:    resd 1
3517 0001897C <res 00000004>            tss.eflags: resd 1
3518                                    ; tss offset 40
3519 00018980 <res 00000004>            tss.eax:    resd 1
3520 00018984 <res 00000004>            tss.ecx:    resd 1
3521 00018988 <res 00000004>            tss.edx:    resd 1
3522 0001898C <res 00000004>            tss.ebx:    resd 1
3523 00018990 <res 00000004>            tss.esp:    resd 1
3524 00018994 <res 00000004>            tss.ebp:    resd 1
3525 00018998 <res 00000004>            tss.esi:    resd 1
3526 0001899C <res 00000004>            tss.edi:    resd 1
3527                                    ; tss offset 72
3528 000189A0 <res 00000002>            tss.ES:     resw 1
3529 000189A2 <res 00000002>                        resw 1
3530 000189A4 <res 00000002>            tss.CS:          resw 1
3531 000189A6 <res 00000002>                        resw 1
3532 000189A8 <res 00000002>            tss.SS:          resw 1
3533 000189AA <res 00000002>                        resw 1
3534 000189AC <res 00000002>            tss.DS:          resw 1
3535 000189AE <res 00000002>                        resw 1
3536 000189B0 <res 00000002>            tss.FS:          resw 1
3537 000189B2 <res 00000002>                        resw 1
3538 000189B4 <res 00000002>            tss.GS:          resw 1
3539 000189B6 <res 00000002>                        resw 1
3540 000189B8 <res 00000002>            tss.LDTR:   resw 1
3541 000189BA <res 00000002>                        resw 1
3542                                    ; tss offset 100
3543 000189BC <res 00000002>                        resw 1
3544 000189BE <res 00000002>            tss.IOPB:   resw 1
3545                                    ; tss offset 104
3546                                    tss_end:
3547
3548 000189C0 <res 00000004>            k_page_dir:  resd 1 ; Kernel's (System) Page Directory address
3549                                                         ; (Physical address = Virtual address)
3550 000189C4 <res 00000004>            memory_size: resd 1 ; memory size in pages
3551 000189C8 <res 00000004>            free_pages:  resd 1 ; number of free pages
3552 000189CC <res 00000004>            next_page:   resd 1 ; offset value in M.A.T. for
3553                                                         ; first free page search
3554 000189D0 <res 00000004>            last_page:   resd 1 ; offset value in M.A.T. which
3555                                                         ; next free page search will be
3556                                                         ; stopped after it. (end of M.A.T.)
3557 000189D4 <res 00000004>            first_page:  resd 1 ; offset value in M.A.T. which
3558                                                         ; first free page search
3559                                                         ; will be started on it. (for user)
3560 000189D8 <res 00000004>            mat_size:    resd 1 ; Memory Allocation Table size in pages
3561
3562                                    ; 20/11/2020
3563                                    ;vbe2bios:    resw 1 ; VBE2 video bios ID (bochs/qemu)
3564                                    ;                    ; (0B0C4h or 0B0C5h for bochs/plex86 vgabios)
3565
3566                                    ; 02/09/2014 (Retro UNIX 386 v1)
```

```
                                        ; 04/12/2013 (Retro UNIX 8086 v1)
000189DC <res 00000002>                 CRT_START:   resw 1         ; starting address in regen buffer
                                                              ; NOTE: active page only
000189DE <res 00000010>                 CURSOR_POSN: resw 8 ; cursor positions for video pages
                                        ACTIVE_PAGE:
000189EE <res 00000001>                 ptty:            resb 1 ; current tty
                                        ; 01/07/2015 - 29/01/2016
000189EF <res 00000001>                 ccolor:          resb 1 ; current color attribute
                                        ; 26/10/2015
                                        ; 07/09/2014
000189F0 <res 00000014>                 ttychr:      resw ntty+2 ; Character buffer (multiscreen)

                                        ; 18/05/2015 (03/06/2013 - Retro UNIX 8086 v1 feature only!)
00018A04 <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) ]
00018A08 <res 00000014>                 ttyl:        resw ntty+2 ; opening locks for TTYs.

                                        ; 15/04/2015 (Retro UNIX 386 v1)
                                        ; 22/09/2013 (Retro UNIX 8086 v1)
00018A1C <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
00018A26 <res 00000001>                 com1p:       resb 1  ;;0E3h
00018A27 <res 00000001>                 com2p:       resb 1  ;;0E3h

                                        ; 17/11/2015
                                        ; request for response (from the terminal)
00018A28 <res 00000002>                 req_resp:    resw 1

                                        ; 07/11/2015
00018A2A <res 00000001>                 ccomport:    resb 1 ; current COM (serial) port
                                                             ; (0= COM1, 1= COM2)
                                        ; 09/11/2015
00018A2B <res 00000001>                 comqr:           resb 1 ; 'query or response' sign (u9.s, 'sndc')
                                        ; 07/11/2015
00018A2C <res 00000002>                 rchar:           resw 1 ; last received char for COM 1 and COM 2
00018A2E <res 00000002>                 schar:           resw 1 ; last sent char for COM 1 and COM 2

                                        ; 22/08/2014 (RTC)
                                        ; (Packed BCD)
00018A30 <res 00000001>                 time_seconds: resb 1
00018A31 <res 00000001>                 time_minutes: resb 1
00018A32 <res 00000001>                 time_hours:   resb 1
00018A33 <res 00000001>                 date_wday:    resb 1
00018A34 <res 00000001>                 date_day:     resb 1
00018A35 <res 00000001>                 date_month:   resb 1
00018A36 <res 00000001>                 date_year:    resb 1
00018A37 <res 00000001>                 date_century: resb 1

                                        ; 24/01/2016
00018A38 <res 00000004>                 RTC_LH:            resd 1
00018A3C <res 00000001>                 RTC_WAIT_FLAG: resb 1
00018A3D <res 00000001>                 USER_FLAG:     resb 1
                                        ; 19/05/2016
                                        ;RTC_second:
00018A3E <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>
00018A3F <res 00000001>             <1> alignb 2
                                    <1>
                                    <1> ;----------------------------------------
                                    <1> ;     TIMER DATA AREA           :
                                    <1> ;----------------------------------------
                                    <1>
                                    <1> TIMER_LH:    ; 16/02/205
00018A40 <res 00000002>             <1> TIMER_LOW:      resw     1               ; LOW WORD OF TIMER COUNT
00018A42 <res 00000002>             <1> TIMER_HIGH:     resw     1               ; HIGH WORD OF TIMER COUNT
00018A44 <res 00000001>             <1> TIMER_OFL:      resb     1               ; TIMER HAS ROLLED OVER SINCE LAST READ
                                    <1>
                                    <1> ;----------------------------------------
                                    <1> ;     DISKETTE DATA AREAS       :
                                    <1> ;----------------------------------------
                                    <1>
00018A45 <res 00000001>             <1> SEEK_STATUS: resb   1
00018A46 <res 00000001>             <1> MOTOR_STATUS:      resb   1
00018A47 <res 00000001>             <1> MOTOR_COUNT: resb   1
00018A48 <res 00000001>             <1> DSKETTE_STATUS:    resb   1
00018A49 <res 00000007>             <1> NEC_STATUS:  resb   7
                                    <1>
```

