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

```
; ****************************************************************************
; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0
; ----------------------------------------------------------------------------
; Last Update: 22/10/2017
; ----------------------------------------------------------------------------
; Beginning: 04/01/2016
; ----------------------------------------------------------------------------
; Assembler: NASM version 2.11 (trdos386.s)
; ----------------------------------------------------------------------------
; Turkish Rational DOS
; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
;
; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
; unix386.s (03/01/2016)
;
; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
; TRDOS2.ASM (09/11/2011)
;
; Derived from 'IBM PC-XT-286' BIOS source code (1986)
; ****************************************************************************

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

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



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

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

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

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

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

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

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

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

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

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

```
198 0000006F 0D0A00                     db     0Dh, 0Ah, 0
199                                     ;
200
201                                 L0:
202                                 %include 'diskinit.s' ; 07/03/2015
  1                               <1> ; ****************************************************************************
  2                               <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - diskinit.s
  3                               <1> ; ----------------------------------------------------------------------------
  4                               <1> ; Last Update: 09/07/2016
  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> ; diskinit.inc (10/07/2015)
 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 - DISKINIT.INC
 20                               <1> ; Last Modification: 10/07/2015
 21                               <1>
 22                               <1> ; DISK I/O SYSTEM INITIALIZATION - Erdogan Tan (Retro UNIX 386 v1 project)
 23                               <1>
 24                               <1> ; //////// DISK I/O SYSTEM STRUCTURE INITIALIZATION ///////////////
 25                               <1>
 26                               <1>       ; 10/12/2014 - 02/02/2015 - dsectrm2.s
 27                               <1> ;L0:
 28                               <1>       ; 12/11/2014 (Retro UNIX 386 v1 - beginning)
 29                               <1>       ; Detecting disk drives... (by help of ROM-BIOS)
 30 00000072 BA7F00               <1>       mov    dx, 7Fh
 31                               <1> L1:
 32 00000075 FEC2                 <1>       inc    dl
 33 00000077 B441                 <1>       mov    ah, 41h ; Check extensions present
 34                               <1>                      ; Phoenix EDD v1.1 - EDD v3
 35 00000079 BBAA55               <1>       mov    bx, 55AAh
 36 0000007C CD13                 <1>       int    13h
 37 0000007E 721A                 <1>       jc     short L2
 38                               <1>
 39 00000080 81FB55AA             <1>       cmp    bx, 0AA55h
 40 00000084 7514                 <1>       jne    short L2
 41 00000086 FE06[F55C]           <1>       inc    byte [hdc]   ; count of hard disks (EDD present)
 42 0000008A 8816[F45C]           <1>         mov     [last_drv], dl  ; last hard disk number
 43 0000008E BB[785C]             <1>       mov    bx, hd0_type - 80h
 44 00000091 01D3                 <1>       add    bx, dx
 45 00000093 880F                 <1>       mov    [bx], cl ; Interface support bit map in CX
 46                               <1>                       ; Bit 0 - 1, Fixed disk access subset ready
 47                               <1>                       ; Bit 1 - 1, Drv locking and ejecting ready
 48                               <1>                       ; Bit 2 - 1, Enhanced Disk Drive Support
 49                               <1>                            ;            (EDD) ready (DPTE ready)
 50                               <1>                       ; Bit 3 - 1, 64bit extensions are present
 51                               <1>                            ;            (EDD-3)
 52                               <1>                       ; Bit 4 to 15 - 0, Reserved
 53 00000095 80FA83               <1>       cmp    dl, 83h       ; drive number < 83h
 54 00000098 72DB                 <1>       jb     short L1
 55                               <1> L2:
 56                               <1>       ; 23/11/2014
 57                               <1>       ; 19/11/2014
 58 0000009A 30D2                 <1>       xor    dl, dl  ; 0
 59                               <1>       ; 04/02/2016 (esi -> si)
 60 0000009C BE[F65C]             <1>       mov    si, fd0_type
 61                               <1> L3:
 62                               <1>       ; 14/01/2015
 63 0000009F 8816[F35C]           <1>       mov    [drv], dl
 64                               <1>       ;
 65 000000A3 B408                 <1>       mov    ah, 08h ; Return drive parameters
 66 000000A5 CD13                 <1>       int    13h
 67 000000A7 7210                 <1>       jc     short L4
 68                               <1>              ; BL = drive type (for floppy drives)
 69                               <1>              ; DL = number of floppy drives
 70                               <1>              ;
 71                               <1>              ; ES:DI = Address of DPT from BIOS
 72                               <1>              ;
 73 000000A9 881C                 <1>       mov    [si], bl ;  Drive type
 74                               <1>                       ; 4 = 1.44 MB, 80 track, 3 1/2"
 75                               <1>       ; 14/01/2015
 76 000000AB E8BC01               <1>       call   set_disk_parms
 77                               <1>       ; 10/12/2014
 78 000000AE 81FE[F65C]           <1>       cmp    si, fd0_type
 79 000000B2 7705                 <1>       ja     short L4
 80 000000B4 46                   <1>       inc    si ; fd1_type
 81 000000B5 B201                 <1>       mov    dl, 1
 82 000000B7 EBE6                 <1>       jmp    short L3
 83                               <1> L4:
 84                               <1>       ; Older BIOS (INT 13h, AH = 48h is not available)
 85 000000B9 B27F                 <1>       mov    dl, 7Fh
 86                               <1>       ; 24/12/2014 (Temporary)
 87 000000BB 803E[F55C]00         <1>       cmp    byte [hdc], 0 ; EDD present or not ?
 88 000000C0 0F879000             <1>         ja      L10       ; yes, all fixed disk operations
 89                               <1>                       ; will be performed according to
 90                               <1>                       ; present EDD specification
 91                               <1> L6:
 92 000000C4 FEC2                 <1>       inc    dl
 93 000000C6 8816[F35C]           <1>         mov     [drv], dl
 94 000000CA 8816[F45C]           <1>         mov     [last_drv], dl ; 14/01/2015
 95 000000CE B408                 <1>       mov    ah, 08h ; Return drive parameters
 96 000000D0 CD13                 <1>       int    13h    ; (conventional function)
 97 000000D2 0F828601             <1>         jc      L13       ; fixed disk drive not ready
```

```
 98 000000D6 8816[F55C]          <1>         mov     [hdc], dl ; number of drives
 99                              <1>       ;; 14/01/2013
100                              <1>       ;;push cx
101 000000DA E88D01              <1>       call   set_disk_parms
102                              <1>       ;;pop  cx
103                              <1>       ;
104                              <1>       ;;and  cl, 3Fh       ; sectors per track (bits 0-6)
105 000000DD 8A16[F35C]          <1>         mov     dl, [drv]
106 000000E1 BB0401              <1>       mov    bx, 65*4 ; hd0 parameters table (INT 41h)
107 000000E4 80FA80              <1>       cmp    dl, 80h
108 000000E7 7603                <1>       jna    short L7
109 000000E9 83C314              <1>       add    bx, 5*4       ; hd1 parameters table (INT 46h)
110                              <1> L7:
111 000000EC 31C0                <1>       xor    ax, ax
112 000000EE 8ED8                <1>       mov    ds, ax
113 000000F0 8B37                <1>         mov     si, [bx]
114 000000F2 8B4702              <1>         mov     ax, [bx+2]
115 000000F5 8ED8                <1>       mov    ds, ax
116 000000F7 3A4C0E              <1>         cmp     cl, [si+FDPT_SPT] ; sectors per track
117 000000FA 0F855A01            <1>         jne     L12 ; invalid FDPT
118 000000FE BF0000              <1>       mov    di, HD0_DPT
119 00000101 80FA80              <1>       cmp    dl, 80h
120 00000104 7603                <1>       jna    short L8
121 00000106 BF2000              <1>       mov    di, HD1_DPT
122                              <1> L8:
123                              <1>       ; 30/12/2014
124 00000109 B80090              <1>       mov    ax, DPT_SEGM
125 0000010C 8EC0                <1>       mov    es, ax
126                              <1>       ; 24/12/2014
127 0000010E B90800              <1>       mov    cx, 8
128 00000111 F3A5                <1>       rep    movsw  ; copy 16 bytes to the kernel's DPT location
129 00000113 8CC8                <1>       mov    ax, cs
130 00000115 8ED8                <1>       mov    ds, ax
131                              <1>       ; 02/02/2015
132 00000117 8A0E[F35C]          <1>         mov     cl, [drv]
133 0000011B 88CB                <1>       mov    bl, cl
134 0000011D B8F001              <1>       mov    ax, 1F0h
135 00000120 80E301              <1>       and    bl, 1
136 00000123 7406                <1>       jz     short L9
137 00000125 C0E304              <1>       shl    bl, 4
138 00000128 2D8000              <1>       sub    ax, 1F0h-170h
139                              <1> L9:
140 0000012B AB                  <1>       stosw  ; I/O PORT Base Address (1F0h, 170h)
141 0000012C 050602              <1>       add    ax, 206h
142 0000012F AB                  <1>       stosw  ; CONTROL PORT Address (3F6h, 376h)
143 00000130 88D8                <1>       mov    al, bl
144 00000132 04A0                <1>       add    al, 0A0h
145 00000134 AA                  <1>       stosb  ; Device/Head Register upper nibble
146                              <1>       ;
147 00000135 FE06[F35C]          <1>       inc    byte [drv]
148 00000139 BB[785C]            <1>       mov    bx, hd0_type - 80h
149 0000013C 01CB                <1>       add    bx, cx
150 0000013E 800F80              <1>         or      byte [bx], 80h  ; present sign (when lower nibble is 0)
151 00000141 A0[F55C]            <1>       mov    al, [hdc]
152 00000144 FEC8                <1>       dec    al
153 00000146 0F841201            <1>         jz      L13
154 0000014A 80FA80              <1>       cmp    dl, 80h
155 0000014D 0F8673FF            <1>         jna     L6
156 00000151 E90801              <1>         jmp     L13
157                              <1> L10:
158 00000154 FEC2                <1>       inc    dl
159                              <1>       ; 25/12/2014
160 00000156 8816[F35C]          <1>       mov    [drv], dl
161 0000015A B408                <1>       mov    ah, 08h ; Return drive parameters
162 0000015C CD13                <1>       int    13h    ; (conventional function)
163 0000015E 0F82FA00            <1>         jc      L13
164                              <1>       ; 14/01/2015
165 00000162 8A16[F35C]          <1>       mov    dl, [drv]
166 00000166 52                  <1>       push   dx
167 00000167 51                  <1>       push   cx
168 00000168 E8FF00              <1>       call   set_disk_parms
169 0000016B 59                  <1>       pop    cx
170 0000016C 5A                  <1>       pop    dx
171                              <1>       ; 06/07/2016 (BugFix for >64K kernel files)
172                              <1>       ; 04/02/2016 (esi -> si)
173                              <1>       ;mov   si, _end ; 30 byte temporary buffer address
174                              <1>       ;              ; at the '_end' of kernel.
175                              <1>       ;mov   word [si], 30
176                              <1>       ; 06/07/2016
177 0000016D BE[5000]            <1>       mov    si, _int13h_48h_buffer
178                              <1>       ; 09/07/2016
179 00000170 B81E00              <1>       mov    ax, 001Eh
180 00000173 8824                <1>       mov    [si], ah ; 0
181 00000175 46                  <1>       inc    si
182 00000176 8904                <1>       mov    word [si], ax
183                              <1>       ; word [si] = 30
184                              <1>       ;
185 00000178 B448                <1>       mov    ah, 48h       ; Get drive parameters (EDD function)
186 0000017A CD13                <1>       int    13h
187 0000017C 0F82DC00            <1>         jc      L13
188                              <1>       ; 04/02/2016 (ebx -> bx)
189                              <1>       ; 14/01/2015
190 00000180 28FF                <1>       sub    bh, bh
191 00000182 88D3                <1>       mov    bl, dl
192 00000184 80EB80              <1>       sub    bl, 80h
193 00000187 81C3[F85C]          <1>       add    bx, hd0_type
194 0000018B 8A07                <1>       mov    al, [bx]
195 0000018D 0C80                <1>       or     al, 80h
196 0000018F 8807                <1>       mov    [bx], al
197 00000191 81EB[F65C]          <1>       sub    bx, hd0_type - 2 ; 15/01/2015
198 00000195 81C3[425D]          <1>       add    bx, drv.status
199 00000199 8807                <1>       mov    [bx], al
```

```
                                <1>       ; 04/02/2016 (eax -> ax)
0000019B 8B4410                 <1>       mov    ax, [si+16]
0000019E 854412                 <1>       test   ax, [si+18]
000001A1 7412                   <1>       jz     short L10_A0h
                                <1>                    ; 'CHS only' disks on EDD system
                                <1>                    ;  are reported with ZERO disk size
000001A3 81EB[425D]             <1>       sub    bx, drv.status
000001A7 C1E302                 <1>       shl    bx, 2
000001AA 81C3[265D]             <1>       add    bx, drv.size ; disk size (in sectors)
000001AE 8907                   <1>       mov    [bx], ax
000001B0 8B4412                 <1>       mov    ax, [si+18]
000001B3 8907                   <1>       mov    [bx], ax
                                <1>
                                <1> L10_A0h: ; Jump here to fix a ZERO (LBA) disk size problem
                                <1>        ; for CHS disks (28/02/2015)
                                <1>       ; 30/12/2014
000001B5 BF0000                 <1>       mov    di, HD0_DPT
000001B8 88D0                   <1>       mov    al, dl
000001BA 83E003                 <1>       and    ax, 3
000001BD C0E005                 <1>       shl    al, 5 ; *32
000001C0 01C7                   <1>       add    di, ax
000001C2 B80090                 <1>       mov    ax, DPT_SEGM
000001C5 8EC0                   <1>       mov    es, ax
                                <1>       ;
000001C7 88E8                   <1>       mov    al, ch ; max. cylinder number (bits 0-7)
000001C9 88CC                   <1>       mov    ah, cl
000001CB C0EC06                 <1>       shr    ah, 6  ; max. cylinder number (bits 8-9)
000001CE 40                     <1>       inc    ax     ; logical cylinders (limit 1024)
000001CF AB                     <1>       stosw
000001D0 88F0                   <1>       mov    al, dh ; max. head number
000001D2 FEC0                   <1>       inc    al
000001D4 AA                     <1>       stosb         ; logical heads (limits 256)
000001D5 B0A0                   <1>       mov    al, 0A0h ; Indicates translated table
000001D7 AA                     <1>       stosb
000001D8 8A440C                 <1>       mov    al, [si+12]
000001DB AA                     <1>       stosb         ; physical sectors per track
000001DC 31C0                   <1>       xor    ax, ax
                                <1>       ;dec   ax      ; 02/01/2015
000001DE AB                     <1>       stosw         ; precompensation (obsolete)
                                <1>       ;xor   al, al  ; 02/01/2015
000001DF AA                     <1>       stosb         ; reserved
000001E0 B008                   <1>       mov    al, 8   ; drive control byte
                                <1>                      ; (do not disable retries,
                                <1>                     ; more than 8 heads)
000001E2 AA                     <1>       stosb
000001E3 8B4404                 <1>       mov    ax, [si+4]
000001E6 AB                     <1>       stosw         ; physical number of cylinders
                                <1>       ;push  ax      ; 02/01/2015
000001E7 8A4408                 <1>       mov    al, [si+8]
000001EA AA                     <1>       stosb         ; physical num. of heads (limit 16)
000001EB 29C0                   <1>       sub    ax, ax
                                <1>       ;pop   ax      ; 02/01/2015
000001ED AB                     <1>       stosw         ; landing zone (obsolete)
000001EE 88C8                   <1>       mov    al, cl  ; logical sectors per track (limit 63)
000001F0 243F                   <1>       and    al, 3Fh
000001F2 AA                     <1>       stosb
                                <1>       ;sub   al, al  ; checksum
                                <1>       ;stosb
                                <1>       ;
000001F3 83C61A                 <1>       add    si, 26   ; (BIOS) DPTE address pointer
000001F6 AD                     <1>       lodsw
000001F7 50                     <1>       push   ax      ; (BIOS) DPTE offset
000001F8 AD                     <1>       lodsw
000001F9 50                     <1>       push   ax      ; (BIOS) DPTE segment
                                <1>       ;
                                <1>       ; checksum calculation
000001FA 89FE                   <1>       mov    si, di
000001FC 06                     <1>       push   es
000001FD 1F                     <1>       pop    ds
                                <1>       ;mov   cx, 16
000001FE B90F00                 <1>       mov    cx, 15
00000201 29CE                   <1>       sub    si, cx
00000203 30E4                   <1>       xor    ah, ah
                                <1>       ;del   cl
                                <1> L11:
00000205 AC                     <1>       lodsb
00000206 00C4                   <1>       add    ah, al
00000208 E2FB                   <1>       loop   L11
                                <1>       ;
0000020A 88E0                   <1>       mov    al, ah
0000020C F6D8                   <1>       neg    al     ; -x+x = 0
0000020E AA                     <1>       stosb         ; put checksum in byte 15 of the tbl
                                <1>       ;
0000020F 1F                     <1>       pop    ds     ; (BIOS) DPTE segment
00000210 5E                     <1>       pop    si     ; (BIOS) DPTE offset
                                <1>       ;
                                <1>       ; 23/02/2015
00000211 57                     <1>       push   di
                                <1>       ; ES:DI points to DPTE (FDPTE) location
                                <1>       ;mov   cx, 8
00000212 B108                   <1>       mov    cl, 8
00000214 F3A5                   <1>       rep    movsw
                                <1>       ;
                                <1>       ; 23/02/2015
                                <1>       ; (P)ATA drive and LBA validation
                                <1>       ; (invalidating SATA drives and setting
                                <1>       ; CHS type I/O for old type fixed disks)
00000216 5B                     <1>       pop    bx
00000217 8CC8                   <1>       mov    ax, cs
00000219 8ED8                   <1>       mov    ds, ax
0000021B 268B07                 <1>       mov    ax, [es:bx]
0000021E 3DF001                 <1>       cmp    ax, 1F0h
```

```
302 00000221 7418              <1>       je    short L11a
303 00000223 3D7001            <1>       cmp   ax, 170h
304 00000226 7413              <1>       je    short L11a
305                            <1>       ; invalidation
306                            <1>       ; (because base port address is not 1F0h or 170h)
307 00000228 30FF              <1>       xor   bh, bh
308 0000022A 88D3              <1>       mov   bl, dl
309 0000022C 80EB80            <1>       sub   bl, 80h
310 0000022F C687[F85C]00      <1>       mov   byte [bx+hd0_type], 0 ; not a valid disk drive !
311 00000234 808F[445D]F0      <1>         or      byte [bx+drv.status+2], 0F0h ; (failure sign)
312 00000239 EB14              <1>       jmp   short L11b
313                            <1> L11a:
314                            <1>       ; LBA validation
315 0000023B 268A4704          <1>       mov   al, [es:bx+4] ; Head register upper nibble
316 0000023F A840              <1>       test  al, 40h ; LBA bit (bit 6)
317 00000241 750C              <1>       jnz   short L11b ; LBA type I/O is OK! (E0h or F0h)
318                            <1>       ; force CHS type I/O for this drive (A0h or B0h)
319 00000243 28FF              <1>       sub   bh, bh
320 00000245 88D3              <1>       mov   bl, dl
321 00000247 80EB80            <1>       sub   bl, 80h ; 26/02/2015
322 0000024A 80A7[445D]FE      <1>         and     byte [bx+drv.status+2], 0FEh ; clear bit 0
323                            <1>                           ; bit 0 = LBA ready bit
324                            <1>       ; 'diskio' procedure will check this bit !
325                            <1> L11b:
326 0000024F 3A16[F45C]        <1>       cmp   dl, [last_drv] ; 25/12/2014
327 00000253 7307              <1>         jnb     short L13
328 00000255 E9FCFE            <1>         jmp     L10
329                            <1> L12:
330                            <1>       ; Restore data registers
331 00000258 8CC8              <1>       mov   ax, cs
332 0000025A 8ED8              <1>       mov   ds, ax
333                            <1> L13:
334                            <1>       ; 13/12/2014
335 0000025C 0E                <1>       push  cs
336 0000025D 07                <1>       pop   es
337                            <1> L14:
338 0000025E B411              <1>       mov   ah, 11h
339 00000260 CD16              <1>       int   16h
340 00000262 7466              <1>       jz    short L16 ; no keys in keyboard buffer
341 00000264 B010              <1>       mov   al, 10h
342 00000266 CD16              <1>       int   16h
343 00000268 EBF4              <1>       jmp   short L14
344                            <1>
345                            <1> set_disk_parms:
346                            <1>       ; 04/02/2016 (ebx -> bx)
347                            <1>       ; 10/07/2015
348                            <1>       ; 14/01/2015
349                            <1>       ;push  bx
350 0000026A 28FF              <1>       sub   bh, bh
351 0000026C 8A1E[F35C]        <1>       mov   bl, [drv]
352 00000270 80FB80            <1>       cmp   bl, 80h
353 00000273 7203              <1>       jb    short sdp0
354 00000275 80EB7E            <1>       sub   bl, 7Eh
355                            <1> sdp0:
356 00000278 81C3[425D]        <1>       add   bx, drv.status
357 0000027C C60780            <1>       mov   byte [bx], 80h ; 'Present' flag
358                            <1>       ;
359 0000027F 88E8              <1>       mov   al, ch ; last cylinder (bits 0-7)
360 00000281 88CC              <1>       mov   ah, cl ;
361 00000283 C0EC06            <1>       shr   ah, 6  ; last cylinder (bits 8-9)
362 00000286 81EB[425D]        <1>       sub   bx, drv.status
363 0000028A D0E3              <1>       shl   bl, 1
364 0000028C 81C3[FC5C]        <1>       add   bx, drv.cylinders
365 00000290 40                <1>       inc   ax  ; convert max. cyl number to cyl count
366 00000291 8907              <1>       mov   [bx], ax
367 00000293 50                <1>       push  ax ; ** cylinders
368 00000294 81EB[FC5C]        <1>       sub   bx, drv.cylinders
369 00000298 81C3[0A5D]        <1>       add   bx, drv.heads
370 0000029C 30E4              <1>       xor   ah, ah
371 0000029E 88F0              <1>       mov   al, dh ; heads
372 000002A0 40                <1>       inc   ax
373 000002A1 8907              <1>       mov   [bx], ax
374 000002A3 81EB[0A5D]        <1>         sub     bx, drv.heads
375 000002A7 81C3[185D]        <1>         add     bx, drv.spt
376 000002AB 30ED              <1>       xor   ch, ch
377 000002AD 80E13F            <1>       and   cl, 3Fh      ; sectors (bits 0-6)
378 000002B0 890F              <1>       mov   [bx], cx
379 000002B2 81EB[185D]        <1>         sub     bx, drv.spt
380 000002B6 D1E3              <1>       shl   bx, 1
381 000002B8 81C3[265D]        <1>       add   bx, drv.size ; disk size (in sectors)
382                            <1>       ; LBA size = cylinders * heads * secpertrack
383 000002BC F7E1              <1>       mul   cx
384 000002BE 89C2              <1>       mov   dx, ax ; heads*spt
385 000002C0 58                <1>       pop   ax ; ** cylinders
386 000002C1 48                <1>       dec   ax ; 1 cylinder reserved (!?)
387 000002C2 F7E2              <1>       mul   dx ; cylinders * (heads*spt)
388 000002C4 8907              <1>       mov   [bx], ax
389 000002C6 895702            <1>       mov   [bx+2], dx
390                            <1>       ;
391                            <1>       ;pop   bx
392 000002C9 C3                <1>       retn
393                            <1>
394                            <1> L16:  ; 28/05/2016
203
204                                      ; 10/11/2014
205 000002CA FA                          cli   ; Disable interrupts (clear interrupt flag)
206                                            ; Reset Interrupt MASK Registers (Master&Slave)
207                                      ;mov   al, 0FFh     ; mask off all interrupts
208                                      ;out   21h, al             ; on master PIC (8259)
209                                      ;jmp   $+2  ; (delay)
210                                      ;out   0A1h, al     ; on slave PIC (8259)
211                                      ;
```

```
                                        ; Disable NMI
000002CB B080                           mov    al, 80h
000002CD E670                           out    70h, al             ; set bit 7 to 1 for disabling NMI
                                        ;23/02/2015
                                        ;nop                ;
                                        ;in    al, 71h              ; read in 71h just after writing out to 70h
                                                            ; for preventing unknown state (!?)
                                        ;
                                        ; 20/08/2014
                                        ; Moving the kernel 64 KB back (to physical address 0)
                                        ; DS = CS = 1000h
                                        ; 05/11/2014
000002CF 31C0                           xor    ax, ax
000002D1 8EC0                           mov    es, ax ; ES = 0
                                        ;
                                        ; 04/07/2016 -  TRDOS 386 (64K - 128K kernel)
                                              xor    si, si
000002D3 31F6
000002D5 31FF                           xor    di, di
000002D7 B90040                         mov    cx, 16384
000002DA F366A5                         rep    movsd
                                        ;
000002DD 06                             push   es ; 0
000002DE 68[E202]                       push   L17
000002E1 CB                             retf
                                    L17:
000002E2 B90010                         mov    cx, 1000h
000002E5 8EC1                           mov    es, cx  ; 1000h
000002E7 01C9                           add    cx, cx
000002E9 8ED9                           mov    ds, cx  ; 2000h
000002EB 29F6                           sub    si, si
000002ED 29FF                           sub    di, di
000002EF B90040                         mov    cx, 16384
000002F2 F366A5                         rep    movsd

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

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

                                          ;mov     ax, cs
                                          ;mov     ds, ax

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

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

                                        ; Jump to 32 bit code

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

                                    ; 20/02/2017


                                    [BITS 32]

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

                                    clear_bss: ; Clear uninitialized data area
                                          ; 11/03/2015
00000335 31C0                             xor  eax, eax ; 0
00000337 B9A2700000                       mov  ecx, (bss_end - bss_start)/4
                                          ;shr  ecx, 2 ; bss section is already aligned for double words
0000033C BF[124F0100]                     mov  edi, bss_start
00000341 F3AB                             rep  stosd
```

```
                                memory_init:
                                      ; Initialize memory allocation table and page tables
                                      ; 16/11/2014
                                      ; 15/11/2014
                                      ; 07/11/2014
                                      ; 06/11/2014
                                      ; 05/11/2014
                                      ; 04/11/2014
                                      ; 31/10/2014 (Retro UNIX 386 v1 - Beginning)
                                      ;
                                ;     xor   eax, eax
                                ;     xor   ecx, ecx
00000343 B108                         mov   cl, 8
00000345 BF00001000                   mov   edi, MEM_ALLOC_TBL
0000034A F3AB                         rep   stosd           ; clear Memory Allocation Table
                                                            ; for the first 1 MB memory
                                      ;
0000034C 668B0D[EE5C0000]             mov   cx, [mem_1m_1k]       ; Number of contiguous KB between
                                                            ; 1 and 16 MB, max. 3C00h = 15 MB.
00000353 66C1E902                     shr   cx, 2           ; convert 1 KB count to 4 KB count
00000357 890D[08520100]               mov   [free_pages], ecx
0000035D 668B15[F05C0000]             mov   dx, [mem_16m_64k]  ; Number of contiguous 64 KB blocks
                                                            ; between 16 MB and 4 GB.
00000364 6609D2                       or    dx, dx
00000367 7413                         jz    short mi_0
                                      ;
00000369 6689D0                       mov   ax, dx
0000036C C1E004                       shl   eax, 4          ; 64 KB -> 4 KB (page count)
0000036F 0105[08520100]               add   [free_pages], eax
00000375 0500100000                   add   eax, 4096       ; 16 MB = 4096 pages
0000037A EB07                         jmp   short mi_1
                                mi_0:
0000037C 6689C8                       mov   ax, cx
0000037F 66050001                     add   ax, 256               ; add 256 pages for the first 1 MB
                                mi_1:
00000383 A3[04520100]                 mov   [memory_size], eax ; Total available memory in pages
                                                            ; 1 alloc. tbl. bit = 1 memory page
                                                            ; 32 allocation bits = 32 mem. pages
                                      ;
00000388 05FF7F0000                   add   eax, 32767      ; 32768 memory pages per 1 M.A.T. page
0000038D C1E80F                       shr   eax, 15               ; ((32768 * x) + y) pages (y < 32768)
                                                            ;  --> x + 1 M.A.T. pages, if y > 0
                                                            ;  --> x M.A.T. pages, if y = 0
00000390 66A3[18520100]               mov   [mat_size], ax        ; Memory Alloc. Table Size in pages
00000396 C1E00C                       shl   eax, 12               ; 1 M.A.T. page = 4096 bytes
                                      ;                     ; Max. 32 M.A.T. pages (4 GB memory)
00000399 89C3                         mov   ebx, eax        ; M.A.T. size in bytes
                                      ; Set/Calculate Kernel's Page Directory Address
0000039B 81C300001000                 add   ebx, MEM_ALLOC_TBL
000003A1 891D[00520100]               mov   [k_page_dir], ebx  ; Kernel's Page Directory address
                                                            ; just after the last M.A.T. page
                                      ;
000003A7 83E804                       sub   eax, 4          ; convert M.A.T. size to offset value
000003AA A3[10520100]                 mov   [last_page], eax   ; last page ofset in the M.A.T.
                                      ;                     ; (allocation status search must be
                                                            ; stopped after here)
000003AF 31C0                         xor   eax, eax
000003B1 48                           dec   eax             ; FFFFFFFFh (set all bits to 1)
000003B2 6651                         push  cx
000003B4 C1E905                       shr   ecx, 5          ; convert 1 - 16 MB page count to
                                                            ; count of 32 allocation bits
000003B7 F3AB                         rep   stosd
000003B9 6659                         pop   cx
000003BB 40                           inc   eax             ; 0
000003BC 80E11F                       and   cl, 31          ; remain bits
000003BF 7412                         jz    short mi_4
000003C1 8907                         mov   [edi], eax      ; reset
                                mi_2:
000003C3 0FAB07                       bts   [edi], eax      ; 06/11/2014
000003C6 FEC9                         dec   cl
000003C8 7404                         jz    short mi_3
000003CA FEC0                         inc   al
000003CC EBF5                         jmp   short mi_2
                                mi_3:
000003CE 28C0                         sub   al, al          ; 0
000003D0 83C704                       add   edi, 4          ; 15/11/2014
                                mi_4:
000003D3 6609D2                       or    dx, dx          ; check 16M to 4G memory space
000003D6 7421                         jz    short mi_6      ; max. 16 MB memory, no more...
                                      ;
000003D8 B900021000                   mov   ecx, MEM_ALLOC_TBL + 512 ; End of first 16 MB memory
                                      ;
000003DD 29F9                         sub   ecx, edi        ; displacement (to end of 16 MB)
000003DF 7406                         jz    short mi_5      ; jump if EDI points to
                                                            ;         end of first 16 MB
000003E1 D1E9                         shr   ecx, 1          ; convert to dword count
000003E3 D1E9                         shr   ecx, 1          ; (shift 2 bits right)
000003E5 F3AB                         rep   stosd           ; reset all bits for reserved pages
                                                            ; (memory hole under 16 MB)
                                mi_5:
000003E7 6689D1                       mov   cx, dx          ; count of 64 KB memory blocks
000003EA D1E9                         shr   ecx, 1          ; 1 alloc. dword per 128 KB memory
000003EC 9C                           pushf                 ; 16/11/2014
000003ED 48                           dec   eax             ; FFFFFFFFh (set all bits to 1)
000003EE F3AB                         rep   stosd
000003F0 40                           inc   eax             ; 0
000003F1 9D                           popf                  ; 16/11/2014
000003F2 7305                         jnc   short mi_6
000003F4 6648                         dec   ax              ; eax = 0000FFFFh
000003F6 AB                           stosd
000003F7 6640                         inc   ax              ; 0
                                mi_6:
```

```
411 000003F9 39DF                       cmp   edi, ebx       ; check if EDI points to
412 000003FB 730A                       jnb   short mi_7     ; end of memory allocation table
413                                     ;                    ; (>= MEM_ALLOC_TBL + 4906)
414 000003FD 89D9                       mov   ecx, ebx       ; end of memory allocation table
415 000003FF 29F9                       sub   ecx, edi       ; convert displacement/offset
416 00000401 D1E9                       shr   ecx, 1         ; to dword count
417 00000403 D1E9                       shr   ecx, 1         ; (shift 2 bits right)
418 00000405 F3AB                       rep   stosd          ; reset all remain M.A.T. bits
419                                 mi_7:
420                                     ; Reset M.A.T. bits in M.A.T. (allocate M.A.T. pages)
421 00000407 BA00001000                 mov   edx, MEM_ALLOC_TBL
422                                     ;sub   ebx, edx       ; Mem. Alloc. Tbl. size in bytes
423                                     ;shr   ebx, 12              ; Mem. Alloc. Tbl. size in pages
424 0000040C 668B0D[18520100]           mov   cx, [mat_size]        ; Mem. Alloc. Tbl. size in pages
425 00000413 89D7                       mov   edi, edx
426 00000415 C1EF0F                     shr   edi, 15               ; convert M.A.T. address to
427                                                         ; byte offset in M.A.T.
428                                                         ; (1 M.A.T. byte points to
429                                                         ;         32768 bytes)
430                                                         ; Note: MEM_ALLOC_TBL address
431                                                         ; must be aligned on 128 KB
432                                                         ; boundary!
433 00000418 01D7                       add   edi, edx       ; points to M.A.T.'s itself
434                                     ; eax = 0
435 0000041A 290D[08520100]             sub   [free_pages], ecx ; 07/11/2014
436                                 mi_8:
437 00000420 0FB307                     btr   [edi], eax     ; clear bit 0 to bit x (1 to 31)
438                                     ;dec   bl
439 00000423 FEC9                       dec   cl
440 00000425 7404                       jz    short mi_9
441 00000427 FEC0                       inc   al
442 00000429 EBF5                       jmp   short mi_8
443                                 mi_9:
444                                     ;
445                                     ; Reset Kernel's Page Dir. and Page Table bits in M.A.T.
446                                     ;            (allocate pages for system page tables)
447
448                                     ; edx = MEM_ALLOC_TBL
449 0000042B 8B0D[04520100]             mov   ecx, [memory_size] ; memory size in pages (PTEs)
450 00000431 81C1FF030000               add   ecx, 1023      ; round up (1024 PTEs per table)
451 00000437 C1E90A                     shr   ecx, 10              ; convert memory page count to
452                                                         ; page table count (PDE count)
453                                     ;
454 0000043A 51                         push  ecx            ; (**) PDE count (<= 1024)
455                                     ;
456 0000043B 41                         inc   ecx            ; +1 for kernel page directory
457                                     ;
458 0000043C 290D[08520100]             sub   [free_pages], ecx ; 07/11/2014
459                                     ;
460 00000442 8B35[00520100]             mov   esi, [k_page_dir] ; Kernel's Page Directory address
461 00000448 C1EE0C                     shr   esi, 12              ; convert to page number
462                                 mi_10:
463 0000044B 89F0                       mov   eax, esi       ; allocation bit offset
464 0000044D 89C3                       mov   ebx, eax
465 0000044F C1EB03                     shr   ebx, 3         ; convert to alloc. byte offset
466 00000452 80E3FC                     and   bl,  0FCh      ; clear bit 0 and bit 1
467                                                         ;   to align on dword boundary
468 00000455 83E01F                     and   eax, 31              ; set allocation bit position
469                                                         ;  (bit 0 to bit 31)
470                                     ;
471 00000458 01D3                       add   ebx, edx       ; offset in M.A.T. + M.A.T. address
472                                     ;
473 0000045A 0FB303                     btr   [ebx], eax     ; reset relevant bit (0 to 31)
474                                     ;
475 0000045D 46                         inc   esi            ; next page table
476 0000045E E2EB                       loop  mi_10          ; allocate next kernel page table
477                                                         ; (ecx = page table count + 1)
478                                     ;
479 00000460 59                         pop   ecx            ; (**) PDE count (= pg. tbl. count)
480                                     ;
481                                     ; Initialize Kernel Page Directory and Kernel Page Tables
482                                     ;
483                                     ; Initialize Kernel's Page Directory
484 00000461 8B3D[00520100]             mov   edi, [k_page_dir]
485 00000467 89F8                       mov   eax, edi
486 00000469 0C03                       or    al, PDE_A_PRESENT + PDE_A_WRITE
487                                                         ; supervisor + read&write + present
488 0000046B 89CA                       mov   edx, ecx       ; (**) PDE count (= pg. tbl. count)
489                                 mi_11:
490 0000046D 0500100000                 add   eax, 4096      ; Add page size (PGSZ)
491                                                          ; EAX points to next page table
492 00000472 AB                         stosd
493 00000473 E2F8                       loop  mi_11
494 00000475 29C0                       sub   eax, eax       ; Empty PDE
495 00000477 66B90004                   mov   cx, 1024       ; Entry count (PGSZ/4)
496 0000047B 29D1                       sub   ecx, edx
497 0000047D 7402                       jz    short mi_12
498 0000047F F3AB                       rep   stosd          ; clear remain (empty) PDEs
499                                     ;
500                                     ; Initialization of Kernel's Page Directory is OK, here.
501                                 mi_12:
502                                     ; Initialize Kernel's Page Tables
503                                     ;
504                                     ; (EDI points to address of page table 0)
505                                     ; eax = 0
506 00000481 8B0D[04520100]             mov   ecx, [memory_size] ; memory size in pages
507 00000487 89CA                       mov   edx, ecx       ; (***)
508 00000489 B003                       mov   al, PTE_A_PRESENT + PTE_A_WRITE
509                                                         ; supervisor + read&write + present
```

```
                                mi_13:
                                      stosd
                                      add   eax, 4096
                                      loop  mi_13
                                      and   dx, 1023     ; (***)
                                      jz    short mi_14
                                      mov   cx, 1024
                                      sub   cx, dx       ; from dx (<= 1023) to 1024
                                      xor   eax, eax
                                      rep   stosd        ; clear remain (empty) PTEs
                                                         ; of the last page table
                                mi_14:
                                      ;  Initialization of Kernel's Page Tables is OK, here.
                                      ;
                                      mov   eax, edi     ; end of the last page table page
                                                          ; (beginging of user space pages)
                                      shr   eax, 15            ; convert to M.A.T. byte offset
                                      and   al, 0FCh     ; clear bit 0 and bit 1 for
                                                         ; aligning on dword boundary

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

                                      ; Enable paging
                                      ;
                                        mov     eax, [k_page_dir]
                                      mov   cr3, eax
                                      mov   eax, cr0
                                      or    eax, 80000000h      ; set paging bit (bit 31)
                                      mov   cr0, eax
                                        ;jmp    KCODE:StartPMP

                                      db 0EAh            ; Opcode for far jump
                                        dd StartPMP             ; 32 bit offset
                                      dw KCODE           ; kernel code segment descriptor


                                StartPMP:
                                      ; 06/11//2014
                                      ; Clear video page 0
                                      ;
                                      ; Temporary Code
                                      ;
                                      mov   ecx, 80*25/2
                                      mov   edi, 0B8000h
                                      ; 30/01/2016
                                      ;xor   eax, eax     ; black background, black fore color
                                      mov   eax, 07000700h  ; black background, light gray fore color
                                      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)
                                      mov   esi, starting_msg
                                      ;; 14/08/2015 (kernel version message will appear
                                      ;;           when protected mode and paging is enabled)
                                      mov   edi, 0B8000h ; 27/08/2014
                                      mov   ah, 0Ah ; Black background, light green forecolor
                                      ; 20/08/2014
                                      call  printk

                                      ; 'UNIX v7/x86' source code by Robert Nordier (1999)
                                      ; // Set IRQ offsets
                                      ;
                                      ;  Linux (v0.12) source code by Linus Torvalds (1991)
                                      ;
                                                                ;; ICW1
                                      mov   al, 11h                   ; Initialization sequence
                                      out   20h, al                   ;     8259A-1
                                      ; jmp  $+2
                                      out   0A0h, al            ;     8259A-2
                                                                ;; ICW2
                                      mov   al, 20h                   ; Start of hardware ints (20h)
                                      out   21h, al                   ;     for 8259A-1
                                      ; jmp  $+2
                                      mov   al, 28h                   ; Start of hardware ints (28h)
                                      out   0A1h, al            ;     for 8259A-2
                                                                ;
                                      mov   al, 04h                   ;; ICW3
                                      out   21h, al                   ;     IRQ2 of 8259A-1 (master)
                                      ; jmp  $+2
                                      mov   al, 02h                   ;     is 8259A-2 (slave)
                                      out   0A1h, al            ;
                                                                ;; ICW4
                                      mov   al, 01h                   ;
                                      out   21h, al                   ;     8086 mode, normal EOI
                                      ; jmp  $+2
                                      out   0A1h, al            ;     for both chips.