```
                            <1> ;----------------------------------------
                            <1> ;     ADDITIONAL MEDIA DATA            :
                            <1> ;----------------------------------------
                            <1>
00018A50 <res 00000001>     <1> LASTRATE:    resb   1
00018A51 <res 00000001>     <1> HF_STATUS:   resb   1
00018A52 <res 00000001>     <1> HF_ERROR:    resb   1
00018A53 <res 00000001>     <1> HF_INT_FLAG: resb   1
00018A54 <res 00000001>     <1> HF_CNTRL:    resb   1
00018A55 <res 00000004>     <1> DSK_STATE:   resb   4
00018A59 <res 00000002>     <1> DSK_TRK:     resb   2
                            <1>
                            <1> ;----------------------------------------
                            <1> ;     FIXED DISK DATA AREAS            :
                            <1> ;----------------------------------------
                            <1>
00018A5B <res 00000001>     <1> DISK_STATUS1:      resb   1            ; FIXED DISK STATUS
00018A5C <res 00000001>     <1> HF_NUM:            resb   1            ; COUNT OF FIXED DISK DRIVES
00018A5D <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>
00018A5E <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
00018A60 <res 00000004>     <1> HDPM_TBL_VEC:      resd   1           ; Primary master disk param. tbl. pointer
00018A64 <res 00000004>     <1> HDPS_TBL_VEC:      resd   1           ; Primary slave disk param. tbl. pointer
00018A68 <res 00000004>     <1> HDSM_TBL_VEC:      resd   1           ; Secondary master disk param. tbl. pointer
00018A6C <res 00000004>     <1> HDSS_TBL_VEC:      resd   1           ; Secondary slave disk param. tbl. pointer
                            <1>
                            <1> ; 03/01/2015
00018A70 <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 - v2.0.2) - UNINITIALIZED DATA : trdoskx.s
                            <1> ; ----------------------------------------------------------------------------
                            <1> ; Last Update: 30/08/2020
                            <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>
00018A71 <res 00000003>     <1> alignb 4
                            <1>
                            <1> ; MAINPROG.ASM
00018A74 <res 00000004>     <1> MainProgCfg_FileSize:   resd 1 ; 14/04/2016
00018A78 <res 00000004>     <1> MainProgCfg_LineOffset: resd 1 ; 14/04/2016
                            <1>
00018A7C <res 00000004>     <1> Current_VolSerial: resd 1
                            <1>
00018A80 <res 00000004>     <1> Current_Dir_FCluster: resd 1
                            <1>
00018A84 <res 00000001>     <1> Current_Dir_Level: resb 1
00018A85 <res 00000001>     <1> Current_FATType: resb 1
00018A86 <res 00000001>     <1> Current_Drv: resb 1
00018A87 <res 00000001>     <1> Current_Dir_Drv:   resb 1 ; '?'
00018A88 <res 00000001>     <1>                    resb 1 ; ':'
00018A89 <res 00000001>     <1> Current_Dir_Root:  resb 1 ; '/'
00018A8A <res 0000005A>     <1> Current_Directory: resb 90
00018AE4 <res 00000001>     <1> End_Of_Current_Dir_Str: resb 1
00018AE5 <res 00000001>     <1> Current_Dir_StrLen: resb 1
                            <1>
00018AE6 <res 00000001>     <1> CursorColumn:      resb 1
00018AE7 <res 00000001>     <1> CmdArgStart:    resb 1
                            <1>
                            <1> ; 03/02/2016
00018AE8 <res 0000004E>     <1> Remark:            resb 78
                            <1>
00018B36 <res 00000050>     <1> CommandBuffer:     resb 80
                            <1>
00018B86 <res 00000100>     <1> TextBuffer:  resb 256
                            <1>
                            <1> MasterBootBuff:
00018C86 <res 000001BE>     <1> MasterBootCode: resb 1BEh
00018E44 <res 00000040>     <1> PartitionTable: resb 64
00018E84 <res 00000002>     <1> MBIDCode: resw 1
                            <1>
                            <1> PTable_Buffer:
00018E86 <res 00000040>     <1> PTable_hd0: resb 64
00018EC6 <res 00000040>     <1> PTable_hd1: resb 64
00018F06 <res 00000040>     <1> PTable_hd2: resb 64
00018F46 <res 00000040>     <1> PTable_hd3: resb 64
                            <1> ; 15/07/2020
                            <1> ;PTable_ep0: resb 64
                            <1> ;PTable_ep1: resb 64
```

```
                                <1> ;PTable_ep2: resb 64
                                <1> ;PTable_ep3: resb 64
                                <1>
                                <1> ; 13/08/2020
00018F86 <res 00000001>         <1> scount: resb 1 ; 16/05/2016 (diskio.s, 'int33h:')
00018F87 <res 00000001>         <1>       resb 1
00018F88 <res 00000001>         <1>       resb 1
00018F89 <res 00000001>         <1>       resb 1
                                <1>
00018F8A <res 00000001>         <1> HD_LBA_yes: resb 1
00018F8B <res 00000001>         <1> PP_Counter: resb 1
00018F8C <res 00000001>         <1> EP_Counter: resb 1
                                <1> ; 13/08/2020
00018F8D <res 00000001>         <1> LD_Counter: resb 1
                                <1>
                                <1> ; 30/08/2020
00018F8E <res 00000004>         <1> MBR_EP_StartSector: resd 1
                                <1>                   ; Extd partition start sector as in MBR
00018F92 <res 00000004>         <1> EP_StartSector: resd 1 ; next extd partition start sector
                                <1>       ; 15/07/2020
                                <1>                 ;resd 1
                                <1>                 ;resd 1
                                <1>
                                <1> ; 20/07/2020
00018F96 <res 00000200>         <1> DOSBootSectorBuff: resb 512
                                <1> ; 15/07/2020
                                <1> ;DOSBootSectorBuff: resb 446 ; 1BEh
                                <1> ;MiniPartitionTable: resb 64 ;  40h
                                <1> ;MiniPartitionMagic: resw  1 ;  02h
                                <1>
                                <1> FAT_BuffDescriptor:
00019196 <res 00000004>         <1> FAT_CurrentCluster: resd 1
0001919A <res 00000001>         <1> FAT_BuffValidData: resb 1
0001919B <res 00000001>         <1> FAT_BuffDrvName: resb 1
0001919C <res 00000002>         <1> FAT_BuffOffset: resw 1
0001919E <res 00000004>         <1> FAT_BuffSector: resd 1
                                <1>
000191A2 <res 00000004>         <1> FAT_ClusterCounter: resd 1
000191A6 <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:
000191AA <res 00000001>         <1> DirBuff_DRV: resb 1
000191AB <res 00000001>         <1> DirBuff_FATType: resb 1
000191AC <res 00000001>         <1> DirBuff_ValidData: resb 1
000191AD <res 00000002>         <1> DirBuff_CurrentEntry: resw 1
000191AF <res 00000002>         <1> DirBuff_LastEntry: resw 1
000191B1 <res 00000004>         <1> DirBuff_Cluster: resd 1
000191B5 <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>
000191B7 <res 00000004>         <1> FreeClusterCount: resd 1
                                <1>
000191BB <res 00000004>         <1> VolSize_Unit1:   resd 1
000191BF <res 00000004>         <1> VolSize_Unit2:   resd 1
                                <1>
000191C3 <res 00000004>         <1> Vol_Tot_Sec_Str_Start:       resd 1
000191C7 <res 0000000A>         <1> Vol_Tot_Sec_Str:       resb 10
000191D1 <res 00000001>         <1> Vol_Tot_Sec_Str_End:         resb 1
000191D2 <res 00000001>         <1> resb 1
000191D3 <res 00000004>         <1> Vol_Free_Sectors_Str_Start: resd 1
000191D7 <res 0000000A>         <1> Vol_Free_Sectors_Str:        resb 10
000191E1 <res 00000001>         <1> Vol_Free_Sectors_Str_End:   resb 1
                                <1>
                                <1> ; 10/02/2016
000191E2 <res 00000001>         <1> RUN_CDRV: resb 1 ; CMD_INTR.ASM  ; 09/11/2011
                                <1>
                                <1> ; 24/01/2016
000191E3 <res 00000080>         <1> PATH_Array:     resb 128 ; DIR.ASM ; 09/10/2011
                                <1> ; 06/02/2016
00019263 <res 00000004>         <1> CCD_DriveDT: resd 1 ; DIR.ASM ; (word)
00019267 <res 00000001>         <1> CCD_Level:   resb 1 ; DIR.ASM
00019268 <res 00000001>         <1> Last_Dir_Level:    resb 1 ; DIR.ASM
                                <1> ;
00019269 <res 00000002>         <1> CDLF_AttributesMask: resw 1 ; DIR.ASM
0001926B <res 00000004>         <1> CDLF_FNAddress:    resd 1 ; DIR.ASM (word)
0001926F <res 00000002>         <1> CDLF_DEType: resw 1 ; DIR.ASM
                                <1> ;
00019271 <res 00000001>         <1> CD_COMMAND:  resb 1 ; DIR.ASM
                                <1>
00019272 <res 00000002>         <1> alignb 4
                                <1>
                                <1> ; 29/01/2016
00019274 <res 00000001>         <1> Program_Exit:      resb 1 ; CMD_INTR.ASM  ; 09/11/2011
                                <1>
                                <1> ;alignb 4
                                <1> ; 23/02/2016
00019275 <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
00019276 <res 00000001>         <1> retry_count:       resb 1 ; DISK_IO.ASM ; 20/07/2011 (CHS_RetryCount)
00019277 <res 00000001>         <1> disk_rw_err:       resb 1 ; DISK_IO.ASM ; (Disk_IO_err_code)
00019278 <res 00000004>         <1> sector_count:      resd 1 ; DISK_IO.ASM ; (Disk_RW_SectorCount)
                                <1>
```