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

```
                                        ; 02/04/2015
                                        ; 26/03/2015 System call (INT 30h) modification
                                        ;  DPL = 3 (Interrupt service routine can be called from user mode)
                                        ;
                                        ;; Linux (v0.12) source code by Linus Torvalds (1991)
                                        ;  setup_idt:
                                        ;
                                          ;; 16/02/2015
                                        ;;mov     dword [DISKETTE_INT], fdc_int ; IRQ 6 handler
                                        ; 21/08/2014 (timer_int)
0000050E BE[2C100100]                   mov    esi, ilist
00000513 8D3D[184F0100]                 lea    edi, [idt]
                                        ; 26/03/2015
00000519 B930000000                     mov    ecx, 48            ; 48 hardware interrupts (INT 0 to INT 2Fh)
                                        ; 02/04/2015
0000051E BB00000800                     mov    ebx,  80000h
                                rp_sidt1:
00000523 AD                             lodsd
00000524 89C2                           mov    edx, eax
00000526 66BA008E                       mov    dx, 8E00h
0000052A 6689C3                         mov    bx, ax
0000052D 89D8                           mov    eax, ebx     ; /* selector = 0x0008 = cs */
                                                                ; /* interrupt gate - dpl=0, present */
0000052F AB                             stosd  ; selector & offset bits 0-15
00000530 89D0                           mov    eax, edx
00000532 AB                             stosd  ; attributes & offset bits 16-23
00000533 E2EE                           loop   rp_sidt1
                                        ; 15/04/2016
                                        ; TRDOS 386 (TRDOS v2.0) /// 32 sofware interrupts ///
                                        ;mov   cl, 16        ; 16 software interrupts (INT 30h to INT 3Fh)
00000535 B120                           mov    cl, 32      ; 32 software interrupts (INT 30h to INT 4Fh)
                                rp_sidt2:
00000537 AD                             lodsd
00000538 21C0                           and    eax, eax
0000053A 7413                           jz     short rp_sidt3
0000053C 89C2                           mov    edx, eax
0000053E 66BA00EE                       mov    dx, 0EE00h   ; P=1b/DPL=11b/01110b
00000542 6689C3                         mov    bx, ax
00000545 89D8                           mov    eax, ebx     ; selector & offset bits 0-15
00000547 AB                             stosd
00000548 89D0                           mov    eax, edx
0000054A AB                             stosd
0000054B E2EA                           loop   rp_sidt2
0000054D EB16                           jmp    short sidt_OK
                                rp_sidt3:
0000054F B8[AA0A0000]                   mov    eax, ignore_int
00000554 89C2                           mov    edx, eax
00000556 66BA00EE                       mov    dx, 0EE00h   ; P=1b/DPL=11b/01110b
0000055A 6689C3                         mov    bx, ax
0000055D 89D8                           mov    eax, ebx     ; selector & offset bits 0-15
                                rp_sidt4:
0000055F AB                             stosd
00000560 92                             xchg   eax, edx
00000561 AB                             stosd
00000562 92                             xchg   edx, eax
00000563 E2FA                           loop   rp_sidt4
                                sidt_OK:
00000565 0F011D[665C0000]               lidt   [idtd]
                                        ;
                                        ; TSS descriptor setup ; 24/03/2015
0000056C B8[98510100]                   mov    eax, task_state_segment
00000571 66A3[5A5C0000]                 mov    [gdt_tss0], ax
00000577 C1C010                         rol    eax, 16
0000057A A2[5C5C0000]                   mov    [gdt_tss1], al
0000057F 8825[5F5C0000]                 mov    [gdt_tss2], ah
00000585 66C705[FE510100]68-            mov    word [tss.IOPB], tss_end - task_state_segment
0000058D 00
                                                ;
                                                ; IO Map Base address (When this address points
                                                ; to end of the TSS, CPU does not use IO port
                                                ; permission bit map for RING 3 IO permissions,
                                                ; access to any IO ports in ring 3 will be forbidden.)
                                                ;
                                        ;mov   [tss.esp0], esp ; TSS offset 4
                                        ;mov   word [tss.ss0], KDATA ; TSS offset 8 (SS)
0000058E 66B82800                       mov    ax, TSS  ; It is needed when an interrupt
                                                        ; occurs (or a system call -software INT- is requested)
                                                        ; while cpu running in ring 3 (in user mode).
                                                        ; (Kernel stack pointer and segment will be loaded
                                                        ; from offset 4 and 8 of the TSS, by the CPU.)
00000592 0F00D8                         ltr    ax  ; Load task register
                                        ;
                                esp0_set0:
                                        ; 30/07/2015
00000595 8B0D[04520100]                 mov    ecx, [memory_size] ; memory size in pages
0000059B C1E10C                         shl    ecx, 12 ; convert page count to byte count
0000059E 81F900004000                   cmp    ecx, CORE ; beginning of user's memory space (400000h)
                                                         ; (kernel mode virtual address)
000005A4 7605                           jna    short esp0_set1
                                        ;
                                        ; If available memory > CORE (end of the 1st 4 MB)
                                        ; set stack pointer to CORE
                                        ;(Because, PDE 0 is reserved for kernel space in user's page directory)
                                        ;(PDE 0 points to page table of the 1st 4 MB virtual address space)
000005A6 B900004000                     mov    ecx, CORE
                                esp0_set1:
000005AB 89CC                           mov    esp, ecx ; top of kernel stack (**tss.esp0**)
                                esp0_set_ok:
                                        ; 30/07/2015 (**tss.esp0**)
000005AD 8925[9C510100]                 mov    [tss.esp0], esp
000005B3 66C705[A0510100]10-              mov     word [tss.ss0], KDATA
000005BB 00
```

```
                                        ; 14/08/2015
                                        ; 10/11/2014 (Retro UNIX 386 v1 - Erdogan Tan)
                                        ;
                                        ;cli   ; Disable interrupts (for CPU)
                                        ;    (CPU will not handle hardware interrupts, except NMI!)
                                        ;
000005BC 30C0                           xor    al, al       ; Enable all hardware interrupts!
000005BE E621                           out    21h, al             ; (IBM PC-AT compatibility)
000005C0 EB00                           jmp    $+2          ; (All conventional PC-AT hardware
000005C2 E6A1                           out    0A1h, al     ;  interrupts will be in use.)
                                                            ; (Even if related hardware component
                                                            ;  does not exist!)
                                        ; Enable NMI
000005C4 B07F                           mov    al, 7Fh             ; Clear bit 7 to enable NMI (again)
000005C6 E670                           out    70h, al
                                        ; 23/02/2015
000005C8 90                             nop
000005C9 E471                           in     al, 71h             ; read in 71h just after writing out to 70h
                                                            ; for preventing unknown state (!?)
                                        ;
                                        ; Only a NMI can occur here... (Before a 'STI' instruction)
                                        ;
                                        ; 02/09/2014
000005CB 6631DB                         xor    bx, bx
000005CE 66BA0002                       mov    dx, 0200h    ; Row 2, column 0  ; 07/03/2015
000005D2 E871170000                     call   _set_cpos    ; 24/01/2016
                                        ;
                                        ; 06/11/2014
000005D7 E8782C0000                     call   memory_info
                                        ; 14/08/2015
                                        ;call getch ; 28/02/2015
                                    drv_init:
000005DC FB                             sti    ; Enable Interrupts
                                        ; 06/02/2015
000005DD 8B15[F85C0000]                 mov    edx, [hd0_type] ; hd0, hd1, hd2, hd3
000005E3 668B1D[F65C0000]               mov    bx, [fd0_type] ; fd0, fd1
                                        ; 22/02/2015
000005EA 6621DB                         and    bx, bx
000005ED 751C                           jnz    short di1
                                        ;
000005EF 09D2                           or     edx, edx
000005F1 752A                           jnz    short di2
                                        ;
                                    setup_error:
000005F3 BE[32130100]                   mov    esi, setup_error_msg
                                    psem:
000005F8 AC                             lodsb
000005F9 08C0                           or     al, al
                                        ;jz    short haltx ; 22/02/2015
000005FB 7427                           jz     short di3
000005FD 56                             push   esi
                                        ; 13/05/2016
000005FE BB07000000                     mov    ebx, 7 ; Black background,
                                                      ; light gray forecolor
                                                      ; Video page 0 (BH=0)
00000603 E8AA160000                     call   _write_tty
00000608 5E                             pop    esi
00000609 EBED                           jmp    short psem
                                        
                                    di1:
                                        ; supress 'jmp short T6'
                                        ;  (activate fdc motor control code)
0000060B 66C705[EB060000]90-            mov    word [T5], 9090h ; nop
00000613 90
                                        ;
                                        ;mov   ax, int_0Eh  ; IRQ 6 handler
                                        ;mov   di, 0Eh*4    ; IRQ 6 vector
                                        ;stosw
                                        ;mov   ax, cs
                                        ;stosw
                                        ;; 16/02/2015
                                          ;;mov     dword [DISKETTE_INT], fdc_int ; IRQ 6 handler
                                        ;
00000614 E8AF3B0000                     CALL   DSKETTE_SETUP; Initialize Floppy Disks
                                        ;
00000619 09D2                           or     edx, edx
0000061B 7407                             jz      short di3
                                    di2:
0000061D E8EC3B0000                     call         DISK_SETUP   ; Initialize Fixed Disks
00000622 72CF                             jc      short setup_error
                                    di3:
00000624 E8FF2B0000                     call   setup_rtc_int; 22/05/2015 (dsectrpm.s)
                                        ;
00000629 E89A0B0100                     call   display_disks ; 07/03/2015  (Temporary)
                                    ;haltx:
                                        ; 14/08/2015
                                        ;call  getch ; 22/02/2015
                                        ;sti   ; Enable interrupts (for CPU)
                                    ;     ; 29/01/2016
                                    ;     sub    ah, ah ;  read time count
                                    ;     call   int1Ah
                                    ;     mov    edx, ecx ; 18.2 * seconds
                                    ;md_info_msg_wait1:
                                    ;     ; 29/01/2016
                                    ;     mov    ah, 1
                                    ;     call   int16h
                                    ;     jz     short md_info_msg_wait2
                                    ;     xor    ah, ah ; 0
                                    ;       call    int16h
                                    ;     jmp    short md_info_msg_ok
                                    ;md_info_msg_wait2:
                                    ;     sub    ah, ah  ; read time count
```

```
                                ;     call   int1Ah
                                ;     cmp    edx, ecx ; ; 18.2 * seconds
                                ;     jna    short md_info_msg_wait3
                                ;     xchg   edx, ecx
                                ;md_info_msg_wait3:
                                ;     sub    ecx, edx
                                ;     cmp    ecx, 127 ; 7 seconds (18.2 * 7)
                                ;     jb     short md_info_msg_wait1
                                ;md_info_msg_ok:
                                      ; 08/09/2016
0000062E 0F20C0                       mov    eax, cr0
00000631 A810                         test   al, 10h  ; Bit 4, ET (Extension Type)
00000633 7408                         jz     short sysinit
                                      ; 27/02/2017
00000635 FE05[C05F0100]               inc    byte [fpready]
                                      ; 80387 (FPU) is ready
0000063B DBE3                         fninit ; Initialize Floating-Point Unit
                                sysinit:
                                      ; 30/06/2015
0000063D E80C5C0000                   call   sys_init
                                      ;
                                      ;jmp   cpu_reset ; 22/02/2015
                                hang:
                                      ; 23/02/2015
                                      ;sti               ; Enable interrupts
00000642 F4                           hlt
                                      ;
                                      ;nop
                                      ;; 03/12/2014
                                      ;; 28/08/2014
                                      ;mov   ah, 11h
                                      ;call  getc
                                      ;jz      _c8
                                      ;
                                      ; 23/02/2015
                                      ; 06/02/2015
                                      ; 07/09/2014
00000643 31DB                         xor    ebx, ebx
00000645 8A1D[2E520100]               mov    bl, [ptty]   ; active_page
0000064B 89DE                         mov    esi, ebx
0000064D 66D1E6                       shl    si, 1
00000650 81C6[30520100]               add    esi, ttychr
00000656 668B06                       mov    ax, [esi]
00000659 6621C0                       and    ax, ax
                                      ;jz    short _c8
0000065C 74E4                         jz     short hang
0000065E 66C7060000                   mov    word [esi], 0
00000663 80FB03                       cmp    bl, 3        ; Video page 3
                                      ;jb    short _c8
00000666 72DA                         jb     short hang
                                      ;
                                      ; 13/05/2016
                                      ; 07/09/2014
                                nxtl:
00000668 6653                         push   bx
0000066A 66BB0E00                     mov    bx, 0Eh      ; Yellow character
                                                          ; on black background
                                                          ; bh = 0 (video page 0)
                                                          ; Retro UNIX 386 v1 - Video Mode 0
                                                          ; (PC/AT Video Mode 3 - 80x25 Alpha.)
0000066E 6650                         push   ax
00000670 E83D160000                   call   _write_tty
00000675 6658                         pop    ax
00000677 665B                         pop    bx
00000679 3C0D                         cmp    al, 0Dh            ; carriage return (enter)
                                      ;jne   short _c8
0000067B 75C5                         jne    short hang
0000067D B00A                         mov    al, 0Ah            ; next line
0000067F EBE7                         jmp    short nxtl
                                
                                ;_c8:
                                ;     ; 25/08/2014
                                ;     cli                       ; Disable interrupts
                                ;     mov    al, [scounter + 1]
                                ;     and    al, al
                                ;     jnz    hang
                                ;     call   rtc_p
                                ;     jmp     hang
                                
                                
                                      ; 27/08/2014
                                      ; 20/08/2014
                                printk:
                                        ;mov    edi, [scr_row]
                                pkl:
00000681 AC                           lodsb
00000682 08C0                         or     al, al
00000684 7404                         jz     short pkr
00000686 66AB                         stosw
00000688 EBF7                         jmp    short pkl
                                pkr:
0000068A C3                           retn
                                
                                ; 28/02/2017
                                ; 22/01/2017
                                ; 15/01/2017
                                ; 14/01/2017
                                ; 02/01/2017
                                ; 25/12/2016
                                ; 19/12/2016
                                ; 10/12/2016 (callback)
                                ; 06/06/2016
```

```
                        ; 23/05/2016
                        ; 22/05/2016 - TRDOS 386 (TRDOS v2.0) Timer Event Modifications
                        ; 25/07/2015
                        ; 14/05/2015 (multi tasking -time sharing- 'clock', x_timer)
                        ; 17/02/2015
                        ; 06/02/2015 (unix386.s)
                        ; 11/12/2014 - 22/12/2014 (dsectrm2.s)
                        ;
                        ; IBM PC-XT Model 286 Source Code - BIOS2.ASM (06/10/85)
                        ;
                        ;-- HARDWARE INT  08 H - ( IRQ LEVEL 0 ) ------------------------------------
                        ;     THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM FROM CHANNEL 0 OF        :
                        ;     THE 8254 TIMER.  INPUT FREQUENCY IS 1.19318 MHZ AND THE DIVISOR        :
                        ;     IS 65536, RESULTING IN APPROXIMATELY 18.2 INTERRUPTS EVERY SECOND.     :
                        ;                                                            :
                        ;     THE INTERRUPT HANDLER MAINTAINS A COUNT (40:6C) OF INTERRUPTS SINCE    :
                        ;     POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.        :
                        ;     THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT (40:40)  :
                        ;     OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE           :
                        ;     DISKETTE MOTOR(s), AND RESET THE MOTOR RUNNING FLAGS.                   :
                        ;     THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH            :
                        ;     INTERRUPT 1CH AT EVERY TIME TICK.  THE USER MUST CODE A           :
                        ;     ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE.        :
                        ;--------------------------------------------------------------------------
                        ;

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

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

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

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

                              ; 14/01/2017
000006A7 3B0D[00520100]       cmp    ecx, [k_page_dir]
000006AD 7409                 je     short T3

000006AF 8B0D[00520100]       mov    ecx, [k_page_dir]
000006B5 0F22D9               mov    cr3, ecx
                        T3:
                              ;sti                        ; INTERRUPTS BACK ON
000006B8 66FF05[80520100]     INC    word [TIMER_LOW]    ; INCREMENT TIME
000006BF 7507                 JNZ    short T4            ; GO TO TEST_DAY
000006C1 66FF05[82520100]     INC    word [TIMER_HIGH]   ; INCREMENT HIGH WORD OF TIME
                        T4:                               ; TEST_DAY
000006C8 66833D[82520100]18   CMP    word [TIMER_HIGH],018H    ; TEST FOR COUNT EQUALING 24 HOURS
000006D0 7519                 JNZ    short T5            ; GO TO DISKETTE_CTL
000006D2 66813D[80520100]B0-  CMP    word [TIMER_LOW],0B0H
000006DA 00
000006DB 750E                 JNZ    short T5           ; GO TO DISKETTE_CTL

                        ;-----        TIMER HAS GONE 24 HOURS
                              ;;SUB  AX,AX
                              ;MOV   [TIMER_HIGH],AX
                              ;MOV   [TIMER_LOW],AX
000006DD 29C0                 sub    eax, eax
000006DF A3[80520100]         mov    [TIMER_LH], eax
                              ;
000006E4 C605[84520100]01     MOV    byte [TIMER_OFL],1

                        ;-----        TEST FOR DISKETTE TIME OUT

                        T5:
                              ; 23/12/2014
000006EB EB1D                 jmp    short T6           ; will be replaced with nop, nop
                                                        ; (9090h) if a floppy disk
                                                        ; is detected.
                              ;mov   al,[CS:MOTOR_COUNT]
000006ED A0[87520100]         mov    al, [MOTOR_COUNT]
000006F2 FEC8                 dec    al
                              ;mov   [CS:MOTOR_COUNT], al      ; DECREMENT DISKETTE MOTOR CONTROL
000006F4 A2[87520100]         mov    [MOTOR_COUNT], al
                              ;mov   [ORG_MOTOR_COUNT], al
000006F9 750F                 JNZ    short T6           ; RETURN IF COUNT NOT OUT
000006FB B0F0                 mov    al,0F0h
                              ;AND   [CS:MOTOR_STATUS],al      ; TURN OFF MOTOR RUNNING BITS
000006FD 2005[86520100]       and    [MOTOR_STATUS], al
                              ;and   [ORG_MOTOR_STATUS], al
00000703 B00C                 MOV    AL,0CH             ; bit 3 = enable IRQ & DMA,
                                                        ; bit 2 = enable controller
                                                        ;      1 = normal operation
                                                        ;      0 = reset
                                                        ; bit 0, 1 = drive select
                                                        ; bit 4-7 = motor running bits
00000705 66BAF203             MOV    DX,03F2H           ; FDC CTL PORT
00000709 EE                   OUT    DX,AL              ; TURN OFF THE MOTOR
                        T6:
```

```
1016                                    ;inc   word [CS:wait_count]      ; 22/12/2014 (byte -> word)
1017                                                                     ; TIMER TICK INTERRUPT
1018                                    ;;inc  word [wait_count] ;;27/02/2015
1019                                    ;INT   1CH                       ; TRANSFER CONTROL TO A USER ROUTINE
1020                                    ;cli
1021 0000070A E857040000                call   u_timer                   ; TRANSFER CONTROL TO A USER ROUTINE
1022                                    ; 23/05/2016
1023 0000070F E8FDEB0000                call   clock                     ; Multi Tasking control procedure
1024                                T7:
1025                                    ; 14/10/2015
1026 00000714 B020                      MOV    AL,EOI                    ; GET END OF INTERRUPT MASK
1027 00000716 FA                        CLI                              ; DISABLE INTERRUPTS TILL STACK CLEARED
1028 00000717 E620                      OUT    INTA00,AL                 ; END OF INTERRUPT TO 8259 - 1
1029                                    ;
1030                                rtc_int_2:
1031                                    ; 26/12/2016
1032                                    ;mov   ecx, [cr3reg]
1033                                    ; 13/01/2017
1034 00000719 803D[D4030300]00          cmp    byte [u.t_lock], 0        ; T_LOCK
1035 00000720 7730                      ja     short timer_int_return  ; Timer Lock : 'sysrele' is needed !
1036                                    ; 28/02/2017
1037                                    ; We need to exit if the user's IRQ callback service is in progress!
1038                                    ; (To prevent a conflict!)
1039 00000722 803D[D8030300]00          cmp    byte [u.r_lock], 0  ; R_LOCK, IRQ callback service lock !
1040 00000729 7727                      ja     short timer_int_return  ; Timer Lock : 'sysrele' is needed !
1041                                    ; 15/01/2017
1042 0000072B 803D[945F0100]02          cmp    byte [priority], 2
1043 00000732 733A                      jnb    short T8  ; current process has a timer event (15/01/2017)
1044                                    ; 22/05/2016
1045 00000734 803D[955F0100]00          cmp    byte [p_change], 0 ; in 'set_run_sequence', in 'rtc_p'
1046 0000073B 7615                      jna    short timer_int_return ; 23/05/2016
1047
1048                                    ; 15/01/2017
1049
1050                                    ; present process must be changed with high priority process
1051                                    ;xor   al, al
1052 0000073D 31C0                      xor    eax, eax ; 26/12/2016
1053 0000073F A2[955F0100]              mov    [p_change], al ; 0
1054                                    ;mov   byte [priority], 2 ; 15/01/2017 (there is a timer event)
1055
1056 00000744 803D[5B030300]FF          cmp     byte [sysflg], 0FFh ; user or system space ?
1057 0000074B 7416                      je     short rtc_int_3     ; user space ([sysflg]= 0FFh)
1058
1059                                    ; system space, wait for 'sysret'
1060                                    ; to change running process
1061                                    ; with high priority (event) process
1062
1063 0000074D A2[A8030300]              mov    [u.quant], al ; 0
1064
1065                                timer_int_return: ; 23/05/2016 - jump from 'rtc_int' ('rtc_int_2')
1066 00000752 8B0D[5C040300]            mov    ecx, [cr3reg]       ; previous value/content of cr3 register
1067 00000758 0F22D9                    mov    cr3, ecx     ; restore cr3 register content
1068                                    ;
1069 0000075B 61                        popad ; edi, esi, ebp, temp (icrement esp by 4), ebx, edx, ecx, eax
1070                                    ;
1071 0000075C 0FA9                      pop    gs
1072 0000075E 0FA1                      pop    fs
1073 00000760 07                        pop    es
1074 00000761 1F                        pop    ds
1075                                    ;
1076 00000762 CF                        iretd  ; return from interrupt
1077
1078                                rtc_int_3:
1079 00000763 FE05[5B030300]            inc    byte [sysflg]       ; now, we are in system space
1080                                    ;
1081 00000769 E965BD0000                  jmp     sysrelease ; change running process immediatelly
1082
1083                                T8:
1084                                    ; 13/01/2017 (eax -> ebx)
1085                                    ; callback checking... (19/12/2016)
1086 0000076E 31DB                      xor    ebx, ebx
1087 00000770 871D[D0030300]            xchg   ebx, [u.tcb] ; callback address (0 = normal return)
1088 00000776 09DB                      or     ebx, ebx
1089 00000778 74D8                      jz     short timer_int_return
1090
1091                                    ; Set user's callback routine as return address from this interrupt
1092                                    ; and set normal return address as return address from callback
1093                                    ; routine!!! (19/12/2016)
1094
1095                                    ; 14/01/2017
1096                                    ; 13/01/2017 - Timer Lock (T_LOCK)
1097 0000077A FE05[D4030300]            inc    byte [u.t_lock]
1098 00000780 8A0D[5B030300]            mov    cl, [sysflg]
1099 00000786 880D[D5030300]            mov    [u.t_mode], cl
1100
1101 0000078C 8B2D[9C510100]            mov    ebp, [tss.esp0] ; kernel stack address (for ring 0)
1102 00000792 83ED14                    sub    ebp, 20             ; eip, cs, eflags, esp, ss
1103 00000795 892D[5C030300]            mov    [u.sp], ebp
1104 0000079B 8925[60030300]            mov    [u.usp], esp
1105
1106                                    ;or    word [ebp+8], 200h ; 22/01/2017, force enabling interrupts
1107
1108 000007A1 8B44241C                  mov    eax, [esp+28] ; pushed eax
1109 000007A5 A3[64030300]              mov    [u.r0], eax
1110
1111 000007AA E87CDF0000                call   wswap ; save user's registers & status
1112
1113                                    ; software int is in ring 0 but timer int must return to ring 3
1114                                    ; so, ring 3 return address and stack registers
1115                                    ; (eip, cs, eflags, esp, ss)
1116                                    ; must be copied to timer int return
1117                                    ; eip will be replaced by callback service routine address
```

```
1118
1119 000007AF C605[5B030300]FF            mov   byte [sysflg], 0FFh ; user mode
1120
1121                                     ; system mode (system call)
1122                                     ;mov   ebp, [u.sp] ; EIP (u), CS (UCODE), EFLAGS (u),
1123                                                        ; ESP (u), SS (UDATA)
1124
1125 000007B6 8B4510                    mov   eax, [ebp+16]; SS (UDATA
1126 000007B9 89E6                      mov   esi, esp
1127 000007BB 50                        push  eax
1128 000007BC 50                        push  eax
1129 000007BD 89E7                      mov   edi, esp
1130 000007BF 893D[60030300]            mov   [u.usp], edi
1131 000007C5 B908000000                mov   ecx, ((ESPACE/4) - 4) ; except DS, ES, FS, GS
1132 000007CA F3A5                      rep   movsd
1133 000007CC B104                      mov   cl, 4
1134 000007CE F3AB                      rep   stosd
1135 000007D0 893D[5C030300]            mov   [u.sp], edi
1136 000007D6 89EE                      mov   esi, ebp
1137 000007D8 B105                      mov   cl, 5 ; EIP (u), CS (UCODE), EFLAGS (u), ESP (u), SS (UDATA)
1138 000007DA F3A5                      rep   movsd
1139
1140 000007DC 8B0D[B8030300]            mov   ecx, [u.pgdir]
1141 000007E2 890D[5C040300]            mov   [cr3reg], ecx
1142
1143                                     ; 13/01/207 (eax -> ebx)
1144                                     ; EBX = callback routine address (virtual, not physical address!)
1145
1146                                     ; 09/01/2017
1147                                     ; !!! CALLBACK ROUTINE MUST BE ENDED/RETURNED WITH 'sysrele'
1148                                     ;     system call !!!
1149                                     ; 25/12/2016
1150                                     ; Callback Note: (19/12/2016)
1151                                     ; !!! CALLBACK ROUTINE MUST BE ENDED/RETURNED WITH 'RETN' !!!
1152                                     ;     pushf ; save flags
1153                                     ;     <callback service code>
1154                                     ;     popf  ; restore flags
1155                                     ;     retn ; return to normal running address
1156                                     ;
1157
1158                                     ; 15/01/2017
1159                                     ; 14/01/2017
1160                                     ; 13/01/2017 (eax -> ebx)
1161                                     ; 10/01/2017
1162                                   set_callback_addr:
1163                                     ; 09/01/2017 (**)
1164                                     ; 02/01/2017 (*)
1165                                     ; 25/12/2016 (*)
1166                                     ; 19/12/2016 (TRDOS 386 feature only!)
1167                                     ;
1168                                     ; This routine sets return address
1169                                     ; to start of user's interrupt
1170                                     ; service (callback) address
1171                                     ;; and sets callback 'retn' address to normal
1172                                     ;; return address of user's running code!
1173                                     ;
1174                                     ; INPUT:
1175                                     ;     EBX = callback routine/service address
1176                                     ;           (virtual, not physical address!)
1177                                     ;     [u.sp] = kernel stack, points to
1178                                     ;              user's EIP,CS,EFLAGS,ESP,SS
1179                                     ;              registers.
1180                                     ; OUTPUT:
1181                                     ;     EIP (user) = callback (service) address
1182                                     ;     CS (user) = UCODE
1183                                     ;     EFLAGS (user) = flags before callback
1184                                     ;       ESP (user) = ESP-4 (user, before callback)
1185                                     ;     [ESP](user) = EIP (user) before callback
1186                                     ;
1187                                     ; Note: If CPU was in user mode while entering
1188                                     ;     the timer interrupt service routine,
1189                                     ;     'IRET' will get return to callback routine
1190                                     ;     immediately. If CPU was in system/kernel mode
1191                                     ;     'iret' will get return to system call and
1192                                     ;     then, callback routine will be return address
1193                                     ;     from system call. (User's callback/service code
1194                                     ;     will be able to return to normal return address
1195                                     ;     via an 'retn' at the end.)
1196                                     ;
1197                                     ; Note(**): User's callback service code must be ended
1198                                     ;     with a 'sysrele' sytstem call ! (09/01/2017)
1199                                     ;
1200                                     ;     For example:
1201                                     ;
1202                                     ;     timer_callback:
1203                                     ;         ...
1204                                     ;         inc      dword [time_counter]
1205                                     ;         ...
1206                                     ;         mov eax, 39 ; 'sysrele'
1207                                     ;         int 40h ; TRDOS 386 system call (interrupt)
1208                                     ;
1209                                     ;
1210                                     ;; Note(*): User's callback service code must preserve cpu
1211                                     ;;    flags if it has any instructions which changes
1212                                     ;;    flags in the service code. (25/12/2016)
1213                                     ;;
1214                                     ;;    For example:
1215                                     ;;
1216                                     ;;    timer_callback:
1217                                     ;;        pushf ; save flags
1218                                     ;;        ; this instruction changes zero flag
1219                                     ;;        inc      dword [time_counter]
```

```
1220                                    ;;        popf ; restore flags
1221                                    ;;        retn ; return to normal user code
1222                                    ;;             (which is interrupted by the
1223                                    ;;              timer interput)
1224                                    ;;
1225
1226                                    ; 15/01/2017
1227 000007E8 8B2D[5C030300]            mov   ebp, [u.sp]; kernel's stack, points to EIP (user)
1228 000007EE 895D00                    mov   [ebp], ebx
1229 000007F1 E95CFFFFFF                jmp   timer_int_return
1230
1231                                    ; 15/01/2017
1232                                    ; 13/01/2017
1233                                    ; 19/12/2016
1234                                    ; 06/06/2016
1235                                    ; 23/05/2016
1236                                    ; 22/05/2016
1237                                    ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
1238                                    ; 26/02/2015
1239                                    ; 07/09/2014
1240                                    ; 25/08/2014
1241                                rtc_int:       ; Real Time Clock Interrupt (IRQ 8)
1242                                    ; 22/05/2016
1243 000007F6 1E                        push  ds ; ** ; 23/05/2016
1244 000007F7 50                        push  eax ; *
1245 000007F8 66B81000                  mov   ax, KDATA
1246 000007FC 8ED8                      mov   ds, ax
1247                                    ;
1248 000007FE 8A25[7E520100]            mov   ah, [RTC_2Hz] ;  2 Hz interrupt to 1 Hz function
1249 00000804 80F401                    xor   ah, 1
1250 00000807 8825[7E520100]            mov   [RTC_2Hz], ah ; 1 = 0.5 second, 0 = 1 second
1251 0000080D 753B                      jnz   short rtc_int_return ; half second
1252                                    ; 1 second
1253                                rtc_int_0:
1254                                    ; 22/05/2016
1255 0000080F 58                        pop   eax ; *
1256                                    ;
1257                                    ; 14/10/2015 ('timer_int')
1258                                    ; Here, we are simulating system call entry (for task switch)
1259                                    ; (If multitasking is enabled,
1260                                    ; 'clock' procedure may jump to 'sysrelease')
1261                                    ;push  ds ; ** ; 23/05/2016
1262 00000810 06                        push  es
1263 00000811 0FA0                      push  fs
1264 00000813 0FA8                      push  gs
1265 00000815 60                        pushad  ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi
1266 00000816 66B91000                  mov     cx, KDATA
1267                                      ;mov    ds, cx ; 06/06/2016
1268 0000081A 8EC1                      mov     es, cx
1269 0000081C 8EE1                      mov     fs, cx
1270 0000081E 8EE9                      mov     gs, cx
1271                                    ;
1272 00000820 0F20D9                    mov   ecx, cr3
1273 00000823 890D[5C040300]            mov   [cr3reg], ecx ; save current cr3 register value/content
1274                                    ;
1275 00000829 803D[D4030300]00          cmp   byte [u.t_lock], 0 ; timer lock (callback) status ?
1276 00000830 7711                      ja    short rtc_int_1       ; yes
1277
1278                                    ; 15/01/2017
1279 00000832 3B0D[00520100]            cmp   ecx, [k_page_dir]
1280 00000838 7409                      je    short rtc_int_1
1281
1282 0000083A 8B0D[00520100]            mov   ecx, [k_page_dir]
1283 00000840 0F22D9                    mov   cr3, ecx
1284                                rtc_int_1:
1285                                    ; Timer event (kernel) functions must be performed with
1286                                    ; 1 second intervals - TRDOS 386 (TRDOS v2.0) feature ! -
1287                                    ;
1288                                    ; 25/08/2014
1289 00000843 E81A030000                call  rtc_p  ; 19/05/2016 - major modification
1290
1291                                    ; 23/05/2016
1292 00000848 28E4                      sub   ah, ah ; 0
1293                                    ; 22/05/2016 - TRDOS 386 timer event modifications
1294                                rtc_int_return: ; 19/05/2016
1295                                    ; 22/02/2015 - dsectpm.s
1296                                    ; [ source: http://wiki.osdev.org/RTC ]
1297                                    ; read status register C to complete procedure
1298                                    ;(it is needed to get a next IRQ 8)
1299 0000084A B00C                      mov   al, 0Ch ;
1300 0000084C E670                      out   70h, al ; select register C
1301 0000084E 90                        nop
1302 0000084F E471                      in    al, 71h ; just throw away contents
1303                                    ; 22/02/2015
1304 00000851 B020                      MOV   AL,EOI       ; END OF INTERRUPT
1305                                    ;CLI               ; DISABLE INTERRUPTS TILL STACK CLEARED
1306 00000853 E6A0                      OUT   INTB00,AL    ; FOR CONTROLLER #2
1307
1308                                    ; 23/05/2016
1309 00000855 B020                      MOV   AL,EOI       ; GET END OF INTERRUPT MASK
1310 00000857 FA                        CLI                ; DISABLE INTERRUPTS TILL STACK CLEARED
1311 00000858 E620                      OUT   INTA00,AL    ; END OF INTERRUPT TO 8259 - 1
1312                                    ;
1313                                    ; 23/05/2016
1314 0000085A 20E4                      and   ah, ah
1315 0000085C 0F84B7FEFFFF                jz      rtc_int_2
1316
1317                                    ; ah = 1 (half second)
1318 00000862 58                        pop   eax ; *
1319 00000863 1F                        pop   ds  ; **
1320 00000864 CF                        iretd
1321
```

```
                                ; ///////////////

                                        ; 28/08/2014
                                irq0:
00000865 6A00                             push    dword 0
00000867 EB48                           jmp     short which_irq
                                irq1:
00000869 6A01                             push    dword 1
0000086B EB44                           jmp     short which_irq
                                irq2:
0000086D 6A02                             push    dword 2
0000086F EB40                           jmp     short which_irq
                                irq3:
                                        ; 20/11/2015
                                        ; 24/10/2015
00000871 2EFF15[CFF50000]               call    dword [cs:com2_irq3]
00000878 6A03                           push    dword 3
0000087A EB35                           jmp     short which_irq
                                irq4:
                                        ; 20/11/2015
                                        ; 24/10/2015
0000087C 2EFF15[CBF50000]               call    dword [cs:com1_irq4]
00000883 6A04                             push    dword 4
00000885 EB2A                           jmp     short which_irq
                                irq5:
00000887 6A05                             push    dword 5
00000889 EB26                           jmp     short which_irq
                                irq6:
0000088B 6A06                             push    dword 6
0000088D EB22                           jmp     short which_irq
                                irq7:
0000088F 6A07                             push    dword 7
00000891 EB1E                           jmp     short which_irq
                                irq8:
00000893 6A08                             push    dword 8
00000895 EB1A                           jmp     short which_irq
                                irq9:
00000897 6A09                             push    dword 9
00000899 EB16                           jmp     short which_irq
                                irq10:
0000089B 6A0A                             push    dword 10
0000089D EB12                           jmp     short which_irq
                                irq11:
0000089F 6A0B                             push    dword 11
000008A1 EB0E                           jmp     short which_irq
                                irq12:
000008A3 6A0C                             push    dword 12
000008A5 EB0A                           jmp     short which_irq
                                irq13:
000008A7 6A0D                             push    dword 13
000008A9 EB06                           jmp     short which_irq
                                irq14:
000008AB 6A0E                             push    dword 14
000008AD EB02                           jmp     short which_irq
                                irq15:
000008AF 6A0F                             push    dword 15
                                        ;jmp    short which_irq

                                        ; 22/01/2017
                                        ; 19/10/2015
                                        ; 29/08/2014
                                        ; 21/08/2014
                                which_irq:
000008B1 870424                         xchg    eax, [esp]  ; 28/08/2014
000008B4 53                             push    ebx
000008B5 56                             push    esi
000008B6 57                             push    edi
000008B7 1E                             push    ds
000008B8 06                             push    es
                                        ;
000008B9 88C3                           mov     bl, al
                                        ;
000008BB B810000000                     mov     eax, KDATA
000008C0 8ED8                           mov     ds, ax
000008C2 8EC0                           mov     es, ax
                                        ; 19/10/2015
000008C4 FC                             cld
                                          ; 27/08/2014
000008C5 8105[22100100]A000-              add     dword [scr_row], 0A0h
000008CD 0000
                                        ;
000008CF B417                           mov     ah, 17h      ; blue (1) background,
                                                               ; light gray (7) forecolor
000008D1 8B3D[22100100]                   mov     edi, [scr_row]
000008D7 B049                           mov     al, 'I'
000008D9 66AB                           stosw
000008DB B052                           mov     al, 'R'
000008DD 66AB                           stosw
000008DF B051                           mov     al, 'Q'
000008E1 66AB                           stosw
000008E3 B020                           mov     al, ' '
000008E5 66AB                           stosw
000008E7 88D8                           mov     al, bl
000008E9 3C0A                           cmp     al, 10
000008EB 7208                           jb      short ii1
000008ED B031                           mov     al, '1'
000008EF 66AB                           stosw
000008F1 88D8                           mov     al, bl
000008F3 2C0A                           sub     al, 10
                                ii1:
000008F5 0430                           add     al, '0'
000008F7 66AB                           stosw
```

```
1423 000008F9 B020                      mov    al, ' '
1424 000008FB 66AB                      stosw
1425 000008FD B021                      mov    al, '!'
1426 000008FF 66AB                      stosw
1427 00000901 B020                      mov    al, ' '
1428 00000903 66AB                      stosw
1429                                    ; 23/02/2015
1430 00000905 80FB07                    cmp    bl, 7 ; check for IRQ 8 to IRQ 15
1431 00000908 7604                      jna    ii2
1432                                    ; 22/01/2017
1433 0000090A B020                      mov    al, 20h  ; END OF INTERRUPT COMMAND TO
1434 0000090C E6A0                      out    0A0h, al ; the 2nd 8259
1435                                ii2:
1436 0000090E B020                      mov    al, 20h  ; END OF INTERRUPT COMMAND TO
1437 00000910 E620                      out    20h, al ; the 2nd 8259
1438 00000912 E9CD010000                jmp     iiret
1439                                    ;
1440                                    ; 22/08/2014
1441                                    ;mov   al, 20h ; END OF INTERRUPT COMMAND TO 8259
1442                                    ;out   20h, al      ; 8259 PORT
1443                                    ;
1444                                    ;pop   es
1445                                    ;pop   ds
1446                                    ;pop   edi
1447                                    ;pop   esi
1448                                    ;pop   ebx
1449                                    ;pop   eax
1450                                    ;iret
1451
1452                                    ; 02/04/2015
1453                                    ; 25/08/2014
1454                                exc0:
1455 00000917 6A00                      push       dword 0
1456 00000919 E990000000                jmp     cpu_except
1457                                exc1:
1458 0000091E 6A01                      push       dword 1
1459 00000920 E989000000                jmp     cpu_except
1460                                exc2:
1461 00000925 6A02                      push       dword 2
1462 00000927 E982000000                jmp     cpu_except
1463                                exc3:
1464 0000092C 6A03                      push       dword 3
1465 0000092E EB7E                      jmp     cpu_except
1466                                exc4:
1467 00000930 6A04                      push       dword 4
1468 00000932 EB7A                      jmp     cpu_except
1469                                exc5:
1470 00000934 6A05                      push       dword 5
1471 00000936 EB76                      jmp     cpu_except
1472                                exc6:
1473 00000938 6A06                      push       dword 6
1474 0000093A EB72                      jmp     cpu_except
1475                                exc7:
1476 0000093C 6A07                      push       dword 7
1477 0000093E EB6E                      jmp     cpu_except
1478                                exc8:
1479                                      ; [esp] = Error code
1480 00000940 6A08                      push       dword 8
1481 00000942 EB5C                      jmp     cpu_except_en
1482                                exc9:
1483 00000944 6A09                      push       dword 9
1484 00000946 EB66                      jmp     cpu_except
1485                                exc10:
1486                                      ; [esp] = Error code
1487 00000948 6A0A                      push       dword 10
1488 0000094A EB54                      jmp     cpu_except_en
1489                                exc11:
1490                                      ; [esp] = Error code
1491 0000094C 6A0B                      push       dword 11
1492 0000094E EB50                      jmp     cpu_except_en
1493                                exc12:
1494                                      ; [esp] = Error code
1495 00000950 6A0C                      push       dword 12
1496 00000952 EB4C                      jmp     cpu_except_en
1497                                exc13:
1498                                      ; [esp] = Error code
1499 00000954 6A0D                      push       dword 13
1500 00000956 EB48                      jmp     cpu_except_en
1501                                exc14:
1502                                      ; [esp] = Error code
1503 00000958 6A0E                      push       dword 14
1504 0000095A EB44                    jmp    short cpu_except_en
1505                                exc15:
1506 0000095C 6A0F                      push       dword 15
1507 0000095E EB4E                      jmp     cpu_except
1508                                exc16:
1509 00000960 6A10                      push       dword 16
1510 00000962 EB4A                      jmp     cpu_except
1511                                exc17:
1512                                      ; [esp] = Error code
1513 00000964 6A11                      push       dword 17
1514 00000966 EB38                    jmp    short cpu_except_en
1515                                exc18:
1516 00000968 6A12                      push       dword 18
1517 0000096A EB42                    jmp    short cpu_except
1518                                exc19:
1519 0000096C 6A13                      push       dword 19
1520 0000096E EB3E                    jmp    short cpu_except
1521                                exc20:
1522 00000970 6A14                      push       dword 20
1523 00000972 EB3A                    jmp    short cpu_except
1524                                exc21:
```

```
1525 00000974 6A15                      push       dword 21
1526 00000976 EB36                    jmp    short cpu_except
1527                                exc22:
1528 00000978 6A16                      push       dword 22
1529 0000097A EB32                    jmp    short cpu_except
1530                                exc23:
1531 0000097C 6A17                      push       dword 23
1532 0000097E EB2E                    jmp    short cpu_except
1533                                exc24:
1534 00000980 6A18                      push       dword 24
1535 00000982 EB2A                    jmp    short cpu_except
1536                                exc25:
1537 00000984 6A19                      push       dword 25
1538 00000986 EB26                    jmp    short cpu_except
1539                                exc26:
1540 00000988 6A1A                      push       dword 26
1541 0000098A EB22                    jmp    short cpu_except
1542                                exc27:
1543 0000098C 6A1B                      push       dword 27
1544 0000098E EB1E                    jmp    short cpu_except
1545                                exc28:
1546 00000990 6A1C                      push       dword 28
1547 00000992 EB1A                    jmp    short cpu_except
1548                                exc29:
1549 00000994 6A1D                      push       dword 29
1550 00000996 EB16                    jmp    short cpu_except
1551                                exc30:
1552 00000998 6A1E                      push       dword 30
1553 0000099A EB04                    jmp    short cpu_except_en
1554                                exc31:
1555 0000099C 6A1F                      push       dword 31
1556 0000099E EB0E                      jmp     short cpu_except
1557
1558                                          ; 19/10/2015
1559                                          ; 19/09/2015
1560                                          ; 01/09/2015
1561                                          ; 28/08/2015
1562                                          ; 28/08/2014
1563                                cpu_except_en:
1564 000009A0 87442404                        xchg   eax, [esp+4] ; Error code
1565 000009A4 36A3[78050300]                  mov    [ss:error_code], eax
1566 000009AA 58                              pop    eax  ; Exception number
1567 000009AB 870424                          xchg   eax, [esp]
1568                                                 ; eax = eax before exception
1569                                                 ; [esp] -> exception number
1570                                                 ; [esp+4] -> EIP to return
1571                                          ; 22/01/2017
1572                                          ; 19/10/2015
1573                                          ; 19/09/2015
1574                                          ; 01/09/2015
1575                                          ; 28/08/2015
1576                                          ; 29/08/2014
1577                                          ; 28/08/2014
1578                                          ; 25/08/2014
1579                                          ; 21/08/2014
1580                                cpu_except:  ; CPU Exceptions
1581 000009AE FC                              cld
1582 000009AF 870424                          xchg   eax, [esp]
1583                                                 ; eax = Exception number
1584                                                 ; [esp] = eax (before exception)
1585 000009B2 53                              push   ebx
1586 000009B3 56                              push   esi
1587 000009B4 57                              push   edi
1588 000009B5 1E                              push   ds
1589 000009B6 06                              push   es
1590                                          ; 28/08/2015
1591 000009B7 66BB1000                        mov    bx, KDATA
1592 000009BB 8EDB                            mov    ds, bx
1593 000009BD 8EC3                            mov    es, bx
1594 000009BF 0F20DB                          mov    ebx, cr3
1595 000009C2 53                              push   ebx ; (*) page directory
1596                                          ; 19/10/2015
1597 000009C3 FC                              cld
1598                                          ; 25/03/2015
1599 000009C4 8B1D[00520100]                  mov    ebx, [k_page_dir]
1600 000009CA 0F22DB                          mov    cr3, ebx
1601                                          ; 28/08/2015
1602 000009CD 83F80E                          cmp    eax, 0Eh ; 14, PAGE FAULT
1603 000009D0 750F                            jne    short cpu_except_nfp
1604 000009D2 E87A440000                      call   page_fault_handler
1605 000009D7 21C0                            and    eax, eax
1606 000009D9 0F8401010000                      jz   iiretp ; 01/09/2015
1607 000009DF B00E                            mov    al, 0Eh ; 14
1608                                cpu_except_nfp:
1609                                          ; 23/08/2016
1610 000009E1 803D[C25E0000]03                cmp    byte [CRT_MODE], 3
1611 000009E8 7409                            je     short cpu_except_mode_3
1612 000009EA 50                              push   eax
1613 000009EB B003                            mov    al, 3
1614 000009ED E8730B0000                      call   _set_mode
1615 000009F2 58                              pop    eax
1616                                cpu_except_mode_3:
1617                                          ; 02/04/2015
1618 000009F3 BB[42060000]                    mov    ebx, hang
1619 000009F8 875C241C                        xchg   ebx, [esp+28]
1620                                                 ; EIP (points to instruction which faults)
1621                                                 ; New EIP (hang)
1622 000009FC 891D[7C050300]                  mov    [FaultOffset], ebx
1623 00000A02 C744242008000000                mov    dword [esp+32], KCODE ; kernel's code segment
1624 00000A0A 814C242400020000                or     dword [esp+36], 200h ; enable interrupts (set IF)
1625                                          ;
1626 00000A12 88C4                            mov    ah, al
```

```
1627 00000A14 240F                      and    al, 0Fh
1628 00000A16 3C09                      cmp    al, 9
1629 00000A18 7602                      jna    short h1ok
1630 00000A1A 0407                      add    al, 'A'-':'
1631                                h1ok:
1632 00000A1C C0EC04                    shr    ah, 4
1633 00000A1F 80FC09                    cmp    ah, 9
1634 00000A22 7603                      jna    short h2ok
1635 00000A24 80C407                    add    ah, 'A'-':'
1636                                h2ok:
1637 00000A27 86E0                      xchg   ah, al
1638 00000A29 66053030                  add    ax, '00'
1639 00000A2D 66A3[7C120100]            mov    [excnstr], ax
1640                                    ;
1641                                    ; 29/08/2014
1642 00000A33 A1[7C050300]              mov    eax, [FaultOffset]
1643 00000A38 51                        push   ecx
1644 00000A39 52                        push   edx
1645 00000A3A 89E3                      mov    ebx, esp
1646                                    ; 28/08/2015
1647 00000A3C B910000000                mov    ecx, 16         ; divisor value to convert binary number
1648                                                   ; to hexadecimal string
1649                                    ;mov   ecx, 10          ; divisor to convert
1650                                                  ; binary number to decimal string
1651                                b2d1:
1652 00000A41 31D2                      xor    edx, edx
1653 00000A43 F7F1                      div    ecx
1654 00000A45 6652                      push   dx
1655 00000A47 39C8                      cmp    eax, ecx
1656 00000A49 73F6                      jnb    short b2d1
1657 00000A4B BF[87120100]              mov    edi, EIPstr ; EIP value
1658                                                  ; points to instruction which faults
1659                                    ; 28/08/2015
1660 00000A50 89C2                      mov    edx, eax
1661                                b2d2:
1662                                    ;add   al, '0'
1663 00000A52 8A82[1B330000]            mov    al, [edx+hexchrs]
1664 00000A58 AA                        stosb             ; write hexadecimal digit to its place
1665 00000A59 39E3                      cmp    ebx, esp
1666 00000A5B 7606                      jna    short b2d3
1667 00000A5D 6658                      pop    ax
1668 00000A5F 88C2                      mov    dl, al
1669 00000A61 EBEF                      jmp    short b2d2
1670                                b2d3:
1671 00000A63 B068                      mov    al, 'h' ; 28/08/2015
1672 00000A65 AA                        stosb
1673 00000A66 B020                      mov    al, 20h          ; space
1674 00000A68 AA                        stosb
1675 00000A69 30C0                      xor    al, al     ; to do it an ASCIIZ string
1676 00000A6B AA                        stosb
1677                                    ;
1678 00000A6C 5A                        pop    edx
1679 00000A6D 59                        pop    ecx
1680                                    ;
1681 00000A6E B44F                      mov    ah, 4Fh      ; red (4) background,
1682                                                  ; white (F) forecolor
1683 00000A70 BE[6C120100]              mov    esi, exc_msg ; message offset
1684                                    ;
1685                                    ; 20/01/2017 (!cpu exception!)
1686                                    ;
1687 00000A75 8105[22100100]A000-         add    dword [scr_row], 0A0h
1687 00000A7D 0000
1688 00000A7F 8B3D[22100100]              mov    edi, [scr_row]
1689                                    ;
1690 00000A85 C605[5B030300]00          mov    byte [sysflg], 0  ; system mode
1691 00000A8C FB                          sti
1692                                    ;
1693 00000A8D E8EFFBFFFF                call   printk
1694                                    ;
1695 00000A92 B410                      mov    ah, 10h
1696 00000A94 E87D010000                call   int16h ; getc
1697                                    ;
1698 00000A99 B003                      mov    al, 3
1699 00000A9B E8C50A0000                call   _set_mode
1700                                    ;
1701 00000AA0 B801000000                mov    eax, 1
1702 00000AA5 E990BB0000                jmp    sysexit ; terminate process !!!
1703
1704                                    ; 22/01/2017
1705                                    ; 18/04/2016
1706                                    ; 28/08/2015
1707                                    ; 23/02/2015
1708                                    ; 20/08/2014
1709                                ignore_int:
1710 00000AAA 50                        push   eax
1711 00000AAB 53                        push   ebx ; 23/02/2015
1712 00000AAC 56                        push   esi
1713 00000AAD 57                        push   edi
1714 00000AAE 1E                        push   ds
1715 00000AAF 06                        push   es
1716                                    ; 18/04/2016
1717 00000AB0 66B81000                  mov    ax, KDATA
1718 00000AB4 8ED8                      mov    ds, ax
1719 00000AB6 8EC0                      mov    es, ax
1720                                    ; 28/08/2015
1721 00000AB8 0F20D8                    mov    eax, cr3
1722 00000ABB 50                        push   eax ; (*) page directory
1723                                    ;
1724 00000ABC B467                      mov    ah, 67h      ; brown (6) background,
1725                                                  ; light gray (7) forecolor
1726 00000ABE BE[34110100]              mov    esi, int_msg ; message offset
1727                                piemsg:
```

```
                                        ; 27/08/2014
00000AC3 8105[22100100]A000-            add     dword [scr_row], 0A0h
00000ACB 0000
00000ACD 8B3D[22100100]                 mov     edi, [scr_row]
                                        ;
00000AD3 E8A9FBFFFF                   call  printk
                                      ;
                                      ; 23/02/2015
00000AD8 B020                         mov   al, 20h  ; END OF INTERRUPT COMMAND TO
00000ADA E6A0                         out   0A0h, al ; the 2nd 8259
                                      ; 22/08/2014
00000ADC B020                         mov   al, 20h ; END OF INTERRUPT COMMAND TO 8259
00000ADE E620                         out   20h, al      ; 8259 PORT
                                    iiretp:
                                      ; 22/01/2017
                                      ; 01/09/2015
                                      ; 28/08/2015
00000AE0 58                           pop   eax ; (*) page directory
00000AE1 0F22D8                       mov   cr3, eax
                                    iiret:
00000AE4 07                           pop   es
00000AE5 1F                           pop   ds
00000AE6 5F                           pop   edi
00000AE7 5E                           pop   esi
00000AE8 5B                           pop   ebx ; 29/08/2014
00000AE9 58                           pop   eax
00000AEA CF                           iretd
                                      ; 23/05/2016
                                      ; 22/08/2014
                                      ; IBM PC/AT BIOS source code ----- 10/06/85 (bios.asm)
                                      ; (INT 1Ah)
                                      ;; Linux (v0.12) source code (main.c) by Linus Torvalds (1991)
                                    time_of_day:
00000AEB E8ED500000                   call  UPD_IPR                   ; WAIT TILL UPDATE NOT IN PROGRESS
00000AF0 726F                           jc      short time_of_day_retn ; 23/05/2016
00000AF2 B000                         mov   al, CMOS_SECONDS
00000AF4 E8FF500000                   call  CMOS_READ
00000AF9 A2[70520100]                 mov   [time_seconds], al
00000AFE B002                         mov   al, CMOS_MINUTES
00000B00 E8F3500000                   call  CMOS_READ
00000B05 A2[71520100]                 mov   [time_minutes], al
00000B0A B004                         mov   al, CMOS_HOURS
00000B0C E8E7500000                   call  CMOS_READ
00000B11 A2[72520100]                   mov     [time_hours], al
00000B16 B006                         mov   al, CMOS_DAY_WEEK
00000B18 E8DB500000                   call  CMOS_READ
00000B1D A2[73520100]                 mov   [date_wday], al
00000B22 B007                         mov   al, CMOS_DAY_MONTH
00000B24 E8CF500000                   call  CMOS_READ
00000B29 A2[74520100]                 mov   [date_day], al
00000B2E B008                         mov   al, CMOS_MONTH
00000B30 E8C3500000                   call  CMOS_READ
00000B35 A2[75520100]                 mov   [date_month], al
00000B3A B009                         mov   al, CMOS_YEAR
00000B3C E8B7500000                   call  CMOS_READ
00000B41 A2[76520100]                 mov   [date_year], al
00000B46 B032                         mov   al, CMOS_CENTURY
00000B48 E8AB500000                   call  CMOS_READ
00000B4D A2[77520100]                 mov   [date_century], al
                                      ;
00000B52 B000                         mov   al, CMOS_SECONDS
00000B54 E89F500000                   call  CMOS_READ
00000B59 3A05[70520100]               cmp   al, [time_seconds]
00000B5F 758A                         jne   short time_of_day
                                    time_of_day_retn:
00000B61 C3                           retn
                                      ; 15/01/2017
                                      ; 10/06/2016
                                      ; 07/06/2016
                                      ; 06/06/2016
                                      ; 23/05/2016
                                    rtc_p:
00000B62 B101                         mov   cl, 1 ; 15/01/2017
00000B64 EB02                         jmp   short rtc_p0
                                    u_timer:
                                      ; Timer Events with 18.2 Hz Timer Ticks
                                      ; (and also timer events with RTC seconds)
00000B66 28C9                         sub   cl, cl ; mov cl, 0 ; 15/01/2017
                                    rtc_p0:
                                      ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
                                      ; Major Modification:
                                      ; Check and Perform Timer Events (for RTC)
                                      ; 25/08/2014 - 07/09/2014
                                      ; Retro UNIX 386 v1:
                                      ; Print Real Time Clock content
                                      ; 15/01/2017
00000B68 880D[945F0100]               mov   byte [priority], cl ; 0 or 1 (not 2)
00000B6E 8A2D[975F0100]               mov   ch, [timer_events]
00000B74 20ED                         and   ch, ch
00000B76 7420                         jz    short rtc_p3
00000B78 BE[60040300]                 mov   esi, timer_set  ; beginning address of
                                                            ; timer events space
                                    rtc_p1:
00000B7D 8B06                         mov   eax, [esi]
00000B7F 20C0                         and   al, al ; 0 = free, >0 = process no.
00000B81 7416                         jz    short rtc_p4
                                      ;
```

```
1829 00000B83 C1C810                    ror    eax, 16
1830                                    ; ah = response value, al = interrupt type
1831                                    ; 15/01/2017
1832                                    ; cl = interrupt source
1833                                    ;       1 = RTC, 0 = PIT
1834 00000B86 38C8                      cmp    al, cl
1835 00000B88 750A                      jne    short rtc_p2 ; not as requested or undefined !
1836 00000B8A 3C01                      cmp    al, 1 ; 1 ; RTC interrupt ?
1837 00000B8C 7410                      je     short rtc_p5 ; yes, check for response
1838                                    ; 06/06/2016 - 18.2 Hz Timer Ticks
1839 00000B8E 836E080A                  sub    dword [esi+8], 10 ; 1 tick = 10
1840 00000B92 7613                      jna    short rtc_p6  ; continue for responding
1841                                rtc_p2:
1842                                    ; 15/01/2017 (cl -> ch)
1843                                    ; 07/06/2016
1844 00000B94 FECD                      dec    ch    ; remain count of timer events
1845 00000B96 7501                      jnz    short rtc_p4
1846                                rtc_p3:
1847 00000B98 C3                        retn
1848                                rtc_p4:
1849                                    ;cmp   esi, timer_set + 240 ; 15*16 (last event)
1850                                    ;jnb   short rtc_p3 ; end of timer event space
1851 00000B99 83C610                    add    esi, 16 ; next timer event
1852 00000B9C EBDF                      jmp    short rtc_p1
1853                                rtc_p5:
1854                                    ; current timer count ; 06/06/2016 (182)
1855 00000B9E 816E08B6000000            sub    dword [esi+8], 182 ; 1 second (10*18.2)
1856 00000BA5 77ED                      ja     short rtc_p2  ; check for the next
1857                                rtc_p6:
1858                                    ; it is the time of response!
1859 00000BA7 8B5E04                    mov    ebx, [esi+4] ; set (count limit) value
1860 00000BAA 895E08                    mov    [esi+8], ebx ; reset count down value
1861                                                         ; to count limit
1862                                    ; 19/12/2016
1863                                    ; 10/12/2016 - timer callback modification
1864 00000BAD 8B7E0C                    mov    edi, [esi+12] ; response (or callback) address
1865 00000BB0 807E0100                  cmp    byte [esi+1], 0 ; >0 = callback
1866 00000BB4 762A                      jna    short rtc_p8
1867
1868                                    ; timer callback !
1869 00000BB6 0FB61E                    movzx  ebx, byte [esi] ; process number (>0)
1870 00000BB9 89D8                      mov    eax, ebx
1871 00000BBB C0E302                    shl    bl, 2 ; *4
1872 00000BBE 89BB[0C010300]            mov    [ebx+p.tcb-4], edi ; user's callback service addr
1873 00000BC4 3A05[B3030300]            cmp    al, [u.uno]
1874 00000BCA 7521                      jne    short rtc_p9
1875 00000BCC 893D[D0030300]            mov    [u.tcb], edi
1876                                rtc_p7:
1877                                    ; 15/01/2017
1878 00000BD2 B002                      mov    al, 2
1879 00000BD4 A2[945F0100]              mov    [priority], al ; 2
1880                                    ; 10/01/2017
1881                                    ;mov   byte [u.pri], 2
1882 00000BD9 A2[A9030300]              mov    [u.pri], al ; 2
1883 00000BDE EBB4                      jmp    short rtc_p2
1884                                rtc_p8:
1885                                    ; response address is physical address of
1886                                    ; the program's response (signal return) byte
1887                                    ; 06/06/2016
1888                                    ;mov   edi, [esi+12] ; response address
1889 00000BE0 8827                      mov    [edi], ah     ; response value
1890                                    ;
1891 00000BE2 C1C010                    rol    eax, 16
1892                                    ; 15/01/2017
1893 00000BE5 3A05[B3030300]            cmp    al, [u.uno] ; running process ?
1894 00000BEB 74E5                      je     short rtc_p7
1895                                rtc_p9:
1896                                    ; al = process number  ; 10/06/2016
1897 00000BED B202                      mov    dl, 2 ; priority, 2 = event (high)
1898 00000BEF E8D1E60000                call   set_run_sequence ; 19/05/2016
1899 00000BF4 EB9E                      jmp    short rtc_p2 ; 10/06/2016
1900
1901
1902                                ; Default IRQ 7 handler against spurious IRQs (from master PIC)
1903                                ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
1904                                default_irq7:
1905 00000BF6 6650                        push   ax
1906 00000BF8 B00B                      mov    al, 0Bh  ; In-Service register
1907 00000BFA E620                      out    20h, al
1908 00000BFC EB00                        jmp short $+2
1909 00000BFE EB00                      jmp short $+2
1910 00000C00 E420                      in     al, 20h
1911 00000C02 2480                      and    al, 80h ; bit 7 (is it real IRQ 7 or fake?)
1912 00000C04 7404                        jz      short irq7_iret ; Fake (spurious) IRQ, do not send EOI
1913 00000C06 B020                        mov     al, 20h ; EOI
1914 00000C08 E620                      out    20h, al
1915                                irq7_iret:
1916 00000C0A 6658                      pop    ax
1917 00000C0C CF                        iretd
1918
1919                                bcd_to_ascii:
1920                                    ; 25/08/2014
1921                                    ; INPUT ->
1922                                    ;      al = Packed BCD number
1923                                    ; OUTPUT ->
1924                                    ;      ax  = ASCII word/number
1925                                    ;
1926                                    ; Erdogan Tan - 1998 (proc_hex) - TRDOS.ASM (2004-2011)
1927                                    ;
1928 00000C0D D410                      db 0D4h,10h                    ; Undocumented inst. AAM
1929                                                                   ; AH = AL / 10h
1930                                                                   ; AL = AL MOD 10h
```

```
1931 00000C0F 660D3030                  or ax,'00'                      ; Make it ASCII based
1932
1933 00000C13 86E0                        xchg ah, al
1934
1935 00000C15 C3                        retn
1936
1937
1938                                 %include 'keyboard.s' ; 07/03/2015
   1                                 <1> ; ****************************************************************************
   2                                 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - keyboard.s
   3                                 <1> ; ----------------------------------------------------------------------------
   4                                 <1> ; Last Update: 15/01/2017
   5                                 <1> ; ----------------------------------------------------------------------------
   6                                 <1> ; Beginning: 17/01/2016
   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> ; keyboard.inc (17/10/2015)
  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 - KEYBOARD.INC
  20                                 <1> ; Last Modification: 17/10/2015
  21                                 <1> ;                (Keyboard Data is in 'KYBDATA.INC')
  22                                 <1> ;