```
167                             <1> ; 06/02/2016 (long name)
168 0001927C <res 00000002>     <1> FDE_AttrMask:         resw 1 ; DIR.ASM
169 0001927E <res 00000002>     <1> AmbiguousFileName: resw 1 ; DIR.ASM
170 00019280 <res 00000001>     <1> PreviousAttr:         resb 1 ; DIR.ASM
171                             <1> ;
172 00019281 <res 00000001>     <1> LongNameFound:   resb 1     ; DIR.ASM
173 00019282 <res 00000001>     <1> LFN_EntryLength: resb 1   ; DIR.ASM
174 00019283 <res 00000001>     <1> LFN_CheckSum:    resb 1   ; DIR.ASM
175 00019284 <res 00000084>     <1> LongFileName:    resb 132 ; DIR.ASM
176                             <1>
177                             <1> ;PATH_Array_Ptr: resw 1 ; DIR.ASM
178 00019308 <res 00000001>     <1> PATH_CDLevel:       resb 1 ; DIR.ASM
179 00019309 <res 00000001>     <1> PATH_Level:   resb 1 ; DIR.ASM
180                             <1>
181                             <1> ; 07/02/2016
182 0001930A <res 0000000D>     <1> Dir_File_Name:     resb 13 ; DIR.ASM ; 09/10/2011
183                             <1>
184                             <1> ; 10/02/2016
185 00019317 <res 0000000D>     <1> Dir_Entry_Name:    resb 13 ; DIR.ASM
186                             <1>
187                             <1> alignb 2
188                             <1>
189 00019324 <res 00000002>     <1> AttributesMask: resw 1 ; CMD_INTR.ASM ; 09/11/2011
190                             <1>
191                             <1> ; 10/02/2016 (128 bytes -> 126 bytes)
192                             <1> ; 08/02/2016
193                             <1> ;FFF Structure (128 bytes) ; DIR.ASM ; 09/10/2011
194 00019326 <res 00000001>     <1> FindFile_Drv:              resb 1
195 00019327 <res 00000041>     <1> FindFile_Directory:        resb 65
196 00019368 <res 0000000D>     <1> FindFile_Name:             resb 13
197                             <1> FindFile_LongNameEntryLength:
198 00019375 <res 00000001>     <1> FindFile_LongNameYes:      resb 1 ; Sign for longname procedures
199                             <1> ;Above 80 bytes form
200                             <1> ;TR-DOS Source/Destination File FullName Format/Structure
201 00019376 <res 00000002>     <1> FindFile_AttributesMask:  resw 1
202 00019378 <res 00000020>     <1> FindFile_DirEntry:   resb 32
203 00019398 <res 00000004>     <1> FindFile_DirFirstCluster: resd 1
204 0001939C <res 00000004>     <1> FindFile_DirCluster:       resd 1
205 000193A0 <res 00000002>     <1> FindFile_DirEntryNumber:  resw 1
206 000193A2 <res 00000002>     <1> FindFile_MatchCounter:     resw 1
207 000193A4 <res 00000002>     <1> FindFile_Reserved:   resw 1 ; 06/03/2016
208                             <1>
209 000193A6 <res 00000004>     <1> First_Path_Pos: resd 1    ; DIR.ASM ; 09/10/2011
210 000193AA <res 00000004>     <1> Last_Slash_Pos: resd 1    ; DIR.ASM
211                             <1>
212                             <1> ; 10/02/2016
213 000193AE <res 00000002>     <1> File_Count:     resw 1    ; DIR.ASM ; 09/10/2011
214 000193B0 <res 00000002>     <1> Dir_Count:      resw 1
215 000193B2 <res 00000004>     <1> Total_FSize:    resd 1
216 000193B6 <res 00000004>     <1> TFS_Dec_Begin:  resd 1
217 000193BA <res 0000000A>     <1>                 resb 10
218 000193C4 <res 00000001>     <1> TFS_Dec_End:    resb 1
219                             <1>
220 000193C5 <res 00000001>     <1> PrintDir_RowCounter: resb 1
221                             <1>
222 000193C6 <res 00000002>     <1> alignb 4
223                             <1> ; 15/02/2015 ('show' command variables)
224 000193C8 <res 00000004>     <1> Show_FDT:    resd 1
225 000193CC <res 00000004>     <1> Show_LDDDT:  resd 1
226 000193D0 <res 00000004>     <1> Show_Cluster:      resd 1
227 000193D4 <res 00000004>     <1> Show_FileSize:     resd 1
228 000193D8 <res 00000004>     <1> Show_FilePointer: resd 1
229 000193DC <res 00000002>     <1> Show_ClusterPointer: resw 1
230 000193DE <res 00000002>     <1> Show_ClusterSize: resw 1
231 000193E0 <res 00000001>     <1> Show_RowCount:     resb 1
232                             <1>
233 000193E1 <res 00000003>     <1> alignb 4
234                             <1> ; 21/02/2016
235 000193E4 <res 00000004>     <1> DelFile_FNPointer: resd 1 ; ; CMD_INTR.ASM (word) ; 09/11/2011
236                             <1> ; 27/02/2016
237                             <1> ; DIR.ASM (09/10/2011)
238 000193E8 <res 00000004>     <1> DelFile_FCluster:  resd 1
239 000193EC <res 00000002>     <1> DelFile_EntryCounter:     resw 1
240 000193EE <res 00000001>     <1> DelFile_LNEL:             resb 1
241 000193EF <res 00000001>     <1> resb 1
242                             <1>
243                             <1> ; DIR.ASM
244 000193F0 <res 00000004>     <1> mkdir_DirName_Offset:     resd 1
245 000193F4 <res 00000004>     <1> mkdir_FFCluster:   resd 1
246 000193F8 <res 00000004>     <1> mkdir_LastDirCluster:     resd 1
247 000193FC <res 00000004>     <1> mkdir_FreeSectors: resd 1
248 00019400 <res 00000002>     <1> mkdir_attrib:             resw 1
249 00019402 <res 00000001>     <1> mkdir_SecPerClust: resb 1
250 00019403 <res 00000001>     <1> mkdir_add_new_cluster:    resb 1
251 00019404 <res 0000000D>     <1> mkdir_Name:        resb 13
252 00019411 <res 00000002>     <1> resw 1 ; 01/03/2016
253                             <1> ; 27/02/2016
254 00019413 <res 00000001>     <1> RmDir_MultiClusters:      resb 1
255 00019414 <res 00000004>     <1> RmDir_DirEntryOffset:     resd 1 ; 01/03/2016 (word -> dword)
256 00019418 <res 00000004>     <1> RmDir_ParentDirCluster: resd 1
257 0001941C <res 00000004>     <1> RmDir_DirLastCluster:   resd 1
258 00019420 <res 00000004>     <1> RmDir_PreviousCluster:  resd 1
259                             <1> ; 22/02/2016
260 00019424 <res 00000001>     <1> UPDLMDT_CDirLevel: resb 1
261 00019425 <res 00000004>     <1> UPDLMDT_CDirFCluster:     resd 1
262                             <1>
263 00019429 <res 00000003>     <1> alignb 4
264                             <1> ; DRV_FAT.ASM ; 21/08/2011
265 0001942C <res 00000004>     <1> gffc_next_free_cluster:  resd 1
266 00019430 <res 00000004>     <1> gffc_first_free_cluster: resd 1
267 00019434 <res 00000004>     <1> gffc_last_free_cluster:  resd 1
268                             <1>
269                             <1> ;29/04/2016
270                             <1> Cluster_Index: ; resd 1
271                             <1> ; 22/02/2016
```

```
00019438 <res 00000004>      <1> ClusterValue:      resd 1
                             <1> ; 04/03/2016
0001943C <res 00000001>      <1> Attributes: resb 1
                             <1> ;;CFS_error:  resb 1 ;; 01/03/2016
0001943D <res 00000001>      <1> resb 1
0001943E <res 00000001>      <1> CFS_OPType: resb 1
0001943F <res 00000001>      <1> CFS_Drv:    resb 1
00019440 <res 00000004>      <1> CFS_CC:         resd 1
00019444 <res 00000004>      <1> CFS_FAT32FSINFOSEC: resd 1
00019448 <res 00000004>      <1> CFS_FAT32FC: resd 1
                             <1>
                             <1> ; 27/02/2016
                             <1> ;alignb 4
0001944C <res 00000004>      <1> glc_prevcluster: resd 1 ; DRV_FAT.ASM (21/08/2011)
                             <1> ; 22/10/2016
00019450 <res 00000004>      <1> glc_index:   resd 1 ;  Last Cluster Index (22/10/2016)
                             <1>
                             <1> ; DIR.ASM
00019454 <res 00000002>      <1> DLN_EntryNumber: resw 1
00019456 <res 00000001>      <1> DLN_40h:     resb 1
                             <1> ; 28/02/2016
00019457 <res 00000001>      <1> TCC_FATErr:  resb 1 ; DRV_FAT.ASM
                             <1>
                             <1> alignb 4
                             <1> ; DIR.ASM (09/10/2011)
00019458 <res 00000002>      <1> LCDE_EntryIndex: resw 1 ; LCDE_EntryOffset
0001945A <res 00000002>      <1> LCDE_ClusterSN:  resw 1
0001945C <res 00000004>      <1> LCDE_Cluster:       resd 1
00019460 <res 00000004>      <1> LCDE_ByteOffset: resd 1
                             <1>
                             <1> ;alignb4
                             <1> ; 06/03/2016 (word -> dword)
                             <1> ; CMD_INTR.ASM (01/08/2010)
00019464 <res 00000004>      <1> SourceFilePath:         resd 1
00019468 <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)
0001946C <res 00000001>      <1> SourceFile_Drv:                 resb 1
0001946D <res 00000041>      <1> SourceFile_Directory:           resb 65
000194AE <res 0000000D>      <1> SourceFile_Name:          resb 13
                             <1> SourceFile_LongNameEntryLength:
000194BB <res 00000001>      <1> SourceFile_LongNameYes:         resb 1 ; Sign for longname procedures
                             <1> ;Above 80 bytes
                             <1> ;is TR-DOS Source File FullName Format/Structure
000194BC <res 00000002>      <1> SourceFile_AttributesMask:      resw 1
000194BE <res 00000020>      <1> SourceFile_DirEntry:            resb 32
000194DE <res 00000004>      <1> SourceFile_DirFirstCluster:     resd 1
000194E2 <res 00000004>      <1> SourceFile_DirCluster:          resd 1
000194E6 <res 00000002>      <1> SourceFile_DirEntryNumber:      resw 1
000194E8 <res 00000002>      <1> SourceFile_MatchCounter:  resw 1
                             <1> ; 16/03/2016
000194EA <res 00000001>      <1> SourceFile_SecPerClust:         resb 1
000194EB <res 00000001>      <1> SourceFile_Reserved:            resb 1
                             <1> ; Above is 128 bytes
                             <1>
                             <1> ;Destination File Structure (same with 'Find File' Structure)
000194EC <res 00000001>      <1> DestinationFile_Drv:            resb 1
000194ED <res 00000041>      <1> DestinationFile_Directory:      resb 65
0001952E <res 0000000D>      <1> DestinationFile_Name:           resb 13
                             <1> DestinationFile_LongNameEntryLength:
0001953B <res 00000001>      <1> DestinationFile_LongNameYes:    resb 1 ; Sign for longname procedures
                             <1> ;Above 80 bytes
                             <1> ;is TR-DOS Destination File FullName Format/Structure
0001953C <res 00000002>      <1> DestinationFile_AttributesMask: resw 1
0001953E <res 00000020>      <1> DestinationFile_DirEntry: resb 32
0001955E <res 00000004>      <1> DestinationFile_DirFirstCluster: resd 1
00019562 <res 00000004>      <1> DestinationFile_DirCluster:     resd 1
00019566 <res 00000002>      <1> DestinationFile_DirEntryNumber: resw 1
00019568 <res 00000002>      <1> DestinationFile_MatchCounter:   resw 1
                             <1> ; 16/03/2016
0001956A <res 00000001>      <1> DestinationFile_SecPerClust:    resb 1
0001956B <res 00000001>      <1> DestinationFile_Reserved: resb 1
                             <1> ; Above is 128 bytes
                             <1>
                             <1> ; 24/04/2016
0001956C <res 00000002>      <1> resw 1
                             <1>
                             <1> ; 10/03/2016
                             <1> ; FILE.ASM
0001956E <res 00000001>      <1> move_cmd_phase:       resb 1
0001956F <res 00000001>      <1> msftdf_sf_df_drv:  resb 1
00019570 <res 00000004>      <1> msftdf_drv_offset: resd 1
                             <1>
                             <1> ; 11/03/2016
                             <1> ; DRV_FAT.ASM (21/08/2011)
00019574 <res 00000004>      <1> FAT_anc_LCluster:  resd 1
00019578 <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'
0001957C <res 00000004>      <1> mem_ipg_count:     resd 1 ; page count (for contiguous allocation)
00019580 <res 00000004>      <1> mem_pg_count:      resd 1 ; page count (for count down)
00019584 <res 00000004>      <1> mem_aperture:      resd 1 ; contiguous free pages (current)
00019588 <res 00000004>      <1> mem_max_aperture: resd 1 ; maximum value of contiguous free pages
0001958C <res 00000004>      <1> mem_pg_pos:  resd 1 ; mem. position (page #) of current aperture
00019590 <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')
```