  23                                 <1> ; ///////// KEYBOARD FUNCTIONS (PROCEDURES) ///////////////
  24                                 <1>
  25                                 <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
  26                                 <1>
  27                                 <1> ; 03/12/2014
  28                                 <1> ; 26/08/2014
  29                                 <1> ; KEYBOARD I/O
  30                                 <1> ; (INT_16h - Retro UNIX 8086 v1 - U9.ASM, 30/06/2014)
  31                                 <1>
  32                                 <1> ;NOTE: 'k0' to 'k7' are name of OPMASK registers.
  33                                 <1> ;     (The reason of using '_k' labels!!!) (27/08/2014)
  34                                 <1> ;NOTE: 'NOT' keyword is '~' unary operator in NASM.
  35                                 <1> ;     ('NOT LC_HC' --> '~LC_HC') (bit reversing operator)
  36                                 <1>
  37                                 <1> int16h:     ; 30/06/2015
  38                                 <1> ;getc:
  39 00000C16 9C                     <1>       pushfd ; 28/08/2014
  40 00000C17 0E                     <1>       push   cs
  41 00000C18 E801000000             <1>       call   KEYBOARD_IO_1 ; getc_int
  42 00000C1D C3                     <1>       retn
  43                                 <1>
  44                                 <1> getc_int:
  45                                 <1>       ; 28/02/2015
  46                                 <1>       ; 03/12/2014 (derivation from pc-xt-286 bios source code -1986-,
  47                                 <1>       ;           instead of pc-at bios - 1985-)
  48                                 <1>       ; 28/08/2014 (_k1d)
  49                                 <1>       ; 30/06/2014
  50                                 <1>       ; 03/03/2014
  51                                 <1>       ; 28/02/2014
  52                                 <1>       ; Derived from "KEYBOARD_IO_1" procedure of IBM "pc-xt-286"
  53                                 <1>       ; rombios source code (21/04/1986)
  54                                 <1>       ;      'keybd.asm', INT 16H, KEYBOARD_IO
  55                                 <1>       ;
  56                                 <1>       ; KYBD --- 03/06/86  KEYBOARD BIOS
  57                                 <1>       ;
  58                                 <1>       ;--- INT 16 H ----------------------------------------------------------------
  59                                 <1>       ; KEYBOARD I/O                                                          :
  60                                 <1>       ;     THESE ROUTINES PROVIDE READ KEYBOARD SUPPORT                       :
  61                                 <1>       ; INPUT                                                                 :
  62                                 <1>       ;     (AH)= 00H  READ THE NEXT ASCII CHARACTER ENTERED FROM THE KEYBOARD,   :
  63                                 <1>       ;               RETURN THE RESULT IN (AL), SCAN CODE IN (AH).             :
  64                                 <1>       ;               THIS IS THE COMPATIBLE READ INTERFACE, EQUIVALENT TO THE   :
  65                                 <1>       ;                  STANDARD PC OR PCAT KEYBOARD                           :
  66                                 <1>       ;-------------------------------------------------------------------------:
  67                                 <1>       ;     (AH)= 01H  SET THE ZERO FLAG TO INDICATE IF AN ASCII CHARACTER IS     :
  68                                 <1>       ;               AVAILABLE TO BE READ FROM THE KEYBOARD BUFFER.          :
  69                                 <1>       ;               (ZF)= 1 -- NO CODE AVAILABLE                             :
  70                                 <1>       ;               (ZF)= 0 -- CODE IS AVAILABLE  (AX)= CHARACTER             :
  71                                 <1>       ;               IF (ZF)= 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ IS :
  72                                 <1>       ;               IN (AX), AND THE ENTRY REMAINS IN THE BUFFER.             :
  73                                 <1>       ;               THIS WILL RETURN ONLY PC/PCAT KEYBOARD COMPATIBLE CODES    :
  74                                 <1>       ;-------------------------------------------------------------------------:
  75                                 <1>       ;     (AH)= 02H  RETURN THE CURRENT SHIFT STATUS IN AL REGISTER             :
  76                                 <1>       ;               THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE        :
  77                                 <1>       ;               EQUATES FOR @KB_FLAG                                      :
  78                                 <1>       ;-------------------------------------------------------------------------:
  79                                 <1>       ;     (AH)= 03H  SET TYPAMATIC RATE AND DELAY                             :
  80                                 <1>       ;           (AL) = 05H                                                    :
  81                                 <1>       ;           (BL) = TYPAMATIC RATE (BITS 5 - 7 MUST BE RESET TO 0)         :
  82                                 <1>       ;                                                                       :
  83                                 <1>       ;                  REGISTER     RATE      REGISTER     RATE              :
  84                                 <1>       ;                   VALUE     SELECTED     VALUE     SELECTED            :
  85                                 <1>       ;                  ---------------------------------------            :
  86                                 <1>       ;                    00H        30.0        10H        7.5               :
  87                                 <1>       ;                    01H        26.7        11H        6.7               :
  88                                 <1>       ;                    02H        24.0        12H        6.0               :
  89                                 <1>       ;                    03H        21.8        13H        5.5               :
  90                                 <1>       ;                    04H        20.0        14H        5.0               :
  91                                 <1>       ;                    05H        18.5        15H        4.6               :
  92                                 <1>       ;                    06H        17.1        16H        4.3               :
  93                                 <1>       ;                    07H        16.0        17H        4.0               :
  94                                 <1>       ;                    08H        15.0        18H        3.7               :
```

```
                        <1>     ;                 09H       13.3       19H       3.3                :
                        <1>     ;                 0AH       12.0       1AH       3.0                :
                        <1>     ;                 0BH       10.9       1BH       2.7                :
                        <1>       ;               0CH       10.0       1CH       2.5                :
                        <1>     ;                 0DH        9.2       1DH       2.3                :
                        <1>     ;                 0EH        8.6       1EH       2.1                :
                        <1>     ;                 0FH        8.0       1FH       2.0                :
                        <1>     ;                                                        :
                        <1>     ;           (BH) = TYPAMATIC DELAY  (BITS 2 - 7 MUST BE RESET TO 0)       :
                        <1>     ;                                                      :
                        <1>     ;                   REGISTER     DELAY                                  :
                        <1>     ;                    VALUE       VALUE                                  :
                        <1>     ;                   ------------------                                  :
                        <1>     ;                 00H        250 ms                                  :
                        <1>     ;                 01H        500 ms                                  :
                        <1>     ;                 02H        750 ms                                  :
                        <1>     ;                 03H       1000 ms                                  :
                        <1>     ;-------------------------------------------------------------------------:
                        <1>     ;     (AH)= 05H  PLACE ASCII CHARACTER/SCAN CODE COMBINATION IN KEYBOARD    :
                        <1>     ;               BUFFER AS IF STRUCK FROM KEYBOARD                       :
                        <1>     ;               ENTRY:  (CL) = ASCII CHARACTER                        :
                        <1>     ;                       (CH) = SCAN CODE                                :
                        <1>     ;               EXIT:   (AH) = 00H = SUCCESSFUL OPERATION               :
                        <1>     ;                       (AL) = 01H = UNSUCCESSFUL - BUFFER FULL         :
                        <1>     ;               FLAGS:  CARRY IF ERROR                                  :
                        <1>     ;-------------------------------------------------------------------------:

                        <1>     ;     (AH)= 10H  EXTENDED READ INTERFACE FOR THE ENHANCED KEYBOARD,         :
                        <1>     ;               OTHERWISE SAME AS FUNCTION AH=0                         :
                        <1>     ;-------------------------------------------------------------------------:
                        <1>     ;     (AH)= 11H  EXTENDED ASCII STATUS FOR THE ENHANCED KEYBOARD,           :
                        <1>     ;               OTHERWISE SAME AS FUNCTION AH=1                         :
                        <1>     ;-------------------------------------------------------------------------:
                        <1>     ;     (AH)= 12H  RETURN THE EXTENDED SHIFT STATUS IN AX REGISTER            :
                        <1>     ;               AL = BITS FROM KB_FLAG, AH = BITS FOR LEFT AND RIGHT       :
                        <1>     ;               CTL AND ALT KEYS FROM KB_FLAG_1 AND KB_FLAG_3              :
                        <1>     ; OUTPUT                                                        :
                        <1>     ;     AS NOTED ABOVE, ONLY (AX) AND FLAGS CHANGED                       :
                        <1>     ;     ALL REGISTERS RETAINED                                             :
                        <1>     ;-------------------------------------------------------------------------
                        <1>
                        <1> ; 15/01/2017
                        <1> ; 14/01/2017
                        <1> ; 02/01/2017
                        <1> ; 29/05/2016
                        <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                        <1> int32h:  ; Keyboard BIOS
                        <1>
                        <1> KEYBOARD_IO_1:
                        <1>     ;sti                    ; INTERRUPTS BACK ON
                        <1>     ; 29/05/2016
00000C1E 80642408BE     <1>       and     byte [esp+8], 10111110b ; clear zero flag and cary flag
                        <1>     ;
00000C23 1E             <1>     push  ds                ; SAVE CURRENT DS
00000C24 53             <1>     push  ebx               ; SAVE BX TEMPORARILY
                        <1>     ;push  ecx              ; SAVE CX TEMPORARILY
00000C25 66BB1000       <1>       mov     bx, KDATA
00000C29 8EDB           <1>     mov   ds, bx            ; PUT SEGMENT VALUE OF DATA AREA INTO DS
                        <1>
                        <1>     ; 14/01/2017
00000C2B 8B1C24         <1>     mov   ebx, [esp]
                        <1>     ;; 15/01/2017
                        <1>     ; 02/01/2017
                        <1>     ;;mov  byte [intflg], 32h  ; keyboard interrupt
00000C2E FB             <1>     sti
                        <1>     ;
                        <1>
00000C2F 08E4           <1>     or    ah, ah            ; CHECK FOR (AH)= 00H
00000C31 743A           <1>     jz    short _K1         ; ASCII_READ
00000C33 FECC           <1>     dec   ah                    ; CHECK FOR (AH)= 01H
00000C35 7453           <1>       jz      short _K2             ; ASCII_STATUS
00000C37 FECC           <1>     dec   ah                ; CHECK FOR (AH)= 02H
00000C39 0F8494000000   <1>       jz      _K3                   ; SHIFT STATUS
00000C3F FECC           <1>     dec   ah                ; CHECK FOR (AH)= 03H
00000C41 0F8493000000   <1>       jz      _K300                 ; SET TYPAMATIC RATE/DELAY
00000C47 80EC02         <1>     sub   ah, 2             ; CHECK FOR (AH)= 05H
00000C4A 0F84BC000000   <1>       jz      _K500                 ; KEYBOARD WRITE
                        <1> _KIO1:
00000C50 80EC0B         <1>     sub   ah, 11            ; AH =  10H
00000C53 740C           <1>     jz    short _K1E        ; EXTENDED ASCII READ
00000C55 FECC           <1>     dec   ah                ; CHECK FOR (AH)= 11H
00000C57 7422           <1>     jz    short _K2E        ; EXTENDED_ASCII_STATUS
00000C59 FECC           <1>     dec   ah                ; CHECK FOR (AH)= 12H
00000C5B 7458           <1>     jz    short _K3E        ; EXTENDED_SHIFT_STATUS
                        <1> _KIO_EXIT:
                        <1>     ; 02/01/2017
00000C5D FA             <1>     cli
                        <1>     ;;mov  byte [intflg], 0 ;; 15/01/2017
                        <1>     ;
                        <1>     ;pop  ecx               ; RECOVER REGISTER
00000C5E 5B             <1>     pop   ebx               ; RECOVER REGISTER
00000C5F 1F             <1>     pop   ds                ; RECOVER SEGMENT
00000C60 CF             <1>     iretd                   ; INVALID COMMAND, EXIT
                        <1>
                        <1>     ;-----  ASCII CHARACTER
                        <1> _K1E:
00000C61 E8D3000000     <1>     call  _K1S              ; GET A CHARACTER FROM THE BUFFER (EXTENDED)
00000C66 E848010000     <1>     call  _KIO_E_XLAT       ; ROUTINE TO XLATE FOR EXTENDED CALLS
00000C6B EBF0           <1>     jmp   short _KIO_EXIT       ; GIVE IT TO THE CALLER
                        <1> _K1:
00000C6D E8C7000000     <1>     call  _K1S              ; GET A CHARACTER FROM THE BUFFER
00000C72 E847010000     <1>     call  _KIO_S_XLAT       ; ROUTINE TO XLATE FOR STANDARD CALLS
```

```
196 00000C77 72F4              <1>       jc    short _K1           ; CARRY SET MEANS TROW CODE AWAY
197                            <1> _K1A:
198 00000C79 EBE2              <1>       jmp   short _KIO_EXIT         ; RETURN TO CALLER
199                            <1>
200                            <1>       ;----- ASCII STATUS
201                            <1> _K2E:
202 00000C7B E804010000        <1>       call  _K2S                ; TEST FOR CHARACTER IN BUFFER (EXTENDED)
203 00000C80 7420              <1>       jz    short _K2B          ; RETURN IF BUFFER EMPTY
204 00000C82 9C                <1>       pushf                     ; SAVE ZF FROM TEST
205 00000C83 E82B010000        <1>       call  _KIO_E_XLAT         ; ROUTINE TO XLATE FOR EXTENDED CALLS
206 00000C88 EB17              <1>       jmp   short _K2A           ; GIVE IT TO THE CALLER
207                            <1> _K2:
208 00000C8A E8F5000000        <1>       call  _K2S                ; TEST FOR CHARACTER IN BUFFER
209 00000C8F 7411              <1>       jz    short _K2B          ; RETURN IF BUFFER EMPTY
210 00000C91 9C                <1>       pushf                     ; SAVE ZF FROM TEST
211 00000C92 E827010000        <1>       call  _KIO_S_XLAT         ; ROUTINE TO XLATE FOR STANDARD CALLS
212 00000C97 7308              <1>       jnc   short _K2A           ; CARRY CLEAR MEANS PASS VALID CODE
213 00000C99 9D                <1>       popf                      ; INVALID CODE FOR THIS TYPE OF CALL
214 00000C9A E89A000000        <1>       call  _K1S                ; THROW THE CHARACTER AWAY
215 00000C9F EBE9              <1>       jmp   short _K2           ; GO LOOK FOR NEXT CHAR, IF ANY
216                            <1> _K2A:
217 00000CA1 9D                <1>       popf                      ; RESTORE ZF FROM TEST
218                            <1> _K2B:
219                            <1>       ; 02/01/2017
220 00000CA2 FA                <1>       cli
221                            <1>       ;; mov byte [intflg], 0 ;; 15/01/2017
222                            <1>       ;
223                            <1>       ;pop  ecx                 ; RECOVER REGISTER
224 00000CA3 5B                <1>       pop   ebx                 ; RECOVER REGISTER
225 00000CA4 1F                <1>       pop   ds                  ; RECOVER SEGMENT
226                            <1>       ; (*) 29/05/2016
227                            <1>       ; (*) retf 4              ; THROW AWAY (e)FLAGS
228 00000CA5 7208              <1>       jc    short _k2d
229 00000CA7 7505              <1>       jnz   short _k2c
230 00000CA9 804C240840        <1>       or    byte [esp+8], 01000000b   ; set zero flag bit of eflags register
231                            <1> _k2c:
232 00000CAE CF                <1>       iretd
233                            <1> _k2d:
234                            <1>       ; 29/05/2016 -set carry flag on stack-
235                            <1>       ; [esp] = EIP
236                            <1>       ; [esp+4] = CS
237                            <1>       ; [esp+8] = E-FLAGS
238 00000CAF 804C240801        <1>       or    byte [esp+8], 1  ; set carry bit of eflags register
239                            <1>       ; [esp+12] = ESP (user)
240                            <1>       ; [esp+16] = SS (User)
241 00000CB4 CF                <1>       iretd
242                            <1>
243                            <1>
244                            <1>       ; (*) 29/05/2016 - 'ref 4' intruction causes to stack fault
245                            <1>       ; (OUTER-PRIVILEGE-LEVEL)
246                            <1>       ; INTEL 80386 PROGRAMMER'S REFERENCE MANUAL 1986
247                            <1>       ; // RETF instruction:
248                            <1>       ;
249                            <1>       ; IF OperandMode=32 THEN
250                            <1>       ;    Load CS:EIP from stack;
251                            <1>       ;    Set CS RPL to CPL;
252                            <1>       ;    Increment eSP by 8 plus the immediate offset if it exists;
253                            <1>       ;    Load SS:eSP from stack;
254                            <1>       ; ELSE (* OperandMode=16 *)
255                            <1>       ;    Load CS:IP from stack;
256                            <1>       ;    Set CS RPL to CPL;
257                            <1>       ;    Increment eSP by 4 plus the immediate offset if it exists;
258                            <1>       ;    Load SS:eSP from stack;
259                            <1>       ; FI;
260                            <1>       ;
261                            <1>       ; //
262                            <1>
263                            <1>       ;----- SHIFT STATUS
264                            <1> _K3E:                                  ; GET THE EXTENDED SHIFT STATUS FLAGS
265 00000CB5 8A25[8E5E0000]    <1>       mov   ah, [KB_FLAG_1]           ; GET SYSTEM SHIFT KEY STATUS
266 00000CBB 80E404            <1>       and   ah, SYS_SHIFT       ; MASK ALL BUT SYS KEY BIT
267                            <1>       ;mov  cl, 5               ; SHIFT THEW SYSTEMKEY BIT OVER TO
268                            <1>       ;shl  ah, cl              ; BIT 7 POSITION
269 00000CBE C0E405            <1>         shl ah, 5
270 00000CC1 A0[8E5E0000]      <1>       mov   al, [KB_FLAG_1]           ; GET SYSTEM SHIFT STATES BACK
271 00000CC6 2473              <1>       and   al, 01110011b       ; ELIMINATE SYS SHIFT, HOLD_STATE AND INS_SHIFT
272 00000CC8 08C4              <1>       or    ah, al                 ; MERGE REMAINING BITS INTO AH
273 00000CCA A0[905E0000]      <1>       mov   al, [KB_FLAG_3]           ; GET RIGHT CTL AND ALT
274 00000CCF 240C              <1>       and   al, 00001100b       ; ELIMINATE LC_E0 AND LC_E1
275 00000CD1 08C4              <1>       or    ah, al              ; OR THE SHIFT FLAGS TOGETHER
276                            <1> _K3:
277 00000CD3 A0[8D5E0000]      <1>       mov   al, [KB_FLAG]       ; GET THE SHIFT STATUS FLAGS
278                            <1>       ;jmp  short _KIO_EXIT           ; RETURN TO CALLER
279 00000CD8 EB83              <1>       jmp   _KIO_EXIT
280                            <1>
281                            <1>       ;----- SET TYPAMATIC RATE AND DELAY
282                            <1> _K300:
283 00000CDA 3C05              <1>       cmp   al, 5               ; CORRECT FUNCTION CALL?
284                            <1>       ;jne  short _KIO_EXIT           ; NO, RETURN
285 00000CDC 0F857BFFFFFF      <1>       jne   _KIO_EXIT
286 00000CE2 F6C3E0            <1>       test  bl, 0E0h            ; TEST FOR OUT-OF-RANGE RATE
287 00000CE5 0F8572FFFFFF      <1>         jnz     _KIO_EXIT             ; RETURN IF SO
288 00000CEB F6C7FC            <1>       test  BH, 0FCh            ; TEST FOR OUT-OF-RANGE DELAY
289 00000CEE 0F8569FFFFFF      <1>         jnz     _KIO_EXIT             ; RETURN IF SO
290 00000CF4 B0F3              <1>       mov   al, KB_TYPA_RD            ; COMMAND FOR TYPAMATIC RATE/DELAY
291 00000CF6 E8DA060000        <1>       call  SND_DATA            ; SEND TO KEYBOARD
292                            <1>       ;mov  cx, 5               ; SHIFT COUNT
293                            <1>       ;shl  bh, cl              ; SHIFT DELAY OVER
294 00000CFB C0E705            <1>       shl   bh, 5
295 00000CFE 88D8              <1>       mov   al, bl              ; PUT IN RATE
296 00000D00 08F8              <1>       or    al, bh              ; AND DELAY
297 00000D02 E8CE060000        <1>       call  SND_DATA            ; SEND TO KEYBOARD
```

```
298 00000D07 E951FFFFFF          <1>         jmp     _KIO_EXIT               ; RETURN TO CALLER
299                              <1>
300                              <1>       ;----- WRITE TO KEYBOARD BUFFER
301                              <1> _K500:
302 00000D0C 56                  <1>       push   esi                ; SAVE SI (esi)
303 00000D0D FA                  <1>       cli                       ;
304 00000D0E 8B1D[9E5E0000]      <1>       mov    ebx, [BUFFER_TAIL] ; GET THE 'IN TO' POINTER TO THE BUFFER
305 00000D14 89DE                <1>       mov    esi, ebx           ; SAVE A COPY IN CASE BUFFER NOT FULL
306 00000D16 E8D3000000          <1>       call   _K4                ; BUMP THE POINTER TO SEE IF BUFFER IS FULL
307 00000D1B 3B1D[9A5E0000]      <1>       cmp    ebx, [BUFFER_HEAD] ; WILL THE BUFFER OVERRUN IF WE STORE THIS?
308 00000D21 740D                <1>       je     short _K502        ; YES - INFORM CALLER OF ERROR
309 00000D23 66890E              <1>       mov    [esi], cx          ; NO - PUT ASCII/SCAN CODE INTO BUFFER
310 00000D26 891D[9E5E0000]      <1>       mov    [BUFFER_TAIL], ebx ; ADJUST 'IN TO' POINTER TO REFLECT CHANGE
311 00000D2C 28C0                <1>       sub    al, al             ; TELL CALLER THAT OPERATION WAS SUCCESSFUL
312 00000D2E EB02                <1>       jmp    short _K504        ; SUB INSTRUCTION ALSO RESETS CARRY FLAG
313                              <1> _K502:
314 00000D30 B001                <1>       mov    al, 01h                  ; BUFFER FULL INDICATION
315                              <1> _K504:
316 00000D32 FB                  <1>       sti
317 00000D33 5E                  <1>       pop    esi                ; RECOVER SI (esi)
318 00000D34 E924FFFFFF          <1>         jmp     _KIO_EXIT               ; RETURN TO CALLER WITH STATUS IN AL
319                              <1>
320                              <1>       ;----- READ THE KEY TO FIGURE OUT WHAT TO DO -----
321                              <1> _K1S:
322 00000D39 FA                  <1>       cli    ; 03/12/2014
323 00000D3A 8B1D[9A5E0000]      <1>         mov     ebx, [BUFFER_HEAD]     ; GET POINTER TO HEAD OF BUFFER
324 00000D40 3B1D[9E5E0000]      <1>         cmp     ebx, [BUFFER_TAIL]     ; TEST END OF BUFFER
325                              <1>       ;jne   short _K1U          ; IF ANYTHING IN BUFFER SKIP INTERRUPT
326 00000D46 750F                <1>       jne    short _k1x ; 03/12/2014
327                              <1>       ;
328                              <1>       ; 03/12/2014
329                              <1>       ; 28/08/2014
330                              <1>       ; PERFORM OTHER FUNCTION ?? here !
331                              <1>       ;; MOV AX, 9002h           ; MOVE IN WAIT CODE & TYPE
332                              <1>       ;; INT       15H                ; PERFORM OTHER FUNCTION
333                              <1> _K1T:                                 ; ASCII READ
334 00000D48 FB                  <1>       sti                       ; INTERRUPTS BACK ON DURING LOOP
335 00000D49 90                  <1>       nop                       ; ALLOW AN INTERRUPT TO OCCUR
336                              <1> _K1U:
337 00000D4A FA                  <1>       cli                       ; INTERRUPTS BACK OFF
338 00000D4B 8B1D[9A5E0000]      <1>         mov        ebx, [BUFFER_HEAD]  ; GET POINTER TO HEAD OF BUFFER
339 00000D51 3B1D[9E5E0000]      <1>         cmp     ebx, [BUFFER_TAIL]     ; TEST END OF BUFFER
340                              <1> _k1x:
341 00000D57 53                  <1>       push   ebx                ; SAVE ADDRESS
342 00000D58 9C                  <1>       pushf                     ; SAVE FLAGS
343 00000D59 E82F070000          <1>       call   MAKE_LED           ; GO GET MODE INDICATOR DATA BYTE
344 00000D5E 8A1D[8F5E0000]      <1>       mov    bl, [KB_FLAG_2]    ; GET PREVIOUS BITS
345 00000D64 30C3                <1>       xor    bl, al             ; SEE IF ANY DIFFERENT
346 00000D66 80E307              <1>       and    bl, 07h      ; KB_LEDS    ; ISOLATE INDICATOR BITS
347 00000D69 7406                <1>       jz     short _K1V         ; IF NO CHANGE BYPASS UPDATE
348 00000D6B E8C9060000          <1>       call   SND_LED1
349 00000D70 FA                  <1>       cli                       ; DISABLE INTERRUPTS
350                              <1> _K1V:
351 00000D71 9D                  <1>       popf                      ; RESTORE FLAGS
352 00000D72 5B                  <1>       pop    ebx                ; RESTORE ADDRESS
353 00000D73 74D3                <1>         je      short _K1T              ; LOOP UNTIL SOMETHING IN BUFFER
354                              <1>       ;
355 00000D75 668B03              <1>       mov    ax, [ebx]          ; GET SCAN CODE AND ASCII CODE
356 00000D78 E871000000          <1>         call    _K4                     ; MOVE POINTER TO NEXT POSITION
357 00000D7D 891D[9A5E0000]      <1>         mov     [BUFFER_HEAD], ebx      ; STORE VALUE IN VARIABLE
358 00000D83 C3                  <1>       retn                      ; RETURN
359                              <1>
360                              <1>       ;----- READ THE KEY TO SEE IF ONE IS PRESENT -----
361                              <1> _K2S:
362 00000D84 FA                  <1>       cli                       ; INTERRUPTS OFF
363 00000D85 8B1D[9A5E0000]      <1>         mov     ebx, [BUFFER_HEAD]      ; GET HEAD POINTER
364 00000D8B 3B1D[9E5E0000]      <1>         cmp     ebx, [BUFFER_TAIL]      ; IF EQUAL (Z=1) THEN NOTHING THERE
365 00000D91 668B03              <1>       mov    ax, [ebx]
366 00000D94 9C                  <1>       pushf                     ; SAVE FLAGS
367 00000D95 6650                <1>       push   ax                 ; SAVE CODE
368 00000D97 E8F1060000          <1>       call   MAKE_LED           ; GO GET MODE INDICATOR DATA BYTE
369 00000D9C 8A1D[8F5E0000]      <1>       mov    bl, [KB_FLAG_2]    ; GET PREVIOUS BITS
370 00000DA2 30C3                <1>       xor    bl, al             ; SEE IF ANY DIFFERENT
371 00000DA4 80E307              <1>       and    bl, 07h ; KB_LEDS  ; ISOLATE INDICATOR BITS
372 00000DA7 7405                <1>       jz     short _K2T         ; IF NO CHANGE BYPASS UPDATE
373 00000DA9 E874060000          <1>       call   SND_LED                   ; GO TURN ON MODE INDICATORS
374                              <1> _K2T:
375 00000DAE 6658                <1>       pop    ax                 ; RESTORE CODE
376 00000DB0 9D                  <1>       popf                      ; RESTORE FLAGS
377 00000DB1 FB                  <1>       sti                       ; INTERRUPTS BACK ON
378 00000DB2 C3                  <1>       retn                      ; RETURN
379                              <1>
380                              <1>       ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR EXTENDED CALLS -----
381                              <1> _KIO_E_XLAT:
382 00000DB3 3CF0                <1>       cmp    al, 0F0h           ; IS IT ONE OF THE FILL-INs?
383 00000DB5 7506                <1>       jne    short _KIO_E_RET   ; NO, PASS IT ON
384 00000DB7 08E4                <1>         or    ah, ah            ; AH = 0 IS SPECIAL CASE
385 00000DB9 7402                <1>         jz    short _KIO_E_RET      ; PASS THIS ON UNCHANGED
386 00000DBB 30C0                <1>       xor    al, al             ; OTHERWISE SET AL = 0
387                              <1> _KIO_E_RET:
388 00000DBD C3                  <1>       retn                      ; GO BACK
389                              <1>
390                              <1>       ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR STANDARD CALLS -----
391                              <1> _KIO_S_XLAT:
392 00000DBE 80FCE0              <1>       cmp    ah, 0E0h           ; IS IT KEYPAD ENTER OR / ?
393 00000DC1 750F                <1>       jne    short _KIO_S2      ; NO, CONTINUE
394 00000DC3 3C0D                <1>       cmp    al, 0Dh                  ; KEYPAD ENTER CODE?
395 00000DC5 7408                <1>         je    short _KIO_S1      ; YES, MASSAGE A BIT
396 00000DC7 3C0A                <1>       cmp    al, 0Ah                  ; CTRL KEYPAD ENTER CODE?
397 00000DC9 7404                <1>         je    short _KIO_S1      ; YES, MASSAGE THE SAME
398 00000DCB B435                <1>       mov    ah, 35h                  ; NO, MUST BE KEYPAD /
399                              <1> _kio_ret: ; 03/12/2014
```

```
00000DCD F8                <1>       clc
00000DCE C3                <1>       retn
                           <1>       ;jmp   short _KIO_USE            ; GIVE TO CALLER
                           <1> _KIO_S1:
00000DCF B41C              <1>       mov    ah, 1Ch                   ; CONVERT TO COMPATIBLE OUTPUT
                           <1>       ;jmp   short _KIO_USE            ; GIVE TO CALLER
00000DD1 C3                <1>       retn
                           <1> _KIO_S2:
00000DD2 80FC84            <1>       cmp    ah, 84h                   ; IS IT ONE OF EXTENDED ONES?
00000DD5 7715              <1>       ja     short _KIO_DIS            ; YES, THROW AWAY AND GET ANOTHER CHAR
00000DD7 3CF0              <1>       cmp    al, 0F0h           ; IS IT ONE OF THE FILL-INs?
00000DD9 7506              <1>         jne  short _KIO_S3       ; NO, TRY LAST TEST
00000DDB 08E4              <1>       or     ah, ah             ; AH = 0 IS SPECIAL CASE
00000DDD 740C              <1>         jz   short _KIO_USE            ; PASS THIS ON UNCHANGED
00000DDF EB0B              <1>       jmp    short _KIO_DIS            ; THROW AWAY THE REST
                           <1> _KIO_S3:
00000DE1 3CE0              <1>       cmp    al, 0E0h           ; IS IT AN EXTENSION OF A PREVIOUS ONE?
                           <1>       ;jne   short _KIO_USE            ; NO, MUST BE A STANDARD CODE
00000DE3 75E8              <1>       jne    short _kio_ret
00000DE5 08E4              <1>       or     ah, ah             ; AH = 0 IS SPECIAL CASE
00000DE7 7402              <1>         jz   short _KIO_USE            ; JUMP IF AH = 0
00000DE9 30C0              <1>       xor    al, al             ; CONVERT TO COMPATIBLE OUTPUT
                           <1>       ;jmp   short _KIO_USE            ; PASS IT ON TO CALLER
                           <1> _KIO_USE:
                           <1>       ;clc                      ; CLEAR CARRY TO INDICATE GOOD CODE
00000DEB C3                <1>       retn                      ; RETURN
                           <1> _KIO_DIS:
00000DEC F9                <1>       stc                       ; SET CARRY TO INDICATE DISCARD CODE
00000DED C3                <1>       retn                      ; RETURN
                           <1>
                           <1>       ;----- INCREMENT BUFFER POINTER ROUTINE -----
                           <1> _K4:
00000DEE 43                <1>       inc     ebx
00000DEF 43                <1>       inc    ebx                ; MOVE TO NEXT WORD IN LIST
00000DF0 3B1D[965E0000]    <1>         cmp     ebx, [BUFFER_END]      ; AT END OF BUFFER?
                           <1>         ;jne    short _K5               ; NO, CONTINUE
00000DF6 7206              <1>       jb     short _K5
00000DF8 8B1D[925E0000]    <1>         mov     ebx, [BUFFER_START]     ; YES, RESET TO BUFFER BEGINNING
                           <1> _K5:
00000DFE C3                <1>       retn
                           <1>
                           <1> ; 20/02/2015
                           <1> ; 05/12/2014
                           <1> ; 26/08/2014
                           <1> ; KEYBOARD (HARDWARE) INTERRUPT -  IRQ LEVEL 1
                           <1> ; (INT_09h - Retro UNIX 8086 v1 - U9.ASM, 07/03/2014)
                           <1> ;
                           <1> ; Derived from "KB_INT_1" procedure of IBM "pc-at"
                           <1> ; rombios source code (06/10/1985)
                           <1> ; 'keybd.asm', HARDWARE INT 09h - (IRQ Level 1)
                           <1>
                           <1> ; EQUATES (IBM PC-XT-286 BIOS, 1986, 'POSQEQU.INC')
                           <1>
                           <1> ;--------- 8042 COMMANDS ----------------------------------------------------
                           <1> ENA_KBD            equ    0AEh         ; ENABLE KEYBOARD COMMAND
                           <1> DIS_KBD            equ    0ADh         ; DISABLE KEYBOARD COMMAND
                           <1> SHUT_CMD    equ    0FEh         ; CAUSE A SHUTDOWN COMMAND
                           <1> ;--------- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS ------------
                           <1> STATUS_PORT  equ    064h         ; 8042 STATUS PORT
                           <1> INPT_BUF_FULL      equ    00000010b    ; 1 = +INPUT BUFFER FULL
                           <1> PORT_A             equ    060h         ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
                           <1> ;---------- 8042 KEYBOARD RESPONSE ------------------------------------------
                           <1> KB_ACK             equ    0FAh         ; ACKNOWLEDGE PROM TRANSMISSION
                           <1> KB_RESEND    equ    0FEh         ; RESEND REQUEST
                           <1> KB_OVER_RUN  equ    0FFh         ; OVER RUN SCAN CODE
                           <1> ;---------- KEYBOARD/LED COMMANDS -------------------------------------------
                           <1> KB_ENABLE    equ    0F4h         ; KEYBOARD ENABLE
                           <1> LED_CMD            equ    0EDh         ; LED WRITE COMMAND
                           <1> KB_TYPA_RD   equ    0F3h         ; TYPAMATIC RATE/DELAY COMMAND
                           <1> ;---------- KEYBOARD SCAN CODES ---------------------------------------------
                           <1> NUM_KEY            equ    69           ; SCAN CODE FOR      NUMBER LOCK KEY
                           <1> SCROLL_KEY   equ    70           ; SCAN CODE FOR      SCROLL LOCK KEY
                           <1> ALT_KEY            equ    56           ; SCAN CODE FOR      ALTERNATE SHIFT KEY
                           <1> CTL_KEY            equ    29           ; SCAN CODE FOR      CONTROL KEY
                           <1> CAPS_KEY     equ    58           ; SCAN CODE FOR      SHIFT LOCK KEY
                           <1> DEL_KEY            equ    83           ; SCAN CODE FOR      DELETE KEY
                           <1> INS_KEY            equ    82           ; SCAN CODE FOR      INSERT KEY
                           <1> LEFT_KEY     equ    42           ; SCAN CODE FOR      LEFT SHIFT
                           <1> RIGHT_KEY    equ    54           ; SCAN CODE FOR      RIGHT SHIFT
                           <1> SYS_KEY            equ    84           ; SCAN CODE FOR      SYSTEM KEY
                           <1> ;---------- ENHANCED KEYBOARD SCAN CODES -------------------------------------
                           <1> ID_1         equ    0ABh         ; 1ST ID CHARACTER FOR KBX
                           <1> ID_2         equ    041h         ; 2ND ID CHARACTER FOR KBX
                           <1> ID_2A        equ    054h         ; ALTERNATE 2ND ID CHARACTER FOR KBX
                           <1> F11_M        equ    87           ; F11 KEY MAKE
                           <1> F12_M        equ    88           ; F12 KEY MAKE
                           <1> MC_E0        equ    224          ; GENERAL MARKER CODE
                           <1> MC_E1        equ    225          ; PAUSE KEY MARKER CODE
                           <1> ;---------- FLAG EQUATES WITHIN @KB_FLAG-------------------------------------
                           <1> RIGHT_SHIFT  equ    00000001b    ; RIGHT SHIFT KEY DEPRESSED
                           <1> LEFT_SHIFT   equ    00000010b    ; LEFT SHIFT KEY DEPRESSED
                           <1> CTL_SHIFT    equ    00000100b    ; CONTROL SHIFT KEY DEPRESSED
                           <1> ALT_SHIFT    equ    00001000b    ; ALTERNATE SHIFT KEY DEPRESSED
                           <1> SCROLL_STATE equ    00010000b    ; SCROLL LOCK STATE IS ACTIVE
                           <1> NUM_STATE    equ    00100000b    ; NUM LOCK STATE IS ACTIVE
                           <1> CAPS_STATE   equ    01000000b    ; CAPS LOCK STATE IS ACTIVE
                           <1> INS_STATE    equ    10000000b    ; INSERT STATE IS ACTIVE
                           <1> ;---------- FLAG EQUATES WITHIN @KB_FLAG_1 -----------------------------------
                           <1> L_CTL_SHIFT  equ    00000001b    ; LEFT CTL KEY DOWN
                           <1> L_ALT_SHIFT  equ    00000010b    ; LEFT ALT KEY DOWN
                           <1> SYS_SHIFT    equ    00000100b    ; SYSTEM KEY DEPRESSED AND HELD
                           <1> HOLD_STATE   equ    00001000b    ; SUSPEND KEY HAS BEEN TOGGLED
```

```
                        <1> SCROLL_SHIFT equ    00010000b    ; SCROLL LOCK KEY IS DEPRESSED
                        <1> NUM_SHIFT    equ    00100000b    ; NUM LOCK KEY IS DEPRESSED
                        <1> CAPS_SHIFT   equ    01000000b    ; CAPS LOCK KEY IS DEPRE55ED
                        <1> INS_SHIFT    equ    10000000b    ; INSERT KEY IS DEPRESSED
                        <1> ;---------- FLAGS EQUATES WITHIN @KB_FLAG_2 ----------------------------------
                        <1> KB_LEDS             equ    00000111b    ; KEYBOARD LED STATE BITS
                        <1> ;           equ    00000001b    ; SCROLL LOCK INDICATOR
                        <1> ;           equ    00000010b    ; NUM LOCK INDICATOR
                        <1> ;           equ    00000100b    ; CAPS LOCK INDICATOR