```
377 00019594 <res 00000001>      <1> copy_cmd_phase:       resb 1
378 00019595 <res 00000001>      <1> csftdf_rw_err:           resb 1
379 00019596 <res 00000001>      <1> DestinationFileFound: resb 1
380 00019597 <res 00000001>      <1> csftdf_cdrv:             resb 1
381 00019598 <res 00000004>      <1> csftdf_filesize:      resd 1
382                              <1> ; TRDOS386 (TRDOS v2.0)
383 0001959C <res 00000004>      <1> csftdf_sf_mem_addr:   resd 1
384 000195A0 <res 00000004>      <1> csftdf_sf_mem_bsize:  resd 1
385                              <1> ;
386                              <1>
387 000195A4 <res 00000004>      <1> csftdf_sf_cluster:    resd 1 ; 16/03/2016
388 000195A8 <res 00000004>      <1> csftdf_df_cluster:    resd 1
389                              <1> ; 16/03/2016
390 000195AC <res 00000004>      <1> csftdf_r_size:        resd 1
391 000195B0 <res 00000004>      <1> csftdf_w_size:        resd 1
392 000195B4 <res 00000004>      <1> csftdf_sf_rbytes:     resd 1
393 000195B8 <res 00000004>      <1> csftdf_df_wbytes:     resd 1
394 000195BC <res 00000001>      <1> csftdf_percentage:    resb 1
395                              <1> ; 17/03/2016
396 000195BD <res 00000001>      <1> csftdf_videopage:     resb 1
397 000195BE <res 00000002>      <1> csftdf_cursorpos:     resw 1
398 000195C0 <res 00000004>      <1> csftdf_sf_drv_dt:     resd 1
399 000195C4 <res 00000004>      <1> csftdf_df_drv_dt:     resd 1
400                              <1>
401                              <1> ; 21/03/2016
402                              <1> ; 20/03/2016
403                              <1> ; FILE.ASM
404 000195C8 <res 00000004>      <1> createfile_Name_Offset:  resd 1
405 000195CC <res 00000004>      <1> createfile_FreeSectors:  resd 1
406 000195D0 <res 00000004>      <1> createfile_size:         resd 1
407 000195D4 <res 00000004>      <1> createfile_FFCluster:    resd 1 ; 11/03/2016
408 000195D8 <res 00000004>      <1> createfile_LastDirCluster: resd 1
409 000195DC <res 00000004>      <1> createfile_Cluster:      resd 1
410 000195E0 <res 00000004>      <1> createfile_PCluster:     resd 1
411 000195E4 <res 00000001>      <1> createfile_attrib:  resb 1
412 000195E5 <res 00000001>      <1> createfile_SecPerClust:  resb 1
413 000195E6 <res 00000002>      <1> createfile_DirIndex:     resw 1
414 000195E8 <res 00000004>      <1> createfile_CCount:  resd 1
415 000195EC <res 00000002>      <1> createfile_BytesPerSec:    resw 1 ; 23/03/2016
416 000195EE <res 00000001>      <1> createfile_wfc:              resb 1
417 000195EF <res 00000001>      <1> createfile_UpdatePDir:     resb 1 ; 31/03/2016
418                              <1>
419                              <1> ;alignb 4
420                              <1>
421                              <1> ; 11/04/2016
422 000195F0 <res 00000002>      <1> env_var_length:       resw 1
423                              <1>
424 000195F2 <res 00000002>      <1> alignb 4
425                              <1>
426                              <1> ; 25/04/2016
427 000195F4 <res 00000001>      <1> readi.valid: resb 1 ; valid data (>0 = valid for readi)
428 000195F5 <res 00000001>      <1> readi.drv:   resb 1 ; drive number (0, 1,2,3,4..)
429 000195F6 <res 00000001>      <1> readi.spc:   resb 1 ; sectors per cluster for 'readi' drive
430 000195F7 <res 00000001>      <1> readi.s_index:  resb 1 ; sector index in current cluster (buffer)
431 000195F8 <res 00000004>      <1> readi.sector:      resd 1 ; current disk sector
432 000195FC <res 00000002>      <1> readi.bpc:   resw 1 ; bytes per cluster - 1
433 000195FE <res 00000002>      <1> readi.offset:      resw 1 ; byte offset in cluster buffer
434 00019600 <res 00000004>      <1> readi.cluster:  resd 1 ; current cluster number
435 00019604 <res 00000004>      <1> readi.c_index:     resd 1 ; cluster index of the current cluster (0,1,2,3..)
436 00019608 <res 00000004>      <1> readi.fclust:      resd 1 ; first cluster of the current cluster
437 0001960C <res 00000004>      <1> readi.fs_index: resd 1 ; sector index in disk/file section (for Singlix FS)
438                              <1> ;readi.buffer:     resd 1 ; readi sector buffer address
439                              <1>
440                              <1> ;alignb 4
441                              <1>
442 00019610 <res 00000001>      <1> writei.valid:      resb 1 ; valid data (>0 = valid for writei)
443 00019611 <res 00000001>      <1> writei.drv:  resb 1 ; drive number (0, 1,2,3,4..)
444 00019612 <res 00000001>      <1> writei.spc:  resb 1 ; sectors per cluster for 'writei' drive
445 00019613 <res 00000001>      <1> writei.s_index: resb 1 ; sector index in current cluster (buffer)
446 00019614 <res 00000004>      <1> writei.sector:     resd 1 ; current disk sector
447 00019618 <res 00000002>      <1> writei.bpc:  resw 1 ; bytes per cluster - 1
448 0001961A <res 00000002>      <1> writei.offset:     resw 1 ; byte offset in cluster buffer
449 0001961C <res 00000004>      <1> writei.cluster: resd 1 ; current cluster number
450 00019620 <res 00000004>      <1> writei.c_index:    resd 1 ; cluster index of the current cluster (0,1,2,3..)
451 00019624 <res 00000004>      <1> writei.fclust:  resd 1 ; first cluster of the current cluster
452 00019628 <res 00000004>      <1> writei.fs_index: resd 1 ; sector index in disk/file section (for Singlix FS)
453                              <1> ;writei.buffer:    resd 1 ; writei sector buffer address
454 0001962C <res 00000004>      <1> writei.lclust:     resd 1 ; writei last cluster (mget_w) ; 23/10/2016
455 00019630 <res 00000004>      <1> writei.l_index:    resd 1 ; writei last cluster index (mget_w) ; 23/10/2016
456 00019634 <res 00000001>      <1> writei.ofn:  resb 1 ; open file number (to be written) ; 23/10/2016
457                              <1>
458 00019635 <res 00000003>      <1> alignb 4
459                              <1>
460                              <1> ; 29/04/2016
461 00019638 <res 00000004>      <1> Run_CDirFC:  resd 1
462 0001963C <res 00000001>      <1> Run_Auto_Path:     resb 1
463 0001963D <res 00000001>      <1> Run_Manual_Path: resb 1 ; 0 -> auto path sequence needed
464 0001963E <res 00000001>      <1> EXE_ID:            resb 1
465 0001963F <res 00000001>      <1> EXE_dot:    resb 1
466                              <1>
467                              <1> ; 06/05/2016
468 00019640 <res 00000004>      <1> mainprog_return_addr: resd 1
469 00019644 <res 00000004>      <1> last_error:  resd 1  ; this will be used to return error code to MainProg
470                              <1>                      ; 'lasterror' keyword will be used later to get the
471                              <1>                      ; last error code/number/status.
472                              <1> ; 12/05/2016
473 00019648 <res 00000004>      <1> video_eax:   resd 1  ; eax return value of video function
474                              <1>
475                              <1> ; 01/06/2016
476 0001964C <res 00000004>      <1> user_buffer: resd 1  ; 'diskio.s' (INT 33h, Function 08h, floppy disk type)
477                              <1>
478                              <1> ; 21/05/2016 - TRDOS 386 ('swap/switch', 'rswap', [u.pri])
479 00019650 <res 00000001>      <1> priority:    resb 1  ; running priority level of process (0,1,2)
480                              <1>                      ; (run queue which is process comes from)
481                              <1> ; 22/05/2016 - TRDOS 386 ('set_run_sequence', 'rtc_int', 'u_timer')
```

```
00019651 <res 00000001>      <1> p_change:    resb 1  ; process change status (for timer events)
                             <1> ; 23/05/2016 - TRDOS 386 ('clock')
00019652 <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
00019653 <res 00000001>      <1> timer_events:      resb 1  ; number of (active) timer events, <= 16
                             <1>
                             <1> ; 24/06/2016
00019654 <res 00000001>      <1> w_str_cmd:   resb 1 ; WRITE_STRING command (0,1,2,3) ; video.s
00019655 <res 00000001>      <1> p_crt_mode:  resb 1  ; previous video mode (=3 or 0), backup mark/sign
                             <1> ; 26/06/2016
00019656 <res 00000001>      <1> p_crt_page:  resb 1  ; previous active page (for 'set_mode')
                             <1> ; 04/07/2016
00019657 <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
00019658 <res 00000002>      <1> CRT_LEN:     resw 1  ; length of regen buffer in bytes
0001965A <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)
0001966A <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
0001966E <res 00000001>      <1> VGA_MTYPE:   resb 1  ; 0=CTEXT,1=MTEXT,2=CGA,3=PLANAR1,4=PLANAR4,5=LINEAR
                             <1>
                             <1> ; 23/10/2016
0001966F <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
00019670 <res 00000004>      <1> FFF_UBuffer: resd 1  ; User's buffer address for FFF & FNF system calls
                             <1> ; 15/10/2016
00019674 <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)
00019675 <res 00000001>      <1> FFF_Attrib:  resb 1 ; Find First File attributes for Find Next File (LB)
00019676 <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)
00019677 <res 00000001>      <1> SWP_inv_fname:     resb 1 ; Set Working Path - Invalid File Name
00019678 <res 00000002>      <1> SWP_Mode:    resw 1 ; Set Working Path - Mode
0001967A <res 00000001>      <1> SWP_DRV:     resb 1 ; Set Working Path - Drive
0001967B <res 00000001>      <1> SWP_DRV_chg: resb 1 ; Set Working Path - Drive Change
                             <1>
                             <1> ; 27/02/2017
0001967C <res 00000001>      <1> fpready:     resb 1 ; '80387 fpu is ready' flag
                             <1>
                             <1> ; 08/10/2016
0001967D <res 00000009>      <1> device_name:    resb 9  ; capitalized (and zero padded) device canem
                             <1>                    ; (example: "TTY0",0,0,0,0,0")
                             <1>
00019686 <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)
00019688 <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>
000196A8 <res 00000040>         <1> IDEV_NAME:  resb 8*NUMIDEV
                                <1>                     ; 8 byte names of installable device drivers
                                <1>
000196E8 <res 00000008>         <1> IDEV_TYPE:  resb NUMIDEV ; Driver type of installable device drivers
000196F0 <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.)
000196F8 <res 00000020>         <1> IDEV_OADDR: resd NUMIDEV ; open function addr for installable dev driver
00019718 <res 00000020>         <1> IDEV_CADDR: resd NUMIDEV ; close function addr for installable dev driver
00019738 <res 00000020>         <1> IDEV_RADDR: resd NUMIDEV ; read function addr for installable dev driver
00019758 <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
00019778 <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
00019796 <res 0000001E>         <1> DEV_R_OWNER:  resb NUMOFDEVICES   ; Reading owner no (u.uid) of devices
000197B4 <res 0000001E>         <1> DEV_R_OPENCOUNT: resb NUMOFDEVICES   ; Reading open count
000197D2 <res 0000001E>         <1> DEV_W_OWNER:  resb NUMOFDEVICES   ; Writing owner no (u.uid) of devices
000197F0 <res 0000001E>         <1> DEV_W_OPENCOUNT: resb NUMOFDEVICES   ; Writing open count
0001980E <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
0001982C <res 0000001E>         <1> DEV_OPENMODE:       resb NUMOFDEVICES   ; 1 = read mode
                                <1>                                   ; 2 = write mode
                                <1>                                   ; 3 = read & write
                                <1>                                   ; 0 = not open (free)
0001984A <res 00000078>         <1> DEV_NAME_PTR:    resd NUMOFDEVICES   ; pointers to name addresses of drivers
                                <1>                                   ; Address base: KDEV_NAME+
                                <1>                                   ; or IDEV_NAME+
000198C2 <res 00000078>         <1> DEV_R_POINTER:      resd NUMOFDEVICES   ; reading pointer, writing pointer
0001993A <res 00000078>         <1> DEV_W_POINTER:      resd NUMOFDEVICES   ; sector number if block device
                                <1>                                   ; character offset if char device
000199B2 <res 00000002>         <1> alignb 4
                                <1>
                                <1> ; 06/10/2016
                                <1> ; Open File Parameters
000199B4 <res 00000028>         <1> OF_FCLUSTER: resd OPENFILES  ; First clusters of open files
000199DC <res 0000000A>         <1> OF_DRIVE:    resb OPENFILES  ; Logical DOS drive numbers of open files
000199E6 <res 0000000A>         <1> OF_MODE:     resb OPENFILES  ; Open mode (1 = read, 2 = write, 3 = r&w)
000199F0 <res 0000000A>         <1> OF_STATUS:   resb OPENFILES  ; (bit 0 = read, bit 1 = write)
000199FA <res 0000000A>         <1> OF_OPENCOUNT:      resb OPENFILES  ; Open counts of open files
00019A04 <res 00000028>         <1> OF_POINTER:  resd OPENFILES      ; File seek/read/write pointer
00019A2C <res 00000028>         <1> OF_SIZE:     resd OPENFILES      ; File sizes of open files (in bytes)
00019A54 <res 00000028>         <1> OF_DIRFCLUSTER:    resd OPENFILES  ; Directory First Clusters of open files
00019A7C <res 00000028>         <1> OF_DIRCLUSTER:     resd OPENFILES  ; Directory (Entry) Clusters of open files
00019AA4 <res 00000028>         <1> OF_VOLUMEID: resd OPENFILES  ; Vol ID for removable drives of open files
00019ACC <res 00000028>         <1> OF_CCLUSTER: resd OPENFILES  ; Current clusters of open files
00019AF4 <res 00000028>         <1> OF_CCINDEX:  resd OPENFILES  ; Cluster index numbers of current clusters
                                <1> ; 24/10/2016
00019B1C <res 00000014>         <1> OF_DIRENTRY: resw OPENFILES  ; Directory entry index no. in dir cluster
                                <1>                              ; Sector index = entry index / 16
                                <1> ;alignb 2
                                <1>
00019B30 <res 00000060>         <1> DTA:         resd 24            ; Find First File data transfer area
                                <1>
                                <1> ; 19/12/2016
00019B90 <res 00000001>         <1> tcallback:   resb 1       ; Timer callback method flag for 'systimer'
00019B91 <res 00000001>         <1> trtc:        resb 1       ; Timer interrupt type flag for 'systimer'
                                <1> ; 20/02/2017
00019B92 <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>
00019B93 <res 00000001>         <1> alignb 4
                                <1> ; 13/04/2017
00019B94 <res 0000001E>         <1> DEV_INTR:   resb NUMOFDEVICES ; Device Interrupt (IRQ) number + 1
                                <1>                          ; (0= not available, 1= IRQ 0, 16= IRQ 15)
00019BB2 <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
00019BF2 <res 00000009>         <1> IRQ.owner:   resb 9       ; owner, 0 = free, >0 = [u.uno]
00019BFB <res 00000009>         <1> IRQ.dev:     resb 9       ; 0 = default/kernel, >0 = device number
00019C04 <res 00000009>         <1> IRQ.method:  resb 9             ; 0 = Signal Response Byte, 1 = Callback
00019C0D <res 00000009>         <1> IRQ.srb:     resb 9             ; Signal Response/Return Byte value
00019C16 <res 00000024>         <1> IRQ.addr:    resd 9       ; Rignal Response Byte address (physical)
                                <1>                           ; or Callback service address (virtual)
                                <1> ; 28/02/2017
00019C3A <res 00000004>         <1> IRQ_cr3:     resd 1       ; for saving cr3 register in IRQ handler
00019C3E <res 00000001>         <1> IRQnum:            resb 1       ; IRQ number for IRQ handler (trdosk8.s)
                                <1>
```