                        <1> ;           equ    00001000b    ; RESERVED (MUST BE ZERO)
                        <1> KB_FA       equ    00010000b    ; ACKNOWLEDGMENT RECEIVED
                        <1> KB_FE       equ    00100000b    ; RESEND RECEIVED FLAG
                        <1> KB_PR_LED   equ    01000000b    ; MODE INDICATOR UPDATE
                        <1> KB_ERR              equ    10000000b    ; KEYBOARD TRANSMIT ERROR FLAG
                        <1> ;----------- FLAGS EQUATES WITHIN @KB_FLAG_3 ----------------------------------
                        <1> LC_E1       equ    00000001b    ; LAST CODE WAS THE E1 HIDDEN CODE
                        <1> LC_E0       equ    00000010b    ; LAST CODE WAS THE E0 HIDDEN CODE
                        <1> R_CTL_SHIFT equ    00000100b    ; RIGHT CTL KEY DOWN
                        <1> R_ALT_SHIFT equ    00001000b    ; RIGHT ALT KEY DOWN
                        <1> GRAPH_ON    equ    00001000b    ; ALT GRAPHICS KEY DOWN (WT ONLY)
                        <1> KBX         equ    00010000b    ; ENHANCED KEYBOARD INSTALLED
                        <1> SET_NUM_LK  equ    00100000b    ; FORCE NUM LOCK IF READ ID AND KBX
                        <1> LC_AB       equ    01000000b    ; LAST CHARACTER WAS FIRST ID CHARACTER
                        <1> RD_ID       equ    10000000b    ; DOING A READ ID (MUST BE BIT0)
                        <1> ;
                        <1> ;----------- INTERRUPT EQUATES -----------------------------------------------
                        <1> EOI         equ    020h         ; END OF INTERRUPT COMMAND TO 8259
                        <1> INTA00              equ    020h         ; 8259 PORT
                        <1>
                        <1>
                        <1> kb_int:
                        <1>
                        <1> ; 17/10/2015 ('ctrlbrk')
                        <1> ; 05/12/2014
                        <1> ; 04/12/2014 (derived from pc-xt-286 bios source code -1986-)
                        <1> ; 26/08/2014
                        <1> ;
                        <1> ; 03/06/86  KEYBOARD BIOS
                        <1> ;
                        <1> ;--- HARDWARE INT 09H -- (IRQ LEVEL 1) ----------------------------------------
                        <1> ;                                                                            ;
                        <1> ;     KEYBOARD INTERRUPT ROUTINE                                             ;
                        <1> ;                                                                            ;
                        <1> ;--------------------------------------------------------------------------------
                        <1>
                        <1> KB_INT_1:
00000DFF FB             <1>       sti                       ; ENABLE INTERRUPTS
                        <1>       ;push  ebp
00000E00 50             <1>       push   eax
00000E01 53             <1>       push   ebx
00000E02 51             <1>       push   ecx
00000E03 52             <1>       push   edx
00000E04 56             <1>       push   esi
00000E05 57             <1>       push   edi
00000E06 1E             <1>       push   ds
00000E07 06             <1>       push   es
00000E08 FC             <1>       cld                       ; FORWARD DIRECTION
00000E09 66B81000       <1>       mov    ax, KDATA
00000E0D 8ED8           <1>       mov    ds, ax
00000E0F 8EC0           <1>       mov    es, ax
                        <1>       ;
                        <1>       ;-----WAIT FOR KEYBOARD DISABLE COMMAND TO BE ACCEPTED
00000E11 B0AD           <1>       mov    al, DIS_KBD        ; DISABLE THE KEYBOARD COMMAND
00000E13 E8A9050000     <1>       call   SHIP_IT                 ; EXECUTE DISABLE
00000E18 FA             <1>       cli                       ; DISABLE INTERRUPTS
00000E19 B900000100     <1>       mov    ecx, 10000h        ; SET MAXIMUM TIMEOUT
                        <1> KB_INT_01:
00000E1E E464           <1>       in     al, STATUS_PORT           ; READ ADAPTER STATUS
00000E20 A802           <1>       test   al, INPT_BUF_FULL  ; CHECK INPUT BUFFER FULL STATUS BIT
00000E22 E0FA           <1>       loopnz KB_INT_01          ; WAIT FOR COMMAND TO BE ACCEPTED
                        <1>       ;
                        <1>       ;-----READ CHARACTER FROM KEYBOARD INTERFACE
00000E24 E460           <1>       in     al, PORT_A         ; READ IN THE CHARACTER
                        <1>       ;
                        <1>       ;-----SYSTEM HOOK INT 15H - FUNCTION 4FH (ON HARDWARE INT LEVEL 9H)
                        <1>       ;MOV   AH, 04FH           ; SYSTEM INTERCEPT - KEY CODE FUNCTION
                        <1>       ;STC                      ; SET CY=1 (IN CASE OF IRET)
                        <1>       ;INT   15H                ; CASETTE CALL (AL)=KEY SCAN CODE
                        <1>       ;                         ; RETURNS CY=1 FOR INVALID FUNCTION
                        <1>       ;JC    KB_INT_02          ; CONTINUE IF CARRY FLAG SET ((AL)=CODE)
                        <1>       ;JMP   K26                ; EXIT IF SYSTEM HANDLES SCAN CODE
                        <1>       ;                         ; EXIT HANDLES HARDWARE EOI AND ENABLE
                        <1>       ;
                        <1>       ;-----CHECK FOR A RESEND COMMAND TO KEYBOARD
                        <1> KB_INT_02:                      ;        (AL)= SCAN CODE
00000E26 FB             <1>       sti                       ; ENABLE INTERRUPTS AGAIN
00000E27 3CFE           <1>       cmp    al, KB_RESEND      ; IS THE INPUT A RESEND
00000E29 7411           <1>         je     short KB_INT_4          ; GO IF RESEND
                        <1>       ;
                        <1>       ;-----CHECK FOR RESPONSE TO A COMMAND TO KEYBOARD
00000E2B 3CFA           <1>       cmp    al, KB_ACK         ; IS THE INPUT AN ACKNOWLEDGE
00000E2D 751A           <1>         jne    short KB_INT_2          ; GO IF NOT
                        <1>       ;
                        <1>       ;-----A COMMAND TO THE KEYBOARD WAS ISSUED
00000E2F FA             <1>       cli                       ; DISABLE INTERRUPTS
00000E30 800D[8F5E0000]10 <1>     or     byte [KB_FLAG_2], KB_FA ; INDICATE ACK RECEIVED
00000E37 E97A020000     <1>         jmp    K26                     ; RETURN IF NOT (ACK RETURNED FOR DATA)
                        <1>       ;
                        <1>       ;-----RESEND THE LAST BYTE
                        <1> KB_INT_4:
00000E3C FA             <1>       cli                       ; DISABLE INTERRUPTS
00000E3D 800D[8F5E0000]20 <1>     or     byte [KB_FLAG_2], KB_FE ; INDICATE RESEND RECEIVED
```

```
604 00000E44 E96D020000          <1>         jmp     K26                     ; RETURN IF NOT ACK RETURNED FOR DATA)
605                              <1>       ;
606                              <1> ;-----      UPDATE MODE INDICATORS IF CHANGE IN STATE
607                              <1> KB_INT_2:
608 00000E49 6650                <1>       push   ax                  ; SAVE DATA IN
609 00000E4B E83D060000          <1>       call   MAKE_LED            ; GO GET MODE INDICATOR DATA BYTE
610 00000E50 8A1D[8F5E0000]      <1>       mov    bl, [KB_FLAG_2]     ; GET PREVIOUS BITS
611 00000E56 30C3                <1>       xor    bl, al              ; SEE IF ANY DIFFERENT
612 00000E58 80E307              <1>       and    bl, KB_LEDS         ; ISOLATE INDICATOR BITS
613 00000E5B 7405                <1>       jz     short UP0           ; IF NO CHANGE BYPASS UPDATE
614 00000E5D E8C0050000          <1>       call   SND_LED                   ; GO TURN ON MODE INDICATORS
615                              <1> UP0:
616 00000E62 6658                <1>       pop    ax                  ; RESTORE DATA IN
617                              <1> ;--------------------------------------------------------------------------
618                              <1> ;     START OF KEY PROCESSING                                        ;
619                              <1> ;--------------------------------------------------------------------------
620 00000E64 88C4                <1>       mov    ah, al              ; SAVE SCAN CODE IN AH ALSO
621                              <1>       ;
622                              <1>       ;-----TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
623 00000E66 3CFF                <1>       cmp    al, KB_OVER_RUN             ; IS THIS AN OVERRUN CHAR
624 00000E68 0F843F050000        <1>         je     K62                       ; BUFFER_FULL_BEEP
625                              <1>       ;
626                              <1> K16:
627 00000E6E 8A3D[905E0000]      <1>       mov    bh, [KB_FLAG_3]             ; LOAD FLAGS FOR TESTING
628                              <1>       ;
629                              <1>       ;-----TEST TO SEE IF A READ_ID IS IN PROGRESS
630 00000E74 F6C7C0              <1>       test   bh, RD_ID+LC_AB     ; ARE WE DOING A READ ID?
631 00000E77 7449                <1>       jz     short NOT_ID        ; CONTINUE IF NOT
632 00000E79 7917                <1>       jns    short TST_ID_2           ; IS THE RD_ID FLAG ON?
633 00000E7B 3CAB                <1>       cmp    al, ID_1            ; IS THIS THE 1ST ID CHARACTER?
634 00000E7D 7507                <1>       jne    short RST_RD_ID
635 00000E7F 800D[905E0000]40    <1>       or     byte [KB_FLAG_3], LC_AB ; INDICATE 1ST ID WAS OK
636                              <1> RST_RD_ID:
637 00000E86 8025[905E0000]7F    <1>       and    byte [KB_FLAG_3], ~RD_ID ; RESET THE READ ID FLAG
638                              <1>         ;jmp    short ID_EX           ; AND EXIT
639 00000E8D E924020000          <1>       jmp    K26
640                              <1>       ;
641                              <1> TST_ID_2:
642 00000E92 8025[905E0000]BF    <1>       and    byte [KB_FLAG_3], ~LC_AB ; RESET FLAG
643 00000E99 3C54                <1>       cmp    al, ID_2A           ; IS THIS THE 2ND ID CHARACTER?
644 00000E9B 7419                <1>         je   short KX_BIT        ; JUMP IF SO
645 00000E9D 3C41                <1>       cmp    al, ID_2            ; IS THIS THE 2ND ID CHARACTER?
646                              <1>         ;jne short ID_EX         ; LEAVE IF NOT
647 00000E9F 0F8511020000        <1>       jne    K26
648                              <1>       ;
649                              <1>       ;-----A READ ID SAID THAT IT WAS ENHANCED KEYBOARD
650 00000EA5 F6C720              <1>       test   bh, SET_NUM_LK              ; SHOULD WE SET NUM LOCK?
651 00000EA8 740C                <1>         jz     short KX_BIT            ; EXIT IF NOT
652 00000EAA 800D[8D5E0000]20    <1>       or     byte [KB_FLAG], NUM_STATE ; FORCE NUM LOCK ON
653 00000EB1 E86C050000          <1>       call   SND_LED                   ; GO SET THE NUM LOCK INDICATOR
654                              <1> KX_BIT:
655 00000EB6 800D[905E0000]10    <1>       or     byte [KB_FLAG_3], KBX     ; INDICATE ENHANCED KEYBOARD WAS FOUND
656 00000EBD E9F4010000          <1> ID_EX:      jmp     K26               ; EXIT
657                              <1>       ;
658                              <1> NOT_ID:
659 00000EC2 3CE0                <1>       cmp    al, MC_E0           ; IS THIS THE GENERAL MARKER CODE?
660 00000EC4 750C                <1>       jne    short TEST_E1
661 00000EC6 800D[905E0000]12    <1>       or     byte [KB_FLAG_3], LC_E0+KBX ; SET FLAG BIT, SET KBX, AND
662                              <1>       ;jmp   short EXIT          ; THROW AWAY THIS CODE
663 00000ECD E9EB010000          <1>       jmp    K26A
664                              <1> TEST_E1:
665 00000ED2 3CE1                <1>       cmp    al, MC_E1           ; IS THIS THE PAUSE KEY?
666 00000ED4 750C                <1>       jne    short NOT_HC
667 00000ED6 800D[905E0000]11    <1>       or     byte [KB_FLAG_3], LC_E1+KBX ; SET FLAG BIT, SET KBX, AND
668 00000EDD E9DB010000          <1> EXIT: jmp    K26A                ; THROW AWAY THIS CODE
669                              <1>       ;
670                              <1> NOT_HC:
671 00000EE2 247F                <1>       and    al, 07Fh            ; TURN OFF THE BREAK BIT
672 00000EE4 F6C702              <1>       test   bh, LC_E0           ; LAST CODE THE E0 MARKER CODE
673 00000EE7 7414                <1>       jz     short NOT_LC_E0           ; JUMP IF NOT
674                              <1>       ;
675 00000EE9 BF[7A5D0000]        <1>       mov    edi, _K6+6          ; IS THIS A SHIFT KEY?
676 00000EEE AE                  <1>       scasb
677 00000EEF 0F84C1010000        <1>         je      K26 ; K16B              ; YES, THROW AWAY & RESET FLAG
678 00000EF5 AE                  <1>       scasb
679 00000EF6 757C                <1>       jne    short K16A          ; NO, CONTINUE KEY PROCESSING
680                              <1>       ;jmp   short K16B          ; YES, THROW AWAY & RESET FLAG
681 00000EF8 E9B9010000          <1>       jmp    K26
682                              <1>       ;
683                              <1> NOT_LC_E0:
684 00000EFD F6C701              <1>       test   bh, LC_E1           ; LAST CODE THE E1 MARKER CODE?
685 00000F00 7435                <1>       jz     short T_SYS_KEY           ; JUMP IF NOT
686 00000F02 B904000000          <1>       mov    ecx, 4              ; LENGHT OF SEARCH
687 00000F07 BF[785D0000]        <1>       mov    edi, _K6+4          ; IS THIS AN ALT, CTL, OR SHIFT?
688 00000F0C F2AE                <1>       repne  scasb               ; CHECK IT
689                              <1>       ;je    short EXIT          ; THROW AWAY IF SO
690 00000F0E 0F84A9010000        <1>       je     K26A
691                              <1>       ;
692 00000F14 3C45                <1>       cmp    al, NUM_KEY         ; IS IT THE PAUSE KEY?
693                              <1>       ;jne   short K16B          ; NO, THROW AWAY & RESET FLAG
694 00000F16 0F859A010000        <1>       jne    K26
695 00000F1C F6C480              <1>       test   ah, 80h                   ; YES, IS IT THE BREAK OF THE KEY?
696                              <1>       ;jnz   short K16B          ; YES, THROW THIS AWAY, TOO
697 00000F1F 0F8591010000        <1>       jnz    K26
698                              <1>         ; 20/02/2015
699 00000F25 F605[8E5E0000]08    <1>       test   byte [KB_FLAG_1],HOLD_STATE ;  NO, ARE WE PAUSED ALREADY?
700                              <1>       ;jnz   short K16B          ;  YES, THROW AWAY
701 00000F2C 0F8584010000        <1>       jnz    K26
702 00000F32 E9E1020000          <1>       jmp     K39P                    ; NO, THIS IS THE REAL PAUSE STATE
703                              <1>       ;
704                              <1>       ;-----TEST FOR SYSTEM KEY
705                              <1> T_SYS_KEY:
```

```
706 00000F37 3C54                <1>       cmp    al, SYS_KEY         ; IS IT THE SYSTEM KEY?
707 00000F39 7539                <1>       jnz    short K16A          ; CONTINUE IF NOT
708                              <1>       ;
709 00000F3B F6C480              <1>       test   ah, 80h                     ; CHECK IF THIS A BREAK CODE
710 00000F3E 7524                <1>       jnz    short K16C          ; DO NOT TOUCH SYSTEM INDICATOR IF TRUE
711                              <1>       ;
712 00000F40 F605[8E5E0000]04    <1>       test   byte [KB_FLAG_1], SYS_SHIFT ; SEE IF IN SYSTEM KEY HELD DOWN
713                              <1>         ;jnz short K16B          ; IF YES, DO NOT PROCESS SYSTEM INDICATOR
714 00000F47 0F8569010000        <1>       jnz     K26
715                              <1>       ;
716 00000F4D 800D[8E5E0000]04    <1>       or     byte [KB_FLAG_1], SYS_SHIFT ; INDICATE SYSTEM KEY DEPRESSED
717 00000F54 B020                <1>       mov    al, EOI                     ; END OF INTERRUPT COMMAND
718 00000F56 E620                <1>       out    20h, al ;out INTA00, al    ; SEND COMMAND TO INTERRUPT CONTROL PORT
719                              <1>                                        ; INTERRUPT-RETURN-NO-EOI
720 00000F58 B0AE                <1>       mov    al, ENA_KBD         ; INSURE KEYBOARD IS ENABLED
721 00000F5A E862040000          <1>       call   SHIP_IT                     ; EXECUTE ENABLE
722                              <1>       ; !!! SYSREQ !!! function/system call (INTERRUPT) must be here !!!
723                              <1>       ;MOV   AL, 8500H           ; FUNCTION VALUE FOR MAKE OF SYSTEM KEY
724                              <1>       ;STI                       ; MAKE SURE INTERRUPTS ENABLED
725                              <1>       ;INT   15H                 ; USER INTERRUPT
726 00000F5F E965010000          <1>         jmp     K27A                    ; END PROCESSING
727                              <1>       ;
728                              <1> ;K16B:       jmp    K26                ; IGNORE SYSTEM KEY
729                              <1>       ;
730                              <1> K16C:
731 00000F64 8025[8E5E0000]FB    <1>       and    byte [KB_FLAG_1], ~SYS_SHIFT ; TURN OFF SHIFT KEY HELD DOWN
732 00000F6B B020                <1>       mov    al, EOI                     ; END OF INTERRUPT COMMAND
733 00000F6D E620                <1>       out    20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
734                              <1>                                        ; INTERRUPT-RETURN-NO-EOI
735                              <1>       ;MOV   AL, ENA_KBD         ; INSURE KEYBOARD IS ENABLED
736                              <1>       ;CALL  SHIP_IT                     ; EXECUTE ENABLE
737                              <1>       ;
738                              <1>       ;MOV   AX, 8501H           ; FUNCTION VALUE FOR BREAK OF SYSTEM KEY
739                              <1>       ;STI                       ; MAKE SURE INTERRUPTS ENABLED
740                              <1>       ;INT   15H                 ; USER INTERRUPT
741                              <1>       ;JMP   K27A                ; INGONRE SYSTEM KEY
742                              <1>       ;
743 00000F6F E94E010000          <1>       jmp     K27                ; IGNORE SYSTEM KEY
744                              <1>       ;
745                              <1>       ;-----TEST FOR SHIFT KEYS
746                              <1> K16A:
747 00000F74 8A1D[8D5E0000]      <1>       mov    bl, [KB_FLAG]       ; PUT STATE FLAGS IN BL
748 00000F7A BF[745D0000]        <1>       mov    edi, _K6            ; SHIFT KEY TABLE offset
749 00000F7F B908000000          <1>       mov    ecx, _K6L           ; LENGTH
750 00000F84 F2AE                <1>       repne  scasb               ; LOOK THROUGH THE TABLE FOR A MATCH
751 00000F86 88E0                <1>       mov    al, ah              ; RECOVER SCAN CODE
752 00000F88 0F8510010000        <1>         jne     K25                     ; IF NO MATCH, THEN SHIFT NOT FOUND
753                              <1>       ;
754                              <1>       ;------      SHIFT KEY FOUND
755                              <1> K17:
756 00000F8E 81EF[755D0000]      <1>         sub     edi, _K6+1              ; ADJUST PTR TO SCAN CODE MATCH
757 00000F94 8AA7[7C5D0000]      <1>              mov     ah, [edi+_K7]           ; GET MASK INTO AH
758 00000F9A B102                <1>       mov    cl, 2               ; SETUP COUNT FOR FLAG SHIFTS
759 00000F9C A880                <1>       test   al, 80h                    ; TEST FOR BREAK KEY
760 00000F9E 0F8596000000        <1>         jnz     K23                     ; JUMP OF BREAK
761                              <1>       ;
762                              <1>       ;-----SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
763                              <1> K17C:
764 00000FA4 80FC10              <1>       cmp    ah, SCROLL_SHIFT
765 00000FA7 732B                <1>       jae    short K18           ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
766                              <1>       ;
767                              <1>       ;-----PLAIN SHIFT KEY, SET SHIFT ON
768 00000FA9 0825[8D5E0000]      <1>       or     [KB_FLAG], ah       ; TURN ON SHIFT BIT
769 00000FAF A80C                <1>         testal, CTL_SHIFT+ALT_SHIFT ; IS IT ALT OR CTRL?
770                              <1>       ;jnz   short K17D          ; YES, MORE FLAGS TO SET
771 00000FB1 0F84FF000000        <1>       jz     K26                 ; NO, INTERRUPT RETURN
772                              <1> K17D:
773 00000FB7 F6C702              <1>       test   bh, LC_E0           ; IS THIS ONE OF NEW KEYS?
774 00000FBA 740B                <1>       jz     short K17E          ; NO, JUMP
775 00000FBC 0825[905E0000]      <1>       or     [KB_FLAG_3], ah           ; SET BITS FOR RIGHT CTRL, ALT
776 00000FC2 E9EF000000          <1>       jmp    K26                 ; INTERRUPT RETURN
777                              <1> K17E:
778 00000FC7 D2EC                <1>       shr    ah, cl              ; MOVE FLAG BITS TWO POSITIONS
779 00000FC9 0825[8E5E0000]      <1>       or     [KB_FLAG_1], ah           ; SET BITS FOR LEFT CTRL, ALT
780 00000FCF E9E2000000          <1>       jmp    K26
781                              <1>       ;
782                              <1>       ;-----TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
783                              <1> K18:                            ; SHIFT-TOGGLE
784 00000FD4 F6C304              <1>       test   bl, CTL_SHIFT              ; CHECK CTL SHIFT STATE
785                              <1>         ;jz        short K18A              ; JUMP IF NOT CTL STATE
786 00000FD7 0F85C1000000        <1>         jnz     K25                     ; JUMP IF CTL STATE
787                              <1> K18A:
788 00000FDD 3C52                <1>       cmp    al, INS_KEY         ; CHECK FOR INSERT KEY
789 00000FDF 7524                <1>       jne    short K22           ; JUMP IF NOT INSERT KEY
790 00000FE1 F6C308              <1>       test   bl, ALT_SHIFT              ; CHECK FOR ALTERNATE SHIFT
791                              <1>              ;jz    short K18B          ; JUMP IF NOT ALTERNATE SHIFT
792 00000FE4 0F85B4000000        <1>         jnz     K25                     ; JUMP IF ALTERNATE SHIFT
793                              <1> K18B:
794 00000FEA F6C702              <1>       test   bh, LC_E0 ;20/02/2015      ; IS THIS NEW INSERT KEY?
795 00000FED 7516                <1>       jnz    short K22           ; YES, THIS ONE'S NEVER A '0'
796                              <1> K19:
797 00000FEF F6C320              <1>       test   bl, NUM_STATE              ; CHECK FOR BASE STATE
798 00000FF2 750C                <1>       jnz    short K21           ; JUMP IF NUM LOCK IS ON
799 00000FF4 F6C303              <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
800 00000FF7 740C                <1>       jz     short K22           ; JUMP IF BASE STATE
801                              <1> K20:                            ; NUMERIC ZERO, NOT INSERT KEY
802 00000FF9 88C4                <1>       mov    ah, al              ; PUT SCAN CODE BACK IN AH
803 00000FFB E99E000000          <1>         jmp     K25                     ; NUMERAL '0', STNDRD. PROCESSING
804                              <1> K21:                            ; MIGHT BE NUMERIC
805 00001000 F6C303              <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT
806 00001003 74F4                <1>       jz     short K20           ; IS NUMERIC, STD. PROC.
807                              <1>       ;
```

```
                                <1> K22:                            ; SHIFT TOGGLE KEY HIT; PROCESS IT
00001005 8425[8E5E0000]         <1>       test   ah, [KB_FLAG_1]     ; IS KEY ALREADY DEPRESSED
0000100B 0F85A5000000           <1>         jnz     K26                     ; JUMP IF KEY ALREADY DEPRESSED
                                <1> K22A:
00001011 0825[8E5E0000]         <1>         or      [KB_FLAG_1], ah ; INDICATE THAT THE KEY IS DEPRESSED
00001017 3025[8D5E0000]         <1>       xor    [KB_FLAG], ah       ; TOGGLE THE SHIFT STATE
                                <1>       ;
                                <1>       ;----- TOGGLE LED IF CAPS, NUM  OR SCROLL KEY DEPRESSED
0000101D F6C470                 <1>       test   ah, CAPS_SHIFT+NUM_SHIFT+SCROLL_SHIFT ; SHIFT TOGGLE?
00001020 7409                   <1>       jz     short K22B          ; GO IF NOT
                                <1>       ;
00001022 6650                   <1>       push   ax                  ; SAVE SCAN CODE AND SHIFT MASK
00001024 E8F9030000             <1>       call   SND_LED                   ; GO TURN MODE INDICATORS ON
00001029 6658                   <1>       pop    ax                  ; RESTORE SCAN CODE
                                <1> K22B:
0000102B 3C52                   <1>       cmp    al, INS_KEY         ; TEST FOR 1ST MAKE OF INSERT KEY
0000102D 0F8583000000           <1>         jne     K26                     ; JUMP IF NOT INSERT KEY
00001033 88C4                   <1>       mov    ah, al               ; SCAN CODE IN BOTH HALVES OF AX
00001035 E999000000             <1>         jmp     K28                     ; FLAGS UPDATED, PROC. FOR BUFFER
                                <1>       ;
                                <1>       ;----- BREAK SHIFT FOUND
                                <1> K23:                            ; BREAK-SHIFT-FOUND
0000103A 80FC10                 <1>       cmp    ah, SCROLL_SHIFT    ; IS THIS A TOGGLE KEY
0000103D F6D4                   <1>       not    ah                  ; INVERT MASK
0000103F 7355                   <1>       jae    short K24           ; YES, HANDLE BREAK TOGGLE
00001041 2025[8D5E0000]         <1>       and    [KB_FLAG], ah       ; TURN OFF SHIFT BIT
00001047 80FCFB                 <1>       cmp    ah, ~CTL_SHIFT           ; IS THIS ALT OR CTL?
0000104A 7730                   <1>       ja     short K23D          ; NO, ALL DONE
                                <1>       ;
0000104C F6C702                 <1>       test   bh, LC_E0           ; 2ND ALT OR CTL?
0000104F 7408                   <1>       jz     short K23A          ; NO, HANSLE NORMALLY
00001051 2025[905E0000]         <1>       and    [KB_FLAG_3], ah           ; RESET BIT FOR RIGHT ALT OR CTL
00001057 EB08                   <1>       jmp    short K23B          ; CONTINUE
                                <1> K23A:
00001059 D2FC                   <1>       sar    ah, cl              ; MOVE THE MASK BIT TWO POSITIONS
0000105B 2025[8E5E0000]         <1>       and    [KB_FLAG_1], ah           ; RESET BIT FOR LEFT ALT AND CTL
                                <1> K23B:
00001061 88C4                   <1>       mov    ah, al              ; SAVE SCAN CODE
00001063 A0[905E0000]           <1>       mov    al, [KB_FLAG_3]           ; GET RIGHT ALT & CTRL FLAGS
00001068 D2E8                   <1>       shr    al, cl              ; MOVE TO BITS 1 & 0
0000106A 0A05[8E5E0000]         <1>       or     al, [KB_FLAG_1]           ; PUT IN LEFT ALŞT & CTL FLAGS
00001070 D2E0                   <1>       shl    al, cl              ; MOVE BACK TO BITS 3 & 2
00001072 240C                   <1>       and    al, ALT_SHIFT+CTL_SHIFT ; FILTER OUT OTHER GARBAGE
00001074 0805[8D5E0000]         <1>       or     [KB_FLAG], al       ; PUT RESULT IN THE REAL FLAGS
0000107A 88E0                   <1>       mov    al, ah
                                <1> K23D:
0000107C 3CB8                   <1>       cmp    al, ALT_KEY+80h           ; IS THIS ALTERNATE SHIFT RELEASE
0000107E 7536                   <1>       jne    short K26           ; INTERRUPT RETURN
                                <1>       ;
                                <1>       ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
00001080 A0[915E0000]           <1>       mov    al, [ALT_INPUT]
00001085 B400                   <1>       mov    ah, 0               ; SCAN CODE OF 0
00001087 8825[915E0000]         <1>       mov    [ALT_INPUT], ah     ; ZERO OUT THE FIELD
0000108D 3C00                   <1>       cmp    al, 0               ; WAS THE INPUT = 0?
0000108F 7425                   <1>       je     short K26           ; INTERRUPT_RETURN
                                <1>         ; 29/01/2016
                                <1>       ;jmp     K61                    ; IT WASN'T, SO PUT IN BUFFER
00001091 E9D0020000             <1>       jmp    _K60
                                <1>       ;
                                <1> K24:                            ; BREAK-TOGGLE
00001096 2025[8E5E0000]         <1>       and    [KB_FLAG_1], ah     ; INDICATE NO LONGER DEPRESSED
0000109C EB18                   <1>       jmp    short K26           ; INTERRUPT_RETURN
                                <1>       ;
                                <1>       ;----- TEST FOR HOLD STATE
                                <1>                                   ; AL, AH = SCAN CODE
                                <1> K25:                            ; NO-SHIFT-FOUND
0000109E 3C80                   <1>       cmp    al, 80h                  ; TEST FOR BREAK KEY
000010A0 7314                   <1>       jae    short K26           ; NOTHING FOR BREAK CHARS FROM HERE ON
000010A2 F605[8E5E0000]08       <1>       test   byte [KB_FLAG_1], HOLD_STATE ; ARE WE IN HOLD STATE
000010A9 7428                   <1>       jz     short K28           ; BRANCH AROUND TEST IF NOT
000010AB 3C45                   <1>       cmp    al, NUM_KEY
000010AD 7407                   <1>       je     short K26           ; CAN'T END HOLD ON NUM_LOCK
000010AF 8025[8E5E0000]F7       <1>       and    byte [KB_FLAG_1], ~HOLD_STATE ; TURN OFF THE HOLD STATE BIT
                                <1>       ;
                                <1> K26:
000010B6 8025[905E0000]FC       <1>       and    byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; RESET LAST CHAR H.C. FLAG
                                <1> K26A:                             ; INTERRUPT-RETURN
000010BD FA                     <1>       cli                         ; TURN OFF INTERRUPTS
000010BE B020                   <1>       mov    al, EOI                   ; END OF INTERRUPT COMMAND
000010C0 E620                   <1>       out    20h, al      ;out INTA00, al     ; SEND COMMAND TO INTERRUPT CONTROL PORT
                                <1> K27:                              ; INTERRUPT-RETURN-NO-EOI
000010C2 B0AE                   <1>       mov    al, ENA_KBD         ; INSURE KEYBOARD IS ENABLED
000010C4 E8F8020000             <1>       call   SHIP_IT                   ; EXECUTE ENABLE
                                <1> K27A:
000010C9 FA                     <1>       cli                         ; DISABLE INTERRUPTS
                                <1>       ;;mov  byte [intflg], 0 ; 07/01/2017 ;; 15/01/2017
000010CA 07                     <1>       pop    es                  ; RESTORE REGISTERS
000010CB 1F                     <1>       pop    ds
000010CC 5F                     <1>       pop    edi
000010CD 5E                     <1>       pop    esi
000010CE 5A                     <1>       pop    edx
000010CF 59                     <1>       pop    ecx
000010D0 5B                     <1>       pop    ebx
000010D1 58                     <1>       pop    eax
                                <1>       ;pop   ebp
000010D2 CF                     <1>       iretd                       ; RETURN
                                <1>
                                <1>       ;----- NOT IN HOLD STATE
                                <1> K28:                              ; NO-HOLD-STATE
000010D3 3C58                   <1>       cmp    al, 88              ; TEST FOR OUT-OF-RANGE SCAN CODES
000010D5 77DF                   <1>       ja     short K26           ; IGNORE IF OUT-OF-RANGE
                                <1>       ;
```

```
000010D7 F6C308              <1>       test   bl, ALT_SHIFT             ; ARE WE IN ALTERNATE SHIFT
                             <1>         ;jz  short K28A          ; IF NOT ALTERNATE
000010DA 0F84F1000000        <1>         jz      K38
                             <1>       ;
000010E0 F6C710              <1>       test   bh, KBX                   ; IS THIS THE ENCHANCED KEYBOARD?
000010E3 740D                <1>       jz     short K29          ; NO, ALT STATE IS REAL
                             <1>        ;28/02/2015
000010E5 F605[8E5E0000]04    <1>       test   byte [KB_FLAG_1], SYS_SHIFT ; YES, IS SYSREQ KEY DOWN?
                             <1>       ;jz    short K29          ;  NO, ALT STATE IS REAL
000010EC 0F85DF000000        <1>       jnz    K38                ; YES, THIS IS PHONY ALT STATE
                             <1>         ;                        ; DUE TO PRESSING SYSREQ
                             <1> ;K28A:       jmp    short K38
                             <1>       ;
                             <1>       ;----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
                             <1> K29:                             ; TEST-RESET
000010F2 F6C304              <1>       test   bl, CTL_SHIFT             ; ARE WE IN CONTROL SHIFT ALSO?
000010F5 740B                <1>       jz     short K31          ; NO_RESET
000010F7 3C53                <1>       cmp    al, DEL_KEY        ; CTL-ALT STATE, TEST FOR DELETE KEY
000010F9 7507                <1>       jne    short K31          ; NO_RESET, IGNORE
                             <1>       ;
                             <1>       ;----- CTL-ALT-DEL HAS BEEN FOUND
                             <1>       ; 26/08/2014
                             <1> cpu_reset:
                             <1>       ; IBM PC/AT ROM BIOS source code - 10/06/85 (TEST4.ASM - PROC_SHUTDOWN)
                             <1>       ; Send FEh (system reset command) to the keyboard controller.
000010FB B0FE                <1>       mov    al, SHUT_CMD       ; SHUTDOWN COMMAND
000010FD E664                <1>       out    STATUS_PORT, al           ; SEND TO KEYBOARD CONTROL PORT
                             <1> khere:
000010FF F4                  <1>       hlt                       ; WAIT FOR 80286 RESET
00001100 EBFD                <1>       jmp    short khere        ; INSURE HALT
                             <1>
                             <1>       ;
                             <1>       ;----- IN ALTERNATE SHIFT, RESET NOT FOUND
                             <1> K31:                             ; NO-RESET
00001102 3C39                <1>       cmp    al, 57             ; TEST FOR SPACE KEY
00001104 7507                <1>       jne    short K311         ; NOT THERE
00001106 B020                <1>       mov    al, ' '                  ; SET SPACE CHAR
00001108 E948020000          <1>         jmp     K57                   ; BUFFER_FILL
                             <1> K311:
0000110D 3C0F                <1>       cmp    al, 15             ; TEST FOR TAB KEY
0000110F 7509                <1>       jne    short K312         ; NOT THERE
00001111 66B800A5            <1>       mov    ax, 0A500h         ; SET SPECIAL CODE FOR ALT-TAB
00001115 E93B020000          <1>         jmp     K57                   ; BUFFER_FILL
                             <1> K312:
0000111A 3C4A                <1>       cmp    al, 74             ; TEST FOR KEY PAD -
0000111C 0F84A2000000        <1>         je      K37B                  ; GO PROCESS
00001122 3C4E                <1>       cmp    al, 78             ; TEST FOR KEY PAD +
00001124 0F849A000000        <1>         je      K37B                  ; GO PROCESS
                             <1>       ;
                             <1>       ;----- LOOK FOR KEY PAD ENTRY
                             <1> K32:                             ; ALT-KEY-PAD
0000112A BF[505D0000]        <1>       mov    edi, K30           ; ALT-INPUT-TABLE offset
0000112F B90A000000          <1>       mov    ecx, 10                  ; LOOK FOR ENTRY USING KEYPAD
00001134 F2AE                <1>       repne  scasb              ; LOOK FOR MATCH
00001136 7525                <1>       jne    short K33          ; NO_ALT_KEYPAD
00001138 F6C702              <1>       test   bh, LC_E0          ; IS THIS ONE OF THE NEW KEYS?
0000113B 0F858A000000        <1>         jnz     K37C                  ; YES, JUMP, NOT NUMPAD KEY
00001141 81EF[515D0000]      <1>       sub    edi, K30+1         ; DI NOW HAS ENTRY VALUE
00001147 A0[915E0000]        <1>       mov    al, [ALT_INPUT]    ; GET THE CURRENT BYTE
0000114C B40A                <1>       mov    ah, 10             ; MULTIPLY BY 10
0000114E F6E4                <1>       mul    ah
00001150 6601F8              <1>       add    ax, di             ; ADD IN THE LATEST ENTRY
00001153 A2[915E0000]        <1>       mov    [ALT_INPUT], al    ; STORE IT AWAY
                             <1> ;K32A:
00001158 E959FFFFFF          <1>         jmp     K26                   ; THROW AWAY THAT KEYSTROKE
                             <1>       ;
                             <1>       ;----- LOOK FOR SUPERSHIFT ENTRY
                             <1> K33:                             ; NO-ALT-KEYPAD
0000115D C605[915E0000]00    <1>         mov     byte [ALT_INPUT], 0     ; ZERO ANY PREVIOUS ENTRY INTO INPUT
00001164 B91A000000          <1>       mov    ecx, 26                 ; (DI),(ES) ALREADY POINTING
00001169 F2AE                <1>       repne  scasb              ; LOOK FOR MATCH IN ALPHABET
0000116B 7450                <1>       je     short K37A         ; MATCH FOUND, GO FILLL THE BUFFER
                             <1>       ;
                             <1>       ;----- LOOK FOR TOP ROW OF ALTERNATE SHIFT
                             <1> K34:                             ; ALT-TOP-ROW
0000116D 3C02                <1>       cmp    al, 2              ; KEY WITH '1' ON IT
0000116F 7253                <1>       jb     short K37B         ; MUST BE ESCAPE
00001171 3C0D                <1>       cmp    al, 13             ; IS IT IN THE REGION
00001173 7705                <1>       ja     short K35          ; NO, ALT SOMETHING ELSE
00001175 80C476              <1>       add    ah, 118                  ; CONVERT PSEUDO SCAN CODE TO RANGE
00001178 EB43                <1>       jmp    short K37A         ; GO FILL THE BUFFER
                             <1>       ;
                             <1>       ;----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
                             <1> K35:                             ; ALT-FUNCTION
0000117A 3C57                <1>       cmp    al, F11_M          ; IS IT F11?
0000117C 7209                <1>       jb     short K35A ; 20/02/2015   ; NO, BRANCH
0000117E 3C58                <1>       cmp    al, F12_M          ; IS IT F12?
00001180 7705                <1>       ja     short K35A ; 20/02/2015   ; NO, BRANCH
00001182 80C434              <1>       add    ah, 52             ; CONVERT TO PSEUDO SCAN CODE
00001185 EB36                <1>       jmp    short K37A         ; GO FILL THE BUFFER
                             <1> K35A:
00001187 F6C702              <1>       test   bh, LC_E0          ; DO WE HAVE ONE OF THE NEW KEYS?
0000118A 7422                <1>       jz     short K37          ; NO, JUMP
0000118C 3C1C                <1>       cmp    al, 28             ; TEST FOR KEYPAD ENTER
0000118E 7509                <1>         jne     short K35B             ; NOT THERE
00001190 66B800A6            <1>       mov    ax, 0A600h         ; SPECIAL CODE
00001194 E9BC010000          <1>       jmp    K57                ; BUFFER FILL
                             <1> K35B:
00001199 3C53                <1>       cmp    al, 83             ; TEST FOR DELETE KEY
0000119B 742E                <1>       je     short K37C         ; HANDLE WITH OTHER EDIT KEYS
0000119D 3C35                <1>       cmp    al, 53             ; TEST FOR KEYPAD /
                             <1>       ;jne   short K32A         ; NOT THERE, NO OTHER E0 SPECIALS
```

```
1012 0000119F 0F8511FFFFFF      <1>         jne     K26
1013 000011A5 66B800A4          <1>       mov   ax, 0A400h          ; SPECIAL CODE
1014 000011A9 E9A7010000        <1>       jmp   K57                 ; BUFFER FILL
1015                            <1> K37:
1016 000011AE 3C3B              <1>       cmp   al, 59              ; TEST FOR FUNCTION KEYS (F1)
1017 000011B0 7212              <1>         jb      short K37B            ; NO FN, HANDLE W/OTHER EXTENDED
1018 000011B2 3C44              <1>       cmp   al, 68              ; IN KEYPAD REGION?
1019                            <1>         ;ja  short K32A         ; IF SO, IGNORE
1020 000011B4 0F87FCFEFFFF      <1>         ja      K26
1021 000011BA 80C42D            <1>       add   ah, 45              ; CONVERT TO PSEUDO SCAN CODE
1022                            <1> K37A:
1023 000011BD B000              <1>       mov   al, 0               ; ASCII CODE OF ZERO
1024 000011BF E991010000        <1>         jmp     K57                   ; PUT IT IN THE BUFFER
1025                            <1> K37B:
1026 000011C4 B0F0              <1>       mov   al, 0F0h            ; USE SPECIAL ASCII CODE
1027 000011C6 E98A010000        <1>       jmp     K57                   ; PUT IT IN THE BUFFER
1028                            <1> K37C:
1029 000011CB 0450              <1>       add   al, 80              ; CONVERT SCAN CODE (EDIT KEYS)
1030 000011CD 88C4              <1>       mov   ah, al              ; (SCAN CODE NOT IN AH FOR INSERT)
1031 000011CF 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 000011D1 F6C304            <1>       test  bl, CTL_SHIFT             ; ARE WE IN CONTROL SHIFT?
1037                            <1>       ;jnz  short K38A          ; YES, START PROCESSING
1038 000011D4 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 000011DA 3C46              <1>       cmp   al, SCROLL_KEY            ; TEST FOR BREAK
1044 000011DC 7531              <1>       jne   short K39           ; JUMP, NO-BREAK
1045 000011DE F6C710            <1>       test  bh, KBX                   ; IS THIS THE ENHANCED KEYBOARD?
1046 000011E1 7405              <1>       jz    short K38B          ; NO, BREAK IS VALID
1047 000011E3 F6C702            <1>       test  bh, LC_E0           ; YES, WAS LAST CODE AN E0?
1048 000011E6 7427              <1>       jz    short K39           ; NO-BREAK, TEST FOR PAUSE
1049                            <1> K38B:
1050 000011E8 8B1D[9A5E0000]    <1>       mov   ebx, [BUFFER_HEAD]  ; RESET BUFFER TO EMPTY
1051 000011EE 891D[9E5E0000]    <1>       mov   [BUFFER_TAIL], ebx
1052 000011F4 C605[8C5E0000]80  <1>       mov   byte [BIOS_BREAK], 80h  ; TURN ON BIOS_BREAK BIT
1053                            <1>       ;
1054                            <1>       ;----- ENABLE KEYBOARD
1055 000011FB B0AE              <1>       mov   al, ENA_KBD         ; ENABLE KEYBOARD
1056 000011FD 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 00001202 E8CF510000        <1>       call  ctrlbrk ; control+break subroutine
1062                            <1>       ;
1063 00001207 6629C0            <1>       sub   ax, ax              ; PUT OUT DUMMY CHARACTER
1064 0000120A E946010000        <1>         jmp     K57                   ; BUFFER_FILL
1065                            <1>       ;
1066                            <1>       ;----- TEST FOR PAUSE
1067                            <1> K39:                            ; NO_BREAK
1068 0000120F F6C710            <1>       test  bh, KBX                   ; IS THIS THE ENHANCED KEYBOARD?
1069 00001212 7537              <1>       jnz   short K41           ; YES, THEN THIS CAN'T BE PAUSE
1070 00001214 3C45              <1>       cmp   al, NUM_KEY         ; LOOK FOR PAUSE KEY
1071 00001216 7533              <1>       jne   short K41           ; NO-PAUSE
1072                            <1> K39P:
1073 00001218 800D[8E5E0000]08  <1>       or    byte [KB_FLAG_1], HOLD_STATE ; TURN ON THE HOLD FLAG
1074                            <1>       ;
1075                            <1>       ;----- ENABLE KEYBOARD
1076 0000121F B0AE              <1>       mov   al, ENA_KBD         ; ENABLE KEYBOARD
1077 00001221 E89B010000        <1>       call  SHIP_IT                   ; EXECUTE ENABLE
1078                            <1> K39A:
1079 00001226 B020              <1>       mov   al, EOI                   ; END OF INTERRUPT TO CONTROL PORT
1080 00001228 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 0000122A 803D[C25E0000]07  <1>         cmp     byte [CRT_MODE], 7      ; IS THIS BLACK AND WHITE CARD
1084 00001231 740A              <1>         je      short K40                   ; YES, NOTHING TO DO
1085 00001233 66BAD803          <1>       mov   dx, 03D8h           ; PORT FOR COLOR CARD
1086 00001237 A0[C35E0000]      <1>         mov     al, [CRT_MODE_SET]    ; GET THE VALUE OF THE CURRENT MODE
1087 0000123C EE                <1>       out   dx, al              ; SET THE CRT MODE, SO THAT CRT IS ON
1088                            <1>       ;
1089                            <1> K40:                            ; PAUSE-LOOP
1090 0000123D F605[8E5E0000]08  <1>         test    byte [KB_FLAG_1], HOLD_STATE ; CHECK HOLD STATE FLAG
1091 00001244 75F7              <1>       jnz   short K40           ; LOOP UNTIL FLAG TURNED OFF
1092                            <1>       ;
1093 00001246 E977FEFFFF        <1>         jmp     K27                   ; INTERRUPT_RETURN_NO_EOI
1094                            <1>         ;
1095                            <1>       ;----- TEST SPECIAL CASE KEY 55
1096                            <1> K41:                            ; NO-PAUSE
1097 0000124B 3C37              <1>       cmp   al, 55              ; TEST FOR */PRTSC KEY
1098 0000124D 7513              <1>       jne   short K42           ; NOT-KEY-55
1099 0000124F F6C710            <1>       test  bh, KBX                   ; IS THIS THE ENHANCED KEYBOARD?
1100 00001252 7405              <1>       jz    short K41A          ; NO, CTL-PRTSC IS VALID
1101 00001254 F6C702            <1>       test  bh, LC_E0           ; YES, WAS LAST CODE AN E0?
1102 00001257 7421              <1>       jz    short K42B          ; NO, TRANSLATE TO A FUNCTION
1103                            <1> K41A:
1104 00001259 66B80072          <1>       mov   ax, 114*256         ; START/STOP PRINTING SWITCH
1105 0000125D E9F3000000        <1>         jmp     K57                   ; BUFFER_FILL
1106                            <1>       ;
1107                            <1>       ;----- SET UP TO TRANSLATE CONTROL SHIFT
1108                            <1> K42:                            ; NOT-KEY-55
1109 00001262 3C0F              <1>       cmp   al, 15              ; IS IT THE TAB KEY?
1110 00001264 7414              <1>       je    short K42B          ; YES, XLATE TO FUNCTION CODE
1111 00001266 3C35              <1>       cmp   al, 53              ; IS IT THE / KEY?
1112 00001268 750E              <1>       jne   short K42A          ; NO, NO MORE SPECIAL CASES
1113 0000126A F6C702            <1>       test  bh, LC_E0           ; YES, IS IT FROM THE KEY PAD?
```

```
1114 0000126D 7409              <1>       jz    short K42A          ; NO, JUST TRANSLATE
1115 0000126F 66B80095          <1>       mov   ax, 9500h           ; YES, SPECIAL CODE FOR THIS ONE
1116 00001273 E9DD000000        <1>       jmp   K57                 ; BUFFER FILL
1117                            <1> K42A:
1118                            <1>       ;;mov ebx, _K8            ; SET UP TO TRANSLATE CTL
1119 00001278 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 0000127A BB[845D0000]      <1>       mov   ebx, _K8            ; SET UP TO TRANSLATE CTL
1125 0000127F 0F82AE000000      <1>       jb    K56 ;; 20/02/2015
1126 00001285 E9B9000000        <1>       jmp   K64
1127                            <1>         ;
1128                            <1>       ;----- NOT IN CONTROL SHIFT
1129                            <1> K44:                            ; NOT-CTL-SHIFT
1130 0000128A 3C37              <1>       cmp   al, 55              ; PRINT SCREEN KEY?
1131 0000128C 7528              <1>       jne   short K45           ; NOT PRINT SCREEN
1132 0000128E F6C710            <1>       test  bh, KBX                   ; IS THIS ENHANCED KEYBOARD?
1133 00001291 7407              <1>       jz    short K44A          ; NO, TEST FOR SHIFT STATE
1134 00001293 F6C702            <1>       test  bh, LC_E0           ; YES, LAST CODE A MARKER?
1135 00001296 7507              <1>       jnz   short K44B          ; YES, IS PRINT SCREEN
1136 00001298 EB41              <1>       jmp   short K45C          ; NO, TRANSLATE TO '*' CHARACTER
1137                            <1> K44A:
1138 0000129A F6C303            <1>       test  bl, LEFT_SHIFT+RIGHT_SHIFT ; NOT 101 KBD, SHIFT KEY DOWN?
1139 0000129D 743C              <1>       jz    short K45C          ; NO, TRANSLATE TO '*' CHARACTER
1140                            <1>       ;
1141                            <1>       ;----- ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
1142                            <1> K44B:
1143 0000129F B0AE              <1>       mov   al, ENA_KBD         ; INSURE KEYBOARD IS ENABLED
1144 000012A1 E81B010000        <1>       call  SHIP_IT                   ; EXECUTE ENABLE
1145 000012A6 B020              <1>       mov   al, EOI                   ; END OF CURRENT INTERRUPT
1146 000012A8 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 000012AA 8025[905E0000]FC  <1>         and     byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; ZERO OUT THESE FLAGS
1152 000012B1 E90CFEFFFF        <1>         jmp     K27                     ; GO BACK WITHOUT EOI OCCURRING
1153                            <1>       ;
1154                            <1>       ;----- HANDLE IN-CORE KEYS
1155                            <1> K45:                            ; NOT-PRINT-SCREEN
1156 000012B6 3C3A              <1>       cmp   al, 58              ; TEST FOR IN-CORE AREA
1157 000012B8 7734              <1>       ja    short K46           ; JUMP IF NOT
1158 000012BA 3C35              <1>       cmp   al, 53              ; IS THIS THE '/' KEY?
1159 000012BC 7505              <1>       jne   short K45A          ; NO, JUMP
1160 000012BE F6C702            <1>       test  bh, LC_E0           ; WAS THE LAST CODE THE MARKER?
1161 000012C1 7518              <1>       jnz   short K45C          ; YES, TRANSLATE TO CHARACTER
1162                            <1> K45A:
1163 000012C3 B91A000000        <1>       mov   ecx, 26                   ; LENGHT OF SEARCH
1164 000012C8 BF[5A5D0000]      <1>       mov   edi, K30+10         ; POINT TO TABLE OF A-Z CHARS
1165 000012CD F2AE              <1>       repne scasb               ; IS THIS A LETTER KEY?
1166                            <1>             ; 20/02/2015
1167 000012CF 7505              <1>       jne   short K45B              ; NO, SYMBOL KEY
1168                            <1>       ;
1169 000012D1 F6C340            <1>       test  bl, CAPS_STATE            ; ARE WE IN CAPS_LOCK?
1170 000012D4 750C              <1>       jnz   short K45D          ; TEST FOR SURE
1171                            <1> K45B:
1172 000012D6 F6C303            <1>       test  bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
1173 000012D9 750C              <1>       jnz   short K45E          ; YES, UPPERCASE
1174                            <1>                                 ; NO, LOWERCASE
1175                            <1> K45C:
1176 000012DB BB[DC5D0000]      <1>       mov   ebx, K10            ; TRANSLATE TO LOWERCASE LETTERS
1177 000012E0 EB51              <1>       jmp   short K56
1178                            <1> K45D:                           ; ALMOST-CAPS-STATE
1179 000012E2 F6C303            <1>       test  bl, LEFT_SHIFT+RIGHT_SHIFT ; CL ON. IS SHIFT ON, TOO?
1180 000012E5 75F4              <1>       jnz   short K45C          ; SHIFTED TEMP OUT OF CAPS STATE
1181                            <1> K45E:
1182 000012E7 BB[345E0000]      <1>       mov   ebx, K11            ; TRANSLATE TO UPPER CASE LETTERS
1183 000012EC EB45              <1> K45F: jmp   short K56
1184                            <1>       ;
1185                            <1>       ;----- TEST FOR KEYS F1 - F10
1186                            <1> K46:                            ; NOT IN-CORE AREA
1187 000012EE 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 000012F0 7635              <1>       jna   short K53
1191                            <1>       ;
1192                            <1>       ;----- HANDLE THE NUMERIC PAD KEYS
1193                            <1> K47:                            ; NOT F1 - F10
1194 000012F2 3C53              <1>       cmp   al, 83              ; TEST NUMPAD KEYS
1195 000012F4 772D              <1>       ja    short K52           ; JUMP IF NOT
1196                            <1>       ;
1197                            <1>       ;----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
1198                            <1> K48:
1199 000012F6 3C4A              <1>       cmp   al , 74                   ; SPECIAL CASE FOR MINUS
1200 000012F8 74ED              <1>       je    short K45E          ; GO TRANSLATE
1201 000012FA 3C4E              <1>       cmp   al , 78                   ; SPECIAL CASE FOR PLUS
1202 000012FC 74E9              <1>       je    short K45E          ; GO TRANSLATE
1203 000012FE F6C702            <1>       test  bh, LC_E0           ; IS THIS ONE OFTHE NEW KEYS?
1204 00001301 750A              <1>       jnz   short K49           ; YES, TRANSLATE TO BASE STATE
1205                            <1>       ;
1206 00001303 F6C320            <1>       test  bl, NUM_STATE       ; ARE WE IN NUM LOCK
1207 00001306 7514              <1>       jnz   short K50           ; TEST FOR SURE
1208 00001308 F6C303            <1>       test  bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
1209                            <1>       ;jnz  short K51           ; IF SHIFTED, REALLY NUM STATE
1210 0000130B 75DA              <1>       jnz   short K45E
1211                            <1>       ;
1212                            <1>       ;----- BASE CASE FOR KEYPAD
1213                            <1> K49:
1214 0000130D 3C4C              <1>       cmp   al, 76              ; SPECIAL CASE FOR BASE STATE 5
1215 0000130F 7504              <1>       jne   short K49A          ; CONTINUE IF NOT KEYPAD 5
```

```
1216 00001311 B0F0                <1>       mov    al, 0F0h           ; SPECIAL ASCII CODE
1217 00001313 EB40                <1>       jmp    short K57          ; BUFFER FILL
1218                              <1> K49A:
1219 00001315 BB[DC5D0000]        <1>       mov    ebx, K10           ; BASE CASE TABLE
1220 0000131A 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 0000131C F6C303              <1>         test    bl, LEFT_SHIFT+RIGHT_SHIFT
1225 0000131F 75EC                <1>       jnz    short K49          ; SHIFTED TEMP OUT OF NUM STATE
1226 00001321 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 00001323 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 00001325 74AF                <1>       je     short K45B
1234                              <1>       ;
1235                              <1>       ;----- MUST BE F11 OR F12
1236                              <1> K53:                            ; F1 - F10 COME HERE, TOO
1237 00001327 F6C303              <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT ; TEST SHIFT STATE
1238 0000132A 74E1                <1>       jz     short K49          ; JUMP, LOWER CASE PSEUDO SC'S
1239                              <1>              ; 20/02/2015
1240 0000132C BB[345E0000]        <1>       mov    ebx, K11           ; UPPER CASE PSEUDO SCAN CODES
1241 00001331 EB10                <1>       jmp    short K64          ; TRANSLATE SCAN
1242                              <1>       ;
1243                              <1>       ;----- TRANSLATE THE CHARACTER
1244                              <1> K56:                            ; TRANSLATE-CHAR
1245 00001333 FEC8                <1>       dec    al                 ; CONVERT ORIGIN
1246 00001335 D7                  <1>       xlat                      ; CONVERT THE SCAN CODE TO ASCII
1247 00001336 F605[905E0000]02    <1>       test   byte [KB_FLAG_3], LC_E0   ; IS THIS A NEW KEY?
1248 0000133D 7416                <1>       jz     short K57          ; NO, GO FILL BUFFER
1249 0000133F B4E0                <1>       mov    ah, MC_E0          ; YES, PUT SPECIAL MARKER IN AH
1250 00001341 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 00001343 FEC8                <1>       dec    al                 ; CONVERT ORIGIN
1255 00001345 D7                  <1>              xlat                       ; CTL TABLE SCAN
1256 00001346 88C4                <1>       mov    ah, al             ; PUT VALUE INTO AH
1257 00001348 B000                <1>       mov    al, 0              ; ZERO ASCII CODE
1258 0000134A F605[905E0000]02    <1>       test   byte [KB_FLAG_3], LC_E0   ; IS THIS A NEW KEY?
1259 00001351 7402                <1>       jz     short K57          ; NO, GO FILL BUFFER
1260 00001353 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 00001355 3CFF                <1>       cmp    al, -1             ; IS THIS AN IGNORE CHAR
1265                              <1>         ;je  short K59          ; YES, DO NOTHING WITH IT
1266 00001357 0F8459FDFFFF        <1>       je      K26               ; YES, DO NOTHING WITH IT
1267 0000135D 80FCFF              <1>       cmp    ah, -1             ; LOOK FOR -1 PSEUDO SCAN
1268                              <1>         ;jne short K61          ; NEAR_INTERRUPT_RETURN
1269 00001360 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 00001366 80FC68              <1>       cmp    ah, 68h      ; ALT + F1 key
1275 00001369 721F                <1>       jb     short K61
1276 0000136B 80FC6F              <1>       cmp    ah, 6Fh ; ALT + F8 key
1277 0000136E 771A                <1>       ja     short K61
1278                              <1>       ;
1279 00001370 8A1D[2E520100]      <1>       mov    bl, [ACTIVE_PAGE]
1280 00001376 80C368              <1>       add    bl, 68h
1281 00001379 38E3                <1>       cmp    bl, ah
1282 0000137B 740D                <1>       je     short K61
1283 0000137D 6650                <1>       push   ax
1284 0000137F 88E0                <1>       mov    al, ah
1285 00001381 2C68                <1>       sub    al, 68h
1286 00001383 E8F4050000          <1>       call   set_active_page
1287 00001388 6658                <1>       pop    ax
1288                              <1> K61:                            ; NOT-CAPS-STATE
1289 0000138A 8B1D[9E5E0000]      <1>       mov    ebx, [BUFFER_TAIL]  ; GET THE END POINTER TO THE BUFFER
1290 00001390 89DE                <1>       mov    esi, ebx            ; SAVE THE VALUE
1291 00001392 E857FAFFFF          <1>       call   _K4                 ; ADVANCE THE TAIL
1292 00001397 3B1D[9A5E0000]      <1>       cmp    ebx, [BUFFER_HEAD]  ; HAS THE BUFFER WRAPPED AROUND
1293 0000139D 740E                <1>       je     short K62           ; BUFFER_FULL_BEEP
1294 0000139F 668906              <1>       mov    [esi], ax           ; STORE THE VALUE
1295 000013A2 891D[9E5E0000]      <1>       mov    [BUFFER_TAIL], ebx  ; MOVE THE POINTER UP
1296 000013A8 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 000013AD B020                <1>       mov    al, EOI                  ; ENABLE INTERRUPT CONTROLLER CHIP
1311 000013AF E620                <1>       out    INTA00, al
1312 000013B1 66B9A602            <1>       mov    cx, 678                  ; DIVISOR FOR 1760 HZ
1313 000013B5 B304                <1>       mov    bl, 4               ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)
1314 000013B7 E8E5090000          <1>       call   beep                ; GO TO COMMON BEEP HANDLER
1315 000013BC 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 000013C1 6650             <1>     push    ax                      ; SAVE DATA TO SEND
1325                            <1>
1326                            <1>     ;-----  WAIT FOR COMMAND TO ACCEPTED
1327 000013C3 FA               <1>     cli                             ; DISABLE INTERRUPTS TILL DATA SENT
1328                            <1>     ; xor   ecx, ecx                ; CLEAR TIMEOUT COUNTER
1329 000013C4 B900000100       <1>     mov     ecx, 10000h
1330                            <1> S10:
1331 000013C9 E464             <1>     in      al, STATUS_PORT                 ; READ KEYBOARD CONTROLLER STATUS
1332 000013CB A802             <1>     test    al, INPT_BUF_FULL       ; CHECK FOR ITS INPUT BUFFER BUSY
1333 000013CD E0FA             <1>     loopnz  S10                     ; WAIT FOR COMMAND TO BE ACCEPTED
1334                            <1>
1335 000013CF 6658             <1>     pop     ax                      ; GET DATA TO SEND
1336 000013D1 E664             <1>     out     STATUS_PORT, al                 ; SEND TO KEYBOARD CONTROLLER
1337 000013D3 FB               <1>     sti                             ; ENABLE INTERRUPTS AGAIN
1338 000013D4 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 000013D5 6650             <1>     push    ax                      ; SAVE REGISTERS
1349 000013D7 6653             <1>     push    bx
1350 000013D9 51               <1>     push    ecx
1351 000013DA 88C7             <1>     mov     bh, al                  ; SAVE TRANSMITTED BYTE FOR RETRIES
1352 000013DC B303             <1>     mov     bl, 3                   ; LOAD RETRY COUNT
1353                            <1> SD0:
1354 000013DE FA               <1>     cli                             ; DISABLE INTERRUPTS
1355 000013DF 8025[8F5E0000]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 000013E6 B900000100       <1>     mov     ecx, 10000h             ; MAXIMUM WAIT COUNT
1359                            <1> SD5:
1360 000013EB E464             <1>     in      al, STATUS_PORT                 ; READ KEYBOARD PROCESSOR STATUS PORT
1361 000013ED A802             <1>     test    al, INPT_BUF_FULL       ; CHECK FOR ANY PENDING COMMAND
1362 000013EF E0FA             <1>     loopnz  SD5                     ; WAIT FOR COMMAND TO BE ACCEPTED
1363                            <1>     ;
1364 000013F1 88F8             <1>     mov     al, bh                  ; REESTABLISH BYTE TO TRANSMIT
1365 000013F3 E660             <1>     out     PORT_A, al              ; SEND BYTE
1366 000013F5 FB               <1>     sti                             ; ENABLE INTERRUPTS
1367                            <1>     ;mov   cx, 01A00h              ; LOAD COUNT FOR 10 ms+
1368 000013F6 B9FFFF0000       <1>     mov     ecx, 0FFFFh
1369                            <1> SD1:
1370 000013FB F605[8F5E0000]30 <1>     test    byte [KB_FLAG_2], KB_FE+KB_FA ; SEE IF EITHER BIT SET
1371 00001402 750F             <1>     jnz     short SD3               ; IF SET, SOMETHING RECEIVED GO PROCESS
1372 00001404 E2F5             <1>     loop    SD1                     ; OTHERWISE WAIT
1373                            <1> SD2:
1374 00001406 FECB             <1>     dec     bl                      ; DECREMENT RETRY COUNT
1375 00001408 75D4             <1>     jnz     short SD0               ; RETRY TRANSMISSION
1376 0000140A 800D[8F5E0000]80 <1>     or      byte [KB_FLAG_2], KB_ERR ; TURN ON TRANSMIT ERROR FLAG
1377 00001411 EB09             <1>     jmp     short SD4               ; RETRIES EXHAUSTED FORGET TRANSMISSION
1378                            <1> SD3:
1379 00001413 F605[8F5E0000]10 <1>     test    byte [KB_FLAG_2], KB_FA ; SEE IF THIS IS AN ACKNOWLEDGE
1380 0000141A 74EA             <1>     jz      short SD2               ; IF NOT, GO RESEND
1381                            <1> SD4:
1382 0000141C 59               <1>     pop     ecx                     ; RESTORE REGISTERS
1383 0000141D 665B             <1>     pop     bx
1384 0000141F 6658             <1>     pop     ax
1385 00001421 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 00001422 FA               <1>     cli                             ; TURN OFF INTERRUPTS
1395 00001423 F605[8F5E0000]40 <1>     test    byte [KB_FLAG_2], KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
1396 0000142A 755F             <1>     jnz     short SL1               ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
1397                            <1>     ;
1398 0000142C 800D[8F5E0000]40 <1>     or      byte [KB_FLAG_2], KB_PR_LED ; TURN ON UPDATE IN PROCESS
1399 00001433 B020             <1>     mov     al, EOI                 ; END OF INTERRUPT COMMAND
1400 00001435 E620             <1>     out     20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
1401 00001437 EB11             <1>     jmp     short SL0               ; GO SEND MODE INDICATOR COMMAND
1402                            <1> SND_LED1:
1403 00001439 FA               <1>     cli                             ; TURN OFF INTERRUPTS
1404 0000143A F605[8F5E0000]40 <1>     test    byte [KB_FLAG_2], KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
1405 00001441 7548             <1>     jnz     short SL1               ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
1406                            <1>     ;
1407 00001443 800D[8F5E0000]40 <1>     or      byte [KB_FLAG_2], KB_PR_LED ; TURN ON UPDATE IN PROCESS
1408                            <1> SL0:
1409 0000144A B0ED             <1>     mov     al, LED_CMD             ; LED CMD BYTE
1410 0000144C E884FFFFFF       <1>     call    SND_DATA                ; SEND DATA TO KEYBOARD
1411 00001451 FA               <1>     cli
1412 00001452 E836000000       <1>     call    MAKE_LED                ; GO FORM INDICATOR DATA BYTE
1413 00001457 8025[8F5E0000]F8 <1>     and     byte [KB_FLAG_2], 0F8h    ; ~KB_LEDS ; CLEAR MODE INDICATOR BITS
1414 0000145E 0805[8F5E0000]   <1>     or      [KB_FLAG_2], al     ; SAVE PRESENT INDICATORS FOR NEXT TIME
1415 00001464 F605[8F5E0000]80 <1>     test    byte [KB_FLAG_2], KB_ERR ; TRANSMIT ERROR DETECTED
1416 0000146B 750F             <1>     jnz     short SL2               ; IF SO, BYPASS SECOND BYTE TRANSMISSION
1417                            <1>     ;
1418 0000146D E863FFFFFF       <1>     call    SND_DATA                ; SEND DATA TO KEYBOARD
1419 00001472 FA               <1>     cli                             ; TURN OFF INTERRUPTS
```

```
1420 00001473 F605[8F5E0000]80    <1>       test   byte [KB_FLAG_2], KB_ERR ; TRANSMIT ERROR DETECTED
1421 0000147A 7408                <1>       jz     short SL3           ; IF NOT, DON'T SEND AN ENABLE COMMAND
1422                              <1> SL2:
1423 0000147C B0F4                <1>       mov    al, KB_ENABLE       ; GET KEYBOARD CSA ENABLE COMMAND
1424 0000147E E852FFFFFF          <1>       call   SND_DATA            ; SEND DATA TO KEYBOARD
1425 00001483 FA                  <1>       cli                        ; TURN OFF INTERRUPTS
1426                              <1> SL3:
1427 00001484 8025[8F5E0000]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 0000148B FB                  <1>       sti                        ; ENABLE INTERRUPTS
1430 0000148C 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 0000148D A0[8D5E0000]        <1>       mov    al, [KB_FLAG]       ; GET CAPS & NUM LOCK INDICATORS
1441 00001492 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 00001494 C0C004              <1>       rol    al, 4 ; 20/02/2015
1445 00001497 2407                <1>       and    al, 07h                  ; MAKE SURE ONLY MODE BITS ON
1446                              <1>       ;pop   cx
1447 00001499 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 ///
1939
1940                                  %include 'video.s' ; 07/03/2015
   1                              <1> ; ****************************************************************************
   2                              <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - video.s
   3                              <1> ; ----------------------------------------------------------------------------
   4                              <1> ; Last Update: 15/01/2017
   5                              <1> ; ----------------------------------------------------------------------------
   6                              <1> ; Beginning: 16/01/2016
   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> ; 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 0000149A 9C                  <1>       pushfd
  35 0000149B 0E                  <1>       push   cs
  36 0000149C E851000000          <1>       call   VIDEO_IO_1
  37 000014A1 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                                  :
  64                              <1> ;            (DH,DL) = ROW,COLUMN  (00H,00H) IS UPPER LEFT            :
  65                              <1> ;            (BH) = A PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES)          :
  66                              <1> ;    (AH)= 03H     READ CURSOR POSITION                                       :
  67                              <1> ;            (BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES)      :
```

```
68                                      <1> ;           ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR                  :
69                                      <1> ;                   (CH,CL) = CURSOR MODE CURRENTLY SET             :
70                                      <1> ;    (AH)= 04H     READ LIGHT PEN POSITION                                  :
71                                      <1> ;           ON EXIT:                                                :
72                                      <1> ;           (AH) = 00H -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED    :
73                                      <1> ;           (AH) = 01H -- VALID LIGHT PEN VALUE IN REGISTERS         :
74                                      <1> ;                   (DH,DL) = ROW,COLUMN OF CHARACTER LP POSITION    :
75                                      <1> ;                   (CH) = RASTER LINE (0-199)                       :
76                                      <1> ;                   (BX) = PIXEL COLUMN (0-319,639)                  :
77                                      <1> ;    (AH)= 05H     SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES)          :
78                                      <1> ;           (AL) = NEW PAGE VALUE (0-7 FOR MODES 0&1, 0-3 FOR MODES 2&3)     :
79                                      <1> ;    (AH)= 06H     SCROLL ACTIVE PAGE UP                                    :
80                                      <1> ;           (AL) = NUMBER OF LINES. ( LINES BLANKED AT BOTTOM OF WINDOW )    :
81                                      <1> ;                   (AL) = 00H MEANS BLANK ENTIRE WINDOW             :
82                                      <1> ;           (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL      :
83                                      <1> ;           (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL     :
84                                      <1> ;           (BH) = ATTRIBUTE TO BE USED ON BLANK LINE                :
85                                      <1> ;    (AH)= 07H     SCROLL ACTIVE PAGE DOWN                                  :
86                                      <1> ;           (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP OF WINDOW     :
87                                      <1> ;                   (AL) = 00H MEANS BLANK ENTIRE WINDOW             :
88                                      <1> ;           (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL      :
89                                      <1> ;           (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL     :
90                                      <1> ;           (BH) = ATTRIBUTE TO BE USED ON BLANK LINE                :
91                                      <1> ;                                                                    :
92                                      <1> ;   CHARACTER HANDLING ROUTINES                                      :
93                                      <1> ;                                                                    :
94                                      <1> ;    (AH)= 08H     READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION      :
95                                      <1> ;           (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)         :
96                                      <1> ;           ON EXIT:                                                 :
97                                      <1> ;           (AL) = CHAR READ                                         :
98                                      <1> ;           (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY)            :
99                                      <1> ;    (AH)= 09H     WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION      :
100                                     <1> ;           (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)         :
101                                     <1> ;           (CX) = COUNT OF CHARACTERS TO WRITE                      :
102                                     <1> ;           (AL) = CHAR TO WRITE                                     :
103                                     <1> ;           (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR (GRAPHICS)   :
104                                     <1> ;                     SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.     :
105                                     <1> ;    (AH) = 0AH    WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION                  :
106                                     <1> ;           (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)         :
107                                     <1> ;           (CX) = COUNT OF CHARACTERS TO WRITE                      :
108                                     <1> ;           (AL) = CHAR TO WRITE                                     :
109                                     <1> ;                   NOTE: USE FUNCTION (AH)= 09H IN GRAPHICS MODES           :
110                                     <1> ;    FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE           :
111                                     <1> ;           CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE    :
112                                     <1> ;           MAINTAINED IN THE SYSTEM ROM. ONLY THE 1ST 128 CHARS     :
113                                     <1> ;           ARE CONTAINED THERE. TO READ/WRITE THE SECOND 128 CHARS,  :
114                                     <1> ;           THE USER MUST INITIALIZE THE POINTER AT INTERRUPT 1FH     :
115                                     <1> ;           (LOCATION 0007CH) TO POINT TO THE 1K BYTE TABLE CONTAINING :
116                                     <1> ;           THE CODE POINTS FOR THE SECOND 128 CHARS (128-255).      :
117                                     <1> ;    FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION FACTOR  :
118                                     <1> ;           CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID RESULTS ONLY :
119                                     <1> ;           FOR CHARACTERS CONTAINED ON THE SAME ROW. CONTINUATION TO  :
120                                     <1> ;           SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY.              :
121                                     <1> ;                                                                    :
122                                     <1> ;    GRAPHICS INTERFACE                                              :
123                                     <1> ;    (AH)= 0BH     SET COLOR PALETTE                                 :
124                                     <1> ;           (BH) = PALETTE COLOR ID BEING SET (0-127)                :
125                                     <1> ;           (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID          :
126                                     <1> ;                   NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT HAS    :
127                                     <1> ;                           MEANING ONLY FOR 320X200 GRAPHICS.       :
128                                     <1> ;                   COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15)          :
129                                     <1> ;                   COLOR ID = 1 SELECTS THE PALETTE TO BE USED:      :
130                                     <1> ;                           0 = GREEN(1)/RED(2)/YELLOW(3)            :
131                                     <1> ;                           1 = CYAN(1)/MAGENTA(2)/WHITE(3)                  :
132                                     <1> ;                   IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET FOR   :
133                                     <1> ;                           PALETTE COLOR 0 INDICATES THE BORDER COLOR  :
134                                     <1> ;                           TO BE USED (VALUES 0-31, WHERE 16-31 SELECT :
135                                     <1> ;                           THE HIGH INTENSITY BACKGROUND SET.       :
136                                     <1> ;    (AH)= 0CH     WRITE DOT                                         :
137                                     <1> ;           (DX) = ROW NUMBER                                        :
138                                     <1> ;           (CX) = COLUMN NUMBER                                     :
139                                     <1> ;           (AL) = COLOR VALUE                                       :
140                                     <1> ;                   IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS EXCLUSIVE    :
141                                     <1> ;                   ORed WITH THE CURRENT CONTENTS OF THE DOT         :
142                                     <1> ;    (AH)= ODH     READ DOT                                          :
143                                     <1> ;           (DX) = ROW NUMBER                                        :
144                                     <1> ;           (CX) = COLUMN NUMBER                                     :
145                                     <1> ;           (AL) = RETURNS THE DOT READ                              :
146                                     <1> ;                                                                    :
147                                     <1> ;    ASCII TELETYPE ROUTINE FOR OUTPUT                               :
148                                     <1> ;                                                                    :
149                                     <1> ;    (AH)= 0EH     WRITE TELETYPE TO ACTIVE PAGE                             :
150                                     <1> ;           (AL) = CHAR TO WRITE                                     :
151                                     <1> ;           (BL) = FOREGROUND COLOR IN GRAPHICS MODE                 :
152                                     <1> ;           NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET          :
153                                     <1> ;    (AH)= 0FH     CURRENT VIDEO STATE                               :
154                                     <1> ;           RETURNS THE CURRENT VIDEO STATE                          :
155                                     <1> ;           (AL) = MODE CURRENTLY SET ( SEE (AH)=00H FOR EXPLANATION)  :
156                                     <1> ;           (AH) = NUMBER OR CHARACTER COLUMNS ON SCREEN              :
157                                     <1> ;           (BH) = CURRENT ACTIVE DISPLAY PAGE                       :
158                                     <1> ;    (AH)= 10H     RESERVED                                          :
159                                     <1> ;    (AH)= 11H     RESERVED                                          :
160                                     <1> ;    (AH)= 12H     RESERVED                                          :
161                                     <1> ;    (AH)= 13H     WRITE STRING                                      :
162                                     <1> ;                   ES:BP  -  POINTER T0 STRING TO BE WRITTEN        :
163                                     <1> ;                   CX     -  LENGTH OF CHARACTER STRING TO WRITTEN          :