```
692                                     <1> ; 10/04/2017
693                                     <1> ; 03/04/2017
694                                     <1> ; UNINITIALIZED AUDIO DATA
695 00019C3F <res 00000001>             <1> alignb 4
696 00019C40 <res 00000001>             <1> audio_pci:   resb 1
697 00019C41 <res 00000001>             <1> audio_device:       resb 1
698 00019C42 <res 00000001>             <1> audio_mode:  resb 1
699 00019C43 <res 00000001>             <1> audio_intr:  resb 1
700 00019C44 <res 00000001>             <1> audio_busy:  resb 1  ; Busy flag for audio irq ; 21/04/2017
701 00019C45 <res 00000001>             <1> audio_reserved: resb 1
702 00019C46 <res 00000002>             <1> audio_io_base:      resw 1       ; Base I/O address of audio device
703 00019C48 <res 00000004>             <1> audio_dev_id:       resd 1 ; BUS/DEV/FN ; 00000000BBBBBBBBDDDDDFFF00000000
704 00019C4C <res 00000004>             <1> audio_vendor:       resd 1
705 00019C50 <res 00000004>             <1> audio_stats_cmd: resd 1
706                                     <1> ;
707 00019C54 <res 00000004>             <1> audio_buffer:       resd 1 ; virtual address of user's audio buffer
708 00019C58 <res 00000004>             <1> audio_p_buffer:     resd 1 ; Physical address of user's audio buffer
709 00019C5C <res 00000004>             <1> audio_buff_size: resd 1 ; user's audio buffer size (half buffer size)
710 00019C60 <res 00000004>             <1> audio_dma_buff: resd 1  ; dma buffer address
711 00019C64 <res 00000004>             <1> audio_dmabuff_size: resd 1 ; dma buffer size (2 * half buffer size)
712 00019C68 <res 00000001>             <1> audio_flag:  resb 1  ; dma buffer flag (1st half = 0, 2nd half = 1)
713 00019C69 <res 00000001>             <1> audio_user:  resb 1 ; user number of the owner
714 00019C6A <res 00000001>             <1> audio_cb_mode:      resb 1 ; 0 = signal response byte method
715                                     <1>                     ; 1 = callback method
716                                     <1>                     ; 2 = s.r.b. method with auto increment
717 00019C6B <res 00000001>             <1> audio_srb:   resb 1 ; signal response byte value
718 00019C6C <res 00000004>             <1> audio_cb_addr:      resd 1  ; callback service address or s.r.b. address
719                                     <1>                     ; (s.r.b. addr is physical, cbs addr is virtual)
720                                     <1>
721 00019C70 <res 00000001>             <1> audio_bps:   resb 1  ; selected mode: 8 bit, 16 bit
722 00019C71 <res 00000001>             <1> audio_stmo:  resb 1 ; selected mode: mono /stereo
723 00019C72 <res 00000002>             <1> audio_freq:  resw 1 ; sampling rate
724                                     <1>
725                                     <1> ; 21/04/2017
726 00019C74 <res 00000001>             <1> audio_play_cmd: resb 1  ; Play/Stop command (1 = play, 0 = stop)
727                                     <1> audio_civ: ; 28/05/2017 ; Current Buffer Index (AC'97)
728 00019C75 <res 00000001>             <1> audio_flag_eol:     resb 1  ; End of Link status (vt8233, EOL/FLAG)
729                                     <1>
730                                     <1> audio_master_volume:
731 00019C76 <res 00000001>             <1> audio_master_volume_l: resb 1 ; sound volume (lineout) left channel
732 00019C77 <res 00000001>             <1> audio_master_volume_r: resb 1 ; sound volume (lineout) right channel
733                                     <1>
734                                     <1> alignb 4
735                                     <1> ; 28/05/2017
736                                     <1> ; AC'97 Audio Controller Base Adress Registers
737 00019C78 <res 00000002>             <1> NAMBAR:             resw 1 ; Native Audio Mixer Base Address
738 00019C7A <res 00000002>             <1> NABMBAR:    resw 1 ; Native Audio Bus Mastering Base Address
739                                     <1>
740                                     <1> ;alignb 4
741                                     <1> ; 21/04/2017
742 00019C7C <res 00000400>             <1> audio_bdl_buff:     resd 32*8  ; VT8233 (AC97) BDL Buffer Size
743                                     <1> ; 12/05/2017
744 0001A07C <res 00000004>             <1> base_addr:  resd 1 ; 'direct_memory_access' (memory.s)
745                                     <1>
746                                     <1> ; 28/08/2017
747                                     <1> ; 20/08/2017
748 0001A080 <res 00000001>             <1>             resb 1  ;
749 0001A081 <res 00000001>             <1> dma_user:   resb 1 ; user number for sysdma
750 0001A082 <res 00000001>             <1> dma_channel: resb 1 ; dma channel for sysdma
751 0001A083 <res 00000001>             <1> dma_mode:   resb 1  ; dma mode for sysdma
752 0001A084 <res 00000004>             <1> dma_addr:   resd 1 ; dma buffer physical addr for sysdma
753 0001A088 <res 00000004>             <1> dma_size:   resd 1  ; dma buffer size (in bytes) for sysdma
754 0001A08C <res 00000004>             <1> dma_start:  resd 1  ; dma start address for sysdma
755 0001A090 <res 00000004>             <1> dma_count:  resd 1  ; dma count (in bytes) for sysdma
756                                     <1>
757 0001A094 <res 00005F6C>             <1> alignb 65536
758                                     <1> ; 09/08/2017
759                                     <1> ; 12/05/2017
760 00020000 <res 00010000>             <1> sb16_dma_buffer: resb 65536 ; DMA buffer for sb16 audio playing.
3636                                        ; 24/01/2016
3637                                        %include 'ubss.s'  ; UNINITIALIZED KERNEL (USER) DATA
   1                                    <1> ; ****************************************************************************
   2                                    <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - UNINITIALIZED USER DATA : ubss.s
   3                                    <1> ; ----------------------------------------------------------------------------
   4                                    <1> ; Last Update: 28/02/2017
   5                                    <1> ; ----------------------------------------------------------------------------
   6                                    <1> ; Beginning: 24/01/2016
   7                                    <1> ; ----------------------------------------------------------------------------
   8                                    <1> ; Assembler: NASM version 2.11 (trdos386.s)
   9                                    <1> ; ----------------------------------------------------------------------------
  10                                    <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
  11                                    <1> ; ux.s (04/12/2015)
  12                                    <1> ; ****************************************************************************
  13                                    <1>
  14                                    <1> ; Retro UNIX 386 v1 Kernel - ux.s
  15                                    <1> ; Last Modification: 04/12/2015
  16                                    <1> ;
  17                                    <1> ; ///////// RETRO UNIX 386 V1 SYSTEM DEFINITIONS ///////////////
  18                                    <1> ; (Modified from
  19                                    <1> ;     Retro UNIX 8086 v1 system definitions in 'UNIX.ASM', 01/09/2014)
  20                                    <1> ; ((UNIX.ASM (RETRO UNIX 8086 V1 Kernel), 11/03/2013 - 01/09/2014))
  21                                    <1> ; ----------------------------------------------------------------------------
  22                                    <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
  23                                    <1> ; (Original) Source Code by Ken Thompson (1971-1972)
  24                                    <1> ; <Bell Laboratories (17/3/1972)>
  25                                    <1> ; <Preliminary Release of UNIX Implementation Document>
  26                                    <1> ; (Section E10 (17/3/1972) - ux.s)
  27                                    <1> ; ****************************************************************************
  28                                    <1>
  29                                    <1> alignb 2
  30                                    <1>
  31                                    <1> inode:
  32                                    <1>       ; 11/03/2013.
  33                                    <1>       ;Derived from UNIX v1 source code 'inode' structure (ux).
  34                                    <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 requlat' ; 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'
```

```
 245                                    <1>
 246                                    <1> ; 03/06/2015 - Retro UNIX 386 v1 Beginning
 247                                    <1> ; 07/04/2013 - 31/07/2013 - Retro UNIX 8086 v1
 248 00030450 <res 00000001>            <1> rw:    resb 1 ;; Read/Write sign (iget)
 249                                    <1>
 250                                    <1> ;alignb 4
 251                                    <1>
 252                                    <1> ; 24/04/2016
 253 00030451 <res 00000004>            <1> ii:         resd 1 ; first cluster of the program file
 254 00030455 <res 00000004>            <1> i.size:           resd 1 ; size of the program file
3638
3639 00030459 <res 00000003>                alignb 4
3640
3641                                        ; 23/05/2016 (TRDOS 386)
3642                                        ; 14/10/2015 (Retro UNIX 386 v1, 'unix386.s')
3643 0003045C <res 00000004>                cr3reg:      resd 1  ; cr3 register content at the beginning of the timer
3644                                                     ; (or RTC) interrupt handler.
3645
3646                                        ; 10/12/2016 (callback)
3647                                        ; 10/06/2016
3648                                        ; 19/05/2016
3649                                        ; 18/05/2016 - TRDOS 386 feature only !
3650 00030460 <res 00000100>                timer_set: resd 16*4   ; 256 bytes memory space for 16 timer events
3651                                              ; Timer Event Structure: (max. 16 timer events, 16*16 bytes)
3652                                              ;       Owner:              resb 1 ; 0 = free
3653                                              ;                          ;>0 = process number (u.uno)
3654                                              ;      Callback:    resb 1 ; 0 = response byte address (phy)
3655                                              ;                           1 = callback address (virtual)
3656                                              ;      Interrupt:      resb 1 ; 0 = Timer interrupt (or none)
3657                                              ;                          ; 1 = Real Time Clock interrupt
3658                                              ;      Response:       resb 1 ; 0 to 255, signal return value
3659                                              ;      Count Limit: resd 1 ; count of ticks (total/set)
3660                                              ;      Current Count:      resd 1 ; count of ticks (current)
3661                                              ;      Response Addr:  resd 1 ; response byte (pointer) address
3662                                              ;                          ; (or callback -user service- address)
3663
3664                                        ;; Memory (swap) Data (11/03/2015)
3665                                        ; 09/03/2015
3666 00030560 <res 00000002>                swpq_count: resw 1 ; count of pages on the swap queue
3667 00030562 <res 00000004>                swp_drv:    resd 1 ; logical drive description table address of the swap drive/disk
3668 00030566 <res 00000004>                swpd_size:  resd 1 ; size of swap drive/disk (volume) in sectors (512 bytes).

3669 0003056A <res 00000004>                swpd_free:  resd 1 ; free page blocks (4096 bytes) on swap disk/drive (logical)
3670 0003056E <res 00000004>                swpd_next:  resd 1 ; next free page block
3671 00030572 <res 00000004>                swpd_last:  resd 1 ; last swap page block
3672
3673 00030576 <res 00000002>                alignb 4
3674
3675                                        ; 10/07/2015
3676                                        ; 28/08/2014
3677 00030578 <res 00000004>                error_code: resd 1
3678                                        ; 29/08/2014
3679 0003057C <res 00000004>                FaultOffset:       resd 1
3680                                        ; 21/09/2015
3681 00030580 <res 00000004>                PF_Count:   resd 1 ; total page fault count
3682                                                           ; (for debugging - page fault analyze)
3683                                                   ; 'page_fault_handler' (memory.inc)
3684                                                   ; 'sysgeterr' (u9.s)
3685
3686                                        ; 29/04/2016 (TRDOS 386 = TRDOS v2.0)
3687                                        ; 22/08/2015 (Retro UNIX 386 v1)
3688                                        buffer:
3689 00030584 <res 00000008>                      resb   8
3690                                        readi_buffer:
3691 0003058C <res 00000200>                      resb   512
3692 0003078C <res 00000008>                      resb   8
3693                                        writei_buffer:
3694 00030794 <res 00000200>                      resb   512
3695                                        ; 24/10/2016
3696 00030994 <res 00000008>                      resb   8
3697                                        rw_buffer:
3698 0003099C <res 00000800>                      resb   2048  ; general purposed, r/w sector buffer
3699
3700                                        %if 1
3701                                        ; 17/01/2021
3702 0003119C <res 00000080>                edid_info:  resb 128 ; VESA EDID (monitor capabilities) info
3703                                        ; 28/11/2020
3704 0003121C <res 00000001>                pmi32:             resb 1 ; (>0) use VESA VBE3 protected mode calls
3705 0003121D <res 00000001>                vbe_mode_x:  resb 1  ; VESA VBE3 video bios mode set options
3706 0003121E <res 00000002>                video_mode:  resw 1 ; VESA VBE3 video mode (with option flags)
3707                                        ; 30/11/2020
3708 00031220 <res 00000004>                vbe3bios_addr:     resd 1 ; new (writable mem) address of VBE3 bios
3709                                        ; 02/12/2020
3710 00031224 <res 00000004>                pmid_addr:  resd 1 ; PMInfoBlock ('PMID') linear address
3711                                        ; 14/01/2021
3712                                        ; 06/12/2020       ; VESA VBE 3 video state
3713                                        ;vbe3stbufsize: resw 1    ; video regs/dac/bios state buffer size
3714                                        ;                  ; block size in bytes
3715                                        ; 16/01/2021
3716 00031228 <res 00000002>                vbe3stbsflags: resw 1     ; video regs/dac/bios state buffer size
3717                                        ;                  ; pointer flags for buffer state options
3718                                        %endif
3719
3720                                        %if 1
3721                                        ; 10/12/2020
3722                                        LFB_Info:
3723 0003122A <res 00000010>                             resb 16      ; Linear Frame Buffer info block
3724
3725                                        ;24/11/2020 - TRDOS 386 v2.0.3
3726                                        ; BOCHS/PLEX86 VESA VBE3 MODE INFO extension to TRDOS 386 v2 kernel
3727                                        MODE_INFO_LIST:
3728 0003123A <res 00000044>                             resb 68      ; mode + 66 byte VESA vbe3 mode info (4F01h)
3729                                        %endif
3730
3731                                        ; 05/01/2021
```

```
3732 0003127E <res 00000001>          ufont:              resb 1 ; (VGA graphics) user font flags
3733                                                      ; bit 7 - permission flag for int 31h
3734                                                      ; bit 4 - 8x16 user font ready/loaded flag
3735                                                      ; bit 3 - 8x8 user font ready/loaded flag
3736                                                      ; bit 1 - select 8x16 user font (sysvideo)
3737                                                      ; bit 0 - select 8x8 user font (sysvideo)
3738 0003127F <res 00000001>                      resb 1 ; 19/01/2021
3739                                  ; 17/01/2021
3740 00031280 <res 00000001>          srvsf:              resb 1 ; 'save restore video state' permission flag
3741                                  ; 18/01/2021
3742 00031281 <res 00000001>          srvso:              resb 1 ; video state buffer save/restore option
3743 00031282 <res 00000004>          VideoStateID:       resd 1 ; used to verify state saved by same prog
3744                                  ; 29/01/2021
3745 00031286 <res 00000002>          v_width:     resw 1 ; screen (display page) width
3746 00031288 <res 00000001>          v_ops:              resb 1 ; 'sysvideo' graphics data transfer option
3747 00031289 <res 00000001>          v_bpp:              resb 1 ; bits per pixels ('sysvideo')
3748 0003128A <res 00000004>          v_mem:              resd 1 ; video memory ('sysvideo')
3749 0003128E <res 00000004>          v_siz:              resd 1 ; video page size ('sysvideo')
3750 00031292 <res 00000004>          v_str:              resd 1 ; window start adress ('sysvideo')
3751 00031296 <res 00000004>          v_end:              resd 1 ; window end (end+1) adress ('sysvideo')
3752                                  ; 31/01/2021
3753                                  ; 01/01/2021
3754                                  ;maskbuff:     ;resd 1    ; user's bitmask buffer addr ('sysvideo')
3755 0003129A <res 00000004>          maskcolor:   resd 1 ; VGA/SVGA pixel mask color ('sysvideo')
3756                                  ; 02/02/2021
3757 0003129E <res 00000008>          buffer8:     resd 2 ; 8 bytes small buffer for 'sysvideo'
3758
3759                                  bss_end:
3760
3761                                  ; 27/12/2013
3762                                  _end:  ; end of kernel code
```