164                                     <1> ;                   DX     -  CURSOR POSITION FOR STRING TO BE WRITTEN  :
165                                     <1> ;                   BH     -  PAGE NUMBER                            :
166                                     <1> ;           (AL)= 00H    WRITE CHARACTER STRING                      :
167                                     <1> ;                   BL     -  ATTRIBUTE                              :
168                                     <1> ;                   STRING IS  <CHAR,CHAR, ... ,CHAR>                :
169                                     <1> ;                   CURSOR NOT MOVED                                 :
```

```
                                <1> ;           (AL)= 01H    WRITE CHARACTER STRING AND MOVE CURSOR        :
                                <1> ;                 BL     -  ATTRIBUTE                              :
                                <1> ;                 STRING IS  <CHAR,CHAR, ... ,CHAR>                :
                                <1> ;                 CURSOR MOVED                                     :
                                <1> ;           (AL)= 02H    WRITE CHARACTER AND ATTRIBUTE STRING          :
                                <1> ;                       (VALID FOR ALPHA MODES ONLY)                :
                                <1> ;                 STRING IS <CHAR,ATTR,CHAR,ATTR ..  ,CHAR,ATTR>             :
                                <1> ;                 CURSOR IS NOT MOVED                              :
                                <1> ;           (AL)= 03H WRITE CHARACTER AND ATTRIBUTE STRING AND MOVE CURSOR    :
                                <1> ;                       (VALID FOR ALPHA MODES ONLY)                :
                                <1> ;                 STRING IS <CHAR,ATTR,CHAR,ATTR ..  ,CHAR,ATTR>             :
                                <1> ;                 CURSOR IS MOVED                                  :
                                <1> ;            NOTE:  CARRIAGE RETURN, LINE FEED, BACKSPACE, AND BELL ARE       :
                                <1> ;                   TREATED AS COMMANDS RATHER THAN PRINTABLE CHARACTERS.     :
                                <1> ;                                                                    :
                                <1> ;    BX,CX,DX,SI,DI,BP,SP,DS,ES,SS PRESERVED DURING CALLS EXCEPT FOR         :
                                <1> ;    BX,CX,DX RETURN VALUES ON FUNCTIONS 03H,04H,0DH AND 0FH. ON ALL CALLS   :
                                <1> ;    AX IS MODIFIED.                                                 :
                                <1> ;-----------------------------------------------------------------------------
                                <1>
000014A2 [4F150000]             <1> M1:   dd    SET_MODE     ; TABLE OF ROUTINES WITHIN VIDEO I/O
000014A6 [B7180000]             <1>       dd    SET_CTYPE
000014AA [EB180000]             <1>       dd    SET_CPOS
000014AE [13190000]             <1>       dd    READ_CURSOR
                                <1>       ;dd   VIDEO_RETURN ; READ_LPEN
000014B2 [38150000]             <1>       dd    set_mode_ncm ; Set mode without clearing video memory
000014B6 [59190000]             <1>       dd    ACT_DISP_PAGE
000014BA [F0190000]             <1>       dd    SCROLL_UP
000014BE [141B0000]             <1>       dd    SCROLL_DOWN
000014C2 [951B0000]             <1>       dd    READ_AC_CURRENT
000014C6 [ED1B0000]             <1>       dd    WRITE_AC_CURRENT
000014CA [131C0000]             <1>       dd    WRITE_C_CURRENT
000014CE [39250000]             <1>       dd    SET_COLOR
000014D2 [A4250000]             <1>       dd    WRITE_DOT
000014D6 [6F250000]             <1>       dd    READ_DOT
000014DA [951C0000]             <1>       dd    WRITE_TTY
000014DE [20150000]             <1>       dd    VIDEO_STATE
000014E2 [EF2E0000]             <1>       dd    vga_pal_funcs ; 10/08/2016 (TRDOS 386)
000014E6 [A52A0000]             <1>       dd    font_setup    ; 10/07/2016 (TRDOS 386)
000014EA [54150000]             <1>       dd    VIDEO_RETURN ; RESERVED
000014EE [021E0000]             <1>       dd    WRITE_STRING  ; 23/06/2016 (TRDOS 386)
                                <1> M1L   EQU   $ - M1
                                <1>
                                <1> ; 14/01/2017
                                <1> ; 02/01/2017
                                <1> ; 04/07/2016
                                <1> ; 12/05/2016
                                <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                                <1> int31h:  ; Video BIOS
                                <1>
                                <1> ; BH = Video page number
                                <1> ; BL = Color/Attribute
                                <1> ; AH = Function number
                                <1> ; AL = Character
                                <1>
                                <1> VIDEO_IO_1:
                                <1>       ;sti                      ; INTERRUPTS BACK ON
00001500 FC                     <1>       cld                       ; SET DIRECTION FORWARD
00001501 80FC14                 <1>       cmp   ah, M1L/4           ; TEST FOR WITHIN TABLE RANGE
00001504 7327                   <1>       jnb   short M4            ; BRANCH TO EXIT IF NOT A VALID COMMAND
                                <1>
00001506 06                     <1>       push  es
00001507 1E                     <1>       push  ds                  ; SAVE WORK AND PARAMETER REGISTERS
00001508 52                     <1>       push  edx
00001509 51                     <1>       push  ecx
0000150A 53                     <1>       push  ebx
0000150B 56                     <1>       push  esi
0000150C 57                     <1>       push  edi
0000150D 55                     <1>       push  ebp
                                <1>
0000150E 66BE1000               <1>       mov   si, KDATA           ; POINT DS: TO DATA SEGMENT
00001512 8EDE                   <1>       mov   ds, si
00001514 8EC6                   <1>       mov   es, si
00001516 BF00800B00             <1>       mov   edi, 0B8000h        ; GET offset FOR COLOR CARD
0000151B A3[8C5F0100]           <1>       mov   [video_eax], eax ; 12/05/2016
                                <1>       ; 23/03/2016
00001520 C0E402                 <1>       shl   ah, 2  ; dword            ; TIMES 2 FOR WORD TABLE LOOKUP
00001523 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
00001526 FB                     <1>       sti
                                <1>       ;
                                <1>
00001527 FFA6[A2140000]         <1>       JMP   dword [esi+M1]            ; GO TO SELECTED FUNCTION
                                <1>
                                <1> M4:                             ; COMMAND NOT VALID
0000152D CF                     <1>       iretd                     ; DO NOTHING IF NOT IN VALID RANGE
                                <1>
                                <1> VIDEO_STATE:
                                <1>       ; 26/06/2016
                                <1>       ; 12/05/2016
                                <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>
                                <1> ;-------------------------------------------------
                                <1> ; VIDEO STATE
                                <1> ;  RETURNS THE CURRENT VIDEO STATE IN AX
                                <1> ;  AH = NUMBER OF COLUMNS ON THE SCREEN
                                <1> ;  AL = CURRENT VIDEO MODE
```

```
                        <1> ;  BH = CURRENT ACTIVE PAGE
                        <1> ;--------------------------------------------------
                        <1>
00001520 8A25[C45E0000] <1>       mov    ah, [CRT_COLS]      ; GET NUMBER OF COLUMNS
00001526 A0[C25E0000]   <1>       mov    al, [CRT_MODE]      ; CURRENT MODE
                        <1>       ;movzx esi, al
                        <1>       ;mov   ah, [esi+M6]
                        <1>       ; BH = active page
0000152B 8A3D[2E520100] <1>       mov    bh, [ACTIVE_PAGE] ; GET CURRENT ACTIVE PAGE
00001531 FA             <1>       cli    ; 02/01/2017
00001532 5D             <1>       pop    ebp          ; RECOVER REGISTERS
00001533 5F             <1>       pop    edi
00001534 5E             <1>       pop    esi
00001535 59             <1>       pop    ecx          ; DISCARD SAVED BX
00001536 EB26           <1>       jmp    short M15    ; RETURN TO CALLER
                        <1>
                        <1> set_mode_ncm:
                        <1>       ; 04/07/2016 - TRDOS 386 (TRDOS v2.0)
                        <1>       ; set mode without clearing the video memory
                        <1>       ; (ony for graphics modes)
00001538 3C07           <1>       cmp    al, 7 ; IBM PC CGA modes
0000153A 7613           <1>       jna    short SET_MODE ; normal function (clear)
                        <1>       ; do not clear memory
0000153C A2[9B5F0100]   <1>       mov    [noclearmem], al ; > 0
00001541 E81F000000     <1>       call   _set_mode
00001546 C605[9B5F0100]00 <1>     mov    byte [noclearmem], 0
0000154D EB05           <1>         jmp     short VIDEO_RETURN
                        <1>
                        <1>       ; 10/08/2016
                        <1>       ; 08/08/2016
                        <1>       ; 30/07/2016
                        <1>       ; 29/07/2016
                        <1>       ; 27/07/2016
                        <1>       ; 26/07/2016
                        <1>       ; 25/07/2016
                        <1>       ; 23/07/2016
                        <1>       ; 18/07/2016
                        <1>       ; 02/07/2016
                        <1>       ; 26/06/2016
                        <1>       ; 24/06/2016
                        <1>       ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
                        <1> SET_MODE:
                        <1>       ; For 32 bit TRDOS and Retro UNIX 386:
                        <1>       ;     valid video mode: 03h only!
                        <1>       ;     (VGA modes will be selected with another routine)
                        <1>       ;
                        <1>       ; set_txt_mode ; 80*25 (16 fore colors, 8 back colors)
                        <1>
                        <1> ;--------------------------------------------------
                        <1> ; SET MODE                                       :
                        <1> ;     THIS ROUTINE INITIALIZES THE ATTACHMENT TO    :
                        <1> ;     THE SELECTED MODE, THE SCREEN IS BLANKED.     :
                        <1> ; INPUT                                           :
                        <1> ;     (AL) - MODE SELECTED (RANGE 0-7)       :
                        <1> ; OUTPUT                                         :
                        <1> ;     NONE                                       :
                        <1> ;--------------------------------------------------
                        <1>
0000154F E811000000     <1>       call   _set_mode ; 24/06/2016 (set_txt_mode)
                        <1>
                        <1> ; 12/05/2016
                        <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                        <1>
                        <1> ;-----      NORMAL RETURN FROM ALL VIDEO RETURNS
                        <1>
                        <1> VIDEO_RETURN:
00001554 A1[8C5F0100]   <1>       mov    eax, [video_eax] ; 12/05/2016
                        <1> _video_return:
00001559 FA             <1>       cli ; 02/01/2017
0000155A 5D             <1>       pop    ebp
0000155B 5F             <1>       pop    edi
0000155C 5E             <1>       pop    esi
0000155D 5B             <1>       pop    ebx
                        <1> M15:  ; VIDEO_RETURN_C
                        <1>       ;;15/01/2017
                        <1>       ; 02/01/2017
                        <1>       ;;mov  byte [intflg], 0
                        <1>       ;
0000155E 59             <1>       pop    ecx
0000155F 5A             <1>       pop    edx
00001560 1F             <1>       pop    ds
00001561 07             <1>       pop    es     ; RECOVER SEGMENTS
00001562 CF             <1>       iretd         ; ALL DONE
                        <1>
                        <1> set_txt_mode:
                        <1>       ; 29/07/2016
                        <1>       ; 27/06/2016
00001563 B003           <1>       mov    al, 3
                        <1>
                        <1> ; 10/08/2016
                        <1> ; 08/08/2016
                        <1> ; 30/07/2016
                        <1> ; 29/07/2016
                        <1> ; 27/07/2016
                        <1> ; 26/07/2016
                        <1> ; 25/07/2016
                        <1> ; 23/07/2016
                        <1> ; 18/07/2016
                        <1> ; 07/07/2016
                        <1> ; 04/07/2016
                        <1> ; 03/07/2016
                        <1> ; 02/07/2016
```

```
                                <1> ; 26/06/2016
                                <1> ; 24/06/2016 (set_txt_mode -> _set_mode)
                                <1> ; 17/06/2016
                                <1> ; 29/05/2016
                                <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                                <1> _set_mode:
                                <1>       ; 24/06/2016
00001565 3805[C25E0000]         <1>       cmp   [CRT_MODE], al  ; current mode = requested mode ?
0000156B 750D                   <1>       jne   short _sm_0
0000156D 3C03                   <1>       cmp   al, 3        ; text, 80*25 color, default mode
                                <1>                          ; for TRDOS 386 MainProg
0000156F 755F                   <1>         jne     short _sm_2     ;  multiscreen is only for mode 3
                                <1>
                                <1>       ; If '_set_mode' procedure is called for video mode 3
                                <1>       ;     while video mode is 3, video page will be cleared
                                <1>       ;     and cursor position of video page will be reset.
                                <1>
                                <1>       ; 29/07/2016
00001571 800D[995F0100]80       <1>       or     byte [p_crt_mode], 80h ; clear page indicator
00001578 EB5B                   <1>         jmp     short _sm_3
                                <1> _sm_0:
0000157A 803D[C25E0000]03       <1>       cmp   byte [CRT_MODE], 3
00001581 7534                   <1>         jne     short _sm_1
                                <1>
                                <1>       ; If '_set_mode' procedure is called for a video mode
                                <1>       ;     except video mode 3, while current video mode
                                <1>       ;     is 3; all video pages of mode 3 will be copied
                                <1>       ;     to 98000h address as backup, before mode change.
                                <1>
                                <1> _sm_save_pm:
                                <1>       ; 03/07/2016
                                <1>       ; save video pages
00001583 BE00800B00             <1>       mov   esi, 0B8000h
00001588 BF00800900             <1>       mov   edi, 98000h ; 30/07/2016
0000158D B900200000             <1>       mov   ecx, (0B8000h-0B0000h)/4
00001592 F3A5                   <1>       rep   movsd
                                <1>
00001594 C605[995F0100]03       <1>       mov   byte [p_crt_mode], 3 ; previous mode, backup sign
                                <1>       ;mov  cl, [ACTIVE_PAGE]
                                <1>       ;mov  [p_crt_page], cl
                                <1>
                                <1>       ; save cursor positions
0000159B BE[1E520100]           <1>       mov   esi, CURSOR_POSN
000015A0 BF[9E5F0100]           <1>       mov   edi, cursor_pposn    ; cursor positions backup
000015A5 B104                   <1>       mov   cl, 4
000015A7 F3A5                   <1>       rep   movsd
                                <1>
                                <1>       ; 29/07/2016
                                <1>       ;mov  [ACTIVE_PAGE], cl ; 0
000015A9 860D[2E520100]         <1>       xchg  cl, [ACTIVE_PAGE]
000015AF 880D[9A5F0100]         <1>       mov   [p_crt_page], cl     ; previous page (for mode 3)
                                <1>       ; [ACTIVE_PAGE] = 0
000015B5 EB19                   <1>       jmp   short _sm_2
                                <1>
                                <1> _sm_1:
000015B7 3C03                   <1>       cmp   al, 3        ; text, 80*25 color, default mode
                                <1>                          ; for TRDOS 386 MainProg
000015B9 7515                   <1>       jne   short _sm_2  ;  multiscreen is only for mode 3
                                <1>
                                <1>       ; If '_set_mode' procedure is called for video mode 3
                                <1>       ;     while video mode is not 3 and if there is video
                                <1>       ;     page backup for video mode 3, all (of 8) mode 3
                                <1>       ;     video pages will be restored from 98000h.
                                <1>
000015BB 803D[995F0100]03       <1>       cmp   byte [p_crt_mode], 3 ; previous mode, backup sign
000015C2 750C                   <1>       jne   short _sm_2 ; there is no (multiscreen) video pages
                                <1>                         ; to be restored
000015C4 8A0D[9A5F0100]         <1>       mov   cl, [p_crt_page]
000015CA 880D[2E520100]         <1>       mov   [ACTIVE_PAGE], cl
                                <1>
                                <1> _sm_2:
000015D0 A2[C25E0000]           <1>       mov   [CRT_MODE], al  ; save mode in global variable
                                <1> _sm_3:
                                <1>       ; 30/07/2016
                                <1>       ; 26/07/2016
                                <1>       ; 25/07/2016
                                <1>       ; set_mode_vga:
                                <1>       ; 18/07/2016
                                <1>       ; 14/07/2016
                                <1>       ; 09/07/2016
                                <1>       ; 04/07/2016
                                <1>       ; 03/07/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>       ; /// video mode 13h ///
                                <1>       ; derived from 'Plex86/Bochs VGABios' source code
                                <1>       ; vgabios-0.7a (2011)
                                <1>       ; by the LGPL VGABios developers Team (2001-2008)
                                <1>       ; 'vgabios.c', 'vgatables.h'
                                <1>       ;
                                <1>       ; Oracle VirtualBox 5.0.24 VGABios Source Code
                                <1>       ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
                                <1>       ;
000015D5 88C4                   <1>       mov   ah, al
000015D7 B910000000             <1>       mov   ecx, vga_mode_count
000015DC BE[DE5E0000]           <1>       mov   esi, vga_modes
000015E1 31DB                   <1>       xor   ebx, ebx
                                <1> _sm_4:
000015E3 AC                     <1>       lodsb
000015E4 38C4                   <1>       cmp   ah, al
000015E6 740C                   <1>       je    short _sm_5
000015E8 FEC3                   <1>       inc   bl
000015EA E2F7                   <1>       loop  _sm_4
                                <1>
```

```
                                <1>       ; UNIMPLEMENTED VIDEO MODE !
000015EC 31C0                   <1>       xor    eax, eax
000015EE A3[8C5F0100]           <1>         mov     [video_eax], eax ; 0
000015F3 C3                     <1>       retn
                                <1>
                                <1> ;-----      eBX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
                                <1>
                                <1> _sm_5:      ; 25/07/2016
000015F4 89DE                   <1>       mov    esi, ebx
000015F6 81C6[2E5F0000]         <1>       add    esi, vga_memmodel
000015FC 8A06                   <1>       mov    al, [esi]
000015FE A2[B25F0100]           <1>       mov    [VGA_MTYPE], al
                                <1>
00001603 89DF                   <1>       mov    edi, ebx
00001605 81C7[3E5F0000]         <1>       add    edi, vga_dac_s
0000160B C0E302                 <1>       shl    bl, 2 ; byte -> dword
0000160E 81C3[EE5E0000]         <1>       add    ebx, vga_mode_tbl_ptr
                                <1>
                                <1>       ;mov   dword [VGA_BASE], 0B8000h
                                <1>       ;cmp   ah, 0Dh ; [CRT_MODE]
                                <1>       ;jb    short M9
                                <1>       ;mov   dword [VGA_BASE], 0A0000h
                                <1> ;M9:
00001614 8B33                   <1>       mov    esi, [ebx]
00001616 89F3                   <1>       mov    ebx, esi
00001618 83C614                 <1>       add    esi, vga_p_cm_pos ; ebx + 20
0000161B 668B06                 <1>       mov    ax, [esi]       ; get the cursor mode from the table
0000161E 66A3[DB5E0000]         <1>       mov    [CURSOR_MODE], ax ; save cursor mode (initial value)
                                <1>       ; al = 6, ah = 7
                                <1>       ; al = 0Dh, ah = 0Eh ; 25/07/2016
00001624 E83B020000             <1>       call   cursor_shape_fix
                                <1>       ; al = 14, ah = 15  (If [CHAR_HEIGHT] = 16)
00001629 668906                 <1>       mov    [esi], ax
                                <1>
0000162C 56                     <1>       push   esi ; *
                                <1>
0000162D 8A25[C95E0000]         <1>       mov    ah, [VGA_MODESET_CTL]
00001633 80E408                 <1>       and    ah, 8 ; default palette loading ?
00001636 7524                   <1>       jnz    short _sm_6
00001638 66BAC603               <1>       mov    dx, 3C6h ; VGAREG_PEL_MASK (DAC mask register)
0000163C B0FF                   <1>       mov    al, 0FFh ; PEL mask
0000163E EE                     <1>       out    dx, al
0000163F 8A27                   <1>       mov    ah, [edi] ; DAC model (selection number)
00001641 E8ED0F0000             <1>       call   load_dac_palette
                                <1>       ; ecx = 0
00001646 F605[C95E0000]02       <1>       test   byte [VGA_MODESET_CTL], 2 ; gray scale summing
0000164D 740D                   <1>       jz     short _sm_6
0000164F 53                     <1>       push   ebx
00001650 29DB                   <1>       sub    ebx, ebx ; sub bl, bl
00001652 66B90001               <1>         mov     cx, 256
00001656 E82B100000             <1>       call   gray_scale_summing
0000165B 5B                     <1>       pop    ebx
                                <1> _sm_6:
                                <1>       ; Reset Attribute Ctl flip-flop
0000165C 66BADA03               <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
00001660 EC                     <1>       in     al, dx
                                <1>       ; Set Attribute Ctl
00001661 89DE                   <1>       mov    esi, ebx ; addr of params tbl for selected mode
00001663 83C623                 <1>       add    esi, 35  ; actl regs
00001666 30E4                   <1>       xor    ah, ah ; 0
00001668 66BAC003               <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
                                <1> _sm_7:
0000166C 88E0                   <1>       mov    al, ah
0000166E EE                     <1>       out    dx, al ; index
0000166F AC                     <1>       lodsb
                                <1>       ; DX = 3C0h = VGAREG_ACTL_WRITE_DATA
00001670 EE                     <1>       out    dx, al ; value
00001671 FEC4                   <1>       inc    ah
00001673 80FC14                 <1>       cmp    ah, 20 ; number of actl registers
00001676 72F4                   <1>       jb     short _sm_7
                                <1>       ;
00001678 88E0                   <1>       mov    al, ah ; 20
0000167A EE                     <1>       out    dx, al ; index
0000167B 28C0                   <1>       sub    al, al ; 0
0000167D EE                     <1>       out    dx, al ; value
                                <1>       ;
                                <1>       ; Set Sequencer Ctl
0000167E 89DE                   <1>       mov    esi, ebx ; addr of params tbl for selected mode
00001680 83C605                 <1>       add    esi, 5 ; sequ regs
                                <1>       ;
00001683 66BAC403               <1>       mov    dx, 3C4h  ; VGAREG_SEQU_ADDRESS
00001687 EE                     <1>       out    dx, al ; 0
00001688 6642                   <1>       inc    dx ; 3C5h ; VGAREG_SEQU_DATA
0000168A B003                   <1>       mov    al, 3
0000168C EE                     <1>       out    dx, al
0000168D B401                   <1>       mov    ah, 1
                                <1> _sm_8:
0000168F 88E0                   <1>       mov    al, ah
                                <1>       ;mov   dx, 3C4h ; VGAREG_SEQU_ADDRESS
00001691 664A                   <1>       dec    dx
00001693 EE                     <1>       out    dx, al ; index
00001694 AC                     <1>       lodsb
00001695 6642                   <1>       inc    dx ; 3C5h ; VGAREG_SEQU_DATA
00001697 EE                     <1>       out    dx, al
00001698 80FC04                 <1>       cmp    ah, 4 ; number of sequ regs
0000169B 7304                   <1>       jnb    short _sm_9
0000169D FEC4                   <1>       inc    ah
0000169F EBEE                   <1>       jmp    short _sm_8
                                <1> _sm_9:
                                <1>       ; Set Grafx Ctl
000016A1 89DE                   <1>       mov    esi, ebx ; addr of params tbl for selected mode
000016A3 83C637                 <1>       add    esi, 55 ; grdc regs
```

```
000016A6 30E4               <1>       xor    ah, ah ; 0
                            <1> _sm_10:
000016A8 88E0               <1>       mov    al, ah
000016AA 66BACE03           <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
000016AE EE                 <1>       out    dx, al
000016AF AC                 <1>       lodsb
000016B0 6642               <1>       inc    dx ; 3CFh ; VGAREG_GRDC_DATA
000016B2 EE                 <1>       out    dx, al
000016B3 FEC4               <1>       inc    ah
000016B5 80FC09             <1>       cmp    ah, 9 ; number of grdc regs
000016B8 72EE               <1>       jb     short _sm_10
                            <1>       ;
                            <1>       ; Disable CRTC write protection
000016BA 66BAD403           <1>       mov    dx, 3D4h  ; VGAREG_VGA_CRTC_ADDRESS
                            <1>       ;mov   al, 11h
                            <1>       ;our   dx, al
                            <1>       ;inc   dx
                            <1>       ;sub   al, al
                            <1>       ;out   dx, al
000016BE 66B81100           <1>       mov    ax, 11h
000016C2 66EF               <1>       out    dx, ax
000016C4 89DE               <1>       mov    esi, ebx ; addr of params tbl for selected mode
000016C6 83C60A             <1>       add    esi, 10 ; crtc regs
                            <1>       ; ah = 0
                            <1> _sm_11:
000016C9 88E0               <1>       mov    al, ah
                            <1>       ; dx = 3D4h = VGAREG_VGA_CRTC_ADDRESS
000016CB EE                 <1>       out    dx, al ; index
000016CC AC                 <1>       lodsb
000016CD 6642               <1>       inc    dx  ; VGAREG_VGA_CRTC_ADDRESS + 1
000016CF EE                 <1>       out    dx, al ; value
000016D0 80FC18             <1>       cmp    ah, 24 ; number of crtc registers - 1
000016D3 7306               <1>       jnb    short _sm_12
000016D5 FEC4               <1>       inc    ah
000016D7 664A               <1>       dec    dx ; 3D4h
000016D9 EBEE               <1>       jmp    short _sm_11
                            <1> _sm_12:
                            <1>       ; Set the misc register
000016DB 66BACC03           <1>       mov    dx, 3CCh ; VGAREG_READ_MISC_OUTPUT
000016DF 8A4309             <1>       mov    al, [ebx+9] ; misc reg
000016E2 EE                 <1>       out    dx, al
                            <1>       ;
                            <1>       ; Enable video
000016E3 66BAC003           <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
000016E7 B020               <1>       mov    al, 20h
000016E9 EE                 <1>         out     dx, al   ; set bit 5 to 1
000016EA 66BADA03           <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
000016EE EC                 <1>       in     al, dx
                            <1>       ;
000016EF 803D[9B5F0100]00   <1>       cmp    byte [noclearmem], 0
000016F6 7740               <1>         ja      short _sm_15
                            <1>
                            <1>       ; 29/07/2016
000016F8 31C0               <1>       xor    eax, eax
000016FA B900400000         <1>       mov    ecx, 4000h ; 16K words (32K)
000016FF 803D[B25F0100]02   <1>       cmp     byte [VGA_MTYPE], 2  ; CTEXT, MTEXT, CGA
00001706 7715               <1>       ja     short _sm_14    ; no ? (0A0000h)
00001708 BF00800B00         <1>       mov    edi, 0B8000h
0000170D 7409               <1>       je     short _sm_13 ; CGA graphics mode
                            <1>       ; 08/08/2016
0000170F A3[AE5F0100]       <1>       mov    [VGA_INT43H], eax ; 0 ; default font
00001714 66B82007           <1>       mov    ax, 0720h    ; CGA text mode
                            <1> _sm_13:
00001718 F366AB             <1>       rep    stosw
0000171B EB1B               <1>       jmp    short _sm_15
                            <1>
                            <1> _sm_14:
0000171D BF00000A00         <1>       mov    edi, 0A0000h
                            <1>       ; ecx = 16384 dwords  (64K)
00001722 66BAC403           <1>       mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
00001726 B002               <1>       mov    al, 2
00001728 EE                 <1>       out    dx, al
                            <1>       ;mov   dx, 3C5h ; VGAREG_SEQU_DATA
00001729 6642               <1>       inc    dx
0000172B EC                 <1>       in     al, dx ; mmask
0000172C 6650               <1>       push   ax
0000172E B00F               <1>       mov    al, 0Fh ; all planes
00001730 EE                 <1>       out    dx, al
00001731 30C0               <1>       xor    al, al ; 0
00001733 F3AB               <1>       rep    stosd  ; ecx = 163684 (64K)
00001735 6658               <1>       pop    ax
00001737 EE                 <1>       out    dx, al  ; mmask
                            <1> _sm_15:
                            <1>       ; ebx = addr of params tbl for selected mode
                            <1>       ; 10/08/2016
00001738 668B03             <1>       mov    ax, [ebx] ; num of columns, 'twidth'
0000173B A2[C45E0000]       <1>       mov    [CRT_COLS], al
                            <1>       ;; 26/07/2016
                            <1>       ;; CRTC_ADDRESS = 3D4h (always)
                            <1>       ;mov   ah, [ebx+1] ; num of rows, 'theightm1'
00001740 FEC4               <1>       inc    ah ; 09/07/2016
00001742 8825[CA5E0000]     <1>       mov    [VGA_ROWS], ah
                            <1>       ; 10/08/2016
00001748 8A4302             <1>       mov    al, [ebx+2]
0000174B A2[C65E0000]       <1>       mov    [CHAR_HEIGHT], al
                            <1>       ; 29/07/2016
                            <1>       ; length of regen buffer in bytes
00001750 668B4B03           <1>       mov    cx, [ebx+3] ; 'slength_l'
00001754 66890D[9C5F0100]   <1>       mov    [CRT_LEN], cx
                            <1>       ;
                            <1>       ; 27/07/2016
0000175B 30E4               <1>       xor    ah, ah
```

```
0000175D A0[2E520100]        <1>       mov   al, [ACTIVE_PAGE] ; may be > 0 for mode 3
                             <1>       ;mul  word [CRT_LEN] ; 4096 for mode 3
00001762 66F7E1              <1>       mul   cx ; 29/07/2016
00001765 66A3[1C520100]      <1>       mov   [CRT_START], ax
                             <1>       ;
0000176B B060                <1>       mov   al, 60h
0000176D 803D[9B5F0100]00    <1>       cmp   byte [noclearmem], 0
00001774 7602                <1>       jna   short _sm_16
00001776 0480                <1>       add   al, 80h
                             <1> _sm_16:
00001778 A2[C75E0000]        <1>       mov   [VGA_VIDEO_CTL], al
0000177D C605[C85E0000]F9    <1>       mov   byte [VGA_SWITCHES], 0F9h
00001784 8025[C95E0000]7F    <1>       and   byte [VGA_MODESET_CTL], 7Fh
                             <1>
0000178B 5E                  <1>       pop   esi ; *
                             <1>
                             <1>       ; 26/07/2016
                             <1>       ; 07/07/2016
0000178C 668B0D[DB5E0000]    <1>       mov   cx, [CURSOR_MODE] ; restore cursor mode (initial value)
00001793 66870E              <1>       xchg  cx, [esi] ; cl = start line, ch = end line
                             <1>                       ; reset to initial value
00001796 86E9                <1>       xchg  ch, cl  ; ch = start line, cl = end line
00001798 66890D[DB5E0000]    <1>       mov   [CURSOR_MODE], cx ; save (fixed) cursor mode
                             <1>
                             <1>       ; 27/07/2016
0000179F 803D[B25F0100]02    <1>       cmp   byte [VGA_MTYPE], 2 ; CTEXT, MTEXT
000017A6 7317                <1>       jnb   short _sm_17
                             <1>
                             <1>       ; Set cursor shape
                             <1>       ;mov  cx, 0607h
                             <1>       ;call _set_ctype
                             <1>
000017A8 B40A                <1>       ; 29/07/2016
000017AA E8C4050000          <1>       mov   ah, 10 ; 6845 register for cursor set
                             <1>       call  m16    ; output cx register
                             <1>
000017AF 803D[C25E0000]03    <1>       ; 25/07/2016
000017B6 7507                <1>         cmp     byte [CRT_MODE], 03h
                             <1>       jne   short _sm_17
                             <1>       ; 26/07/2016
                             <1>
000017B8 A0[2E520100]        <1>       mov   al, [ACTIVE_PAGE]
000017BD EB0C                <1>       jmp   short _sm_18
                             <1> _sm_17:
                             <1>       ; Set cursor pos for page 0..7
000017BF 6629C0              <1>       sub   ax, ax ; eax = 0
000017C2 BF[1E520100]        <1>       mov   edi, CURSOR_POSN
000017C7 AB                  <1>       stosd
000017C8 AB                  <1>       stosd
000017C9 AB                  <1>       stosd
000017CA AB                  <1>       stosd
                             <1>       ;; Set active page 0
                             <1>       ;mov  [ACTIVE_PAGE], al ; 0
                             <1> _sm_18:
                             <1>       ; 29/07/2016
000017CB 803D[B25F0100]02    <1>       cmp   byte [VGA_MTYPE], 2 ; CTEXT, MTEXT
000017D2 0F8386000000        <1>         jnb _sm_23
                             <1>
                             <1>       ;cmp  byte [CHAR_HEIGHT], 16
                             <1>       ;je   short _sm_19
                             <1>
                             <1>       ;; copy and activate 8x16 font
                             <1>
                             <1>       ; 26/07/2016
000017D8 B004                <1>       mov   al, 04h
                             <1>       ;sub  bl, bl
                             <1>       ; AX = 1104H ; Load ROM 8x16 Character Set
                             <1>       ; (BL = font block to load (EGA: 0-3; VGA: 0-7))
000017DA E83A150000          <1>       call  load_text_8_16_pat
                             <1>
                             <1>       ; video_func_1103h:
                             <1>       ; biosfn_set_text_block_specifier:
                             <1>       ; BL = font block selector code
                             <1>       ; NOTE: TRDOS 386 only uses and sets font block 0
                             <1>       ; (It is as BL = 0 for TRDOS 386)
000017DF 66BAC403            <1>       mov   dx, 3C4h ; VGAREG_SEQU_ADDRESS
                             <1>       ;mov  ah, bl
000017E3 28E4                <1>       sub   ah, ah ; 0
000017E5 B003                <1>       mov   al, 03h
000017E7 66EF                <1>       out   dx, ax
                             <1> _sm_19:
                             <1>       ; 29/07/2016
                             <1>       ; 26/07/2016
                             <1>       ; 24/06/2016
                             <1>       ;mov  edi, 0B8000h
                             <1>       ;mov  cx, 4000h ; 16K words (32K)
                             <1>       ;
000017E9 30C0                <1>       xor   al, al
000017EB 3805[995F0100]      <1>         cmp     byte [p_crt_mode], al ; 0
000017F1 7707                <1>         ja      short _sm_20 ; 3h, 80h or 83h
                             <1>
                             <1>       ; 30/07/2016
                             <1>       ; 24/06/2016
                             <1>       ; TRDOS 386 (TRDOS v2) 'set mode' modification
                             <1>       ; (for multiscreen feature):
                             <1>       ; If '_set_mode' procedure is called for video mode 3
                             <1>       ;     while video mode is 3, video page will be cleared
                             <1>       ;     and cursor position of video page will be reset.
                             <1>       ; If '_set_mode' procedure is called for a video mode
                             <1>       ;     except video mode 3, while current video mode
                             <1>       ;     is 3; all video pages of mode 3 will be copied
                             <1>       ;     to 98000h address as backup, before mode change.
```

```
                                <1>       ; If '_set_mode' procedure is called for video mode 3
                                <1>       ;     while video mode is not 3 and if there is video
                                <1>       ;     page backup for video mode 3, all (of 8) mode 3
                                <1>       ;     video pages will be restored from 98000h.
                                <1>
000017F3 A2[2E520100]           <1>       mov   [ACTIVE_PAGE], al ; 0
                                <1>       ;mov   ax, 0720h
                                <1>       ;;mov  cx, 4000h ; 16K words (32K)
                                <1>       ;;mov  edi, 0B8000h
                                <1>       ;rep   stosw
                                <1>       ;sub   al, al
000017F8 EB64                   <1>       jmp   short _sm_23
                                <1> _sm_20:
                                <1>       ; Previous video mode is 3
                                <1>       ; New video mode is 3 while current video mode is not 3
                                <1>       ; (multi screen) video pages will be restored from 0B0000h
                                <1>
000017FA 0FB61D[2E520100]       <1>       movzx  ebx, byte [ACTIVE_PAGE]
00001801 D0E3                   <1>       shl    bl, 1 ; * 2
00001803 81C3[1E520100]         <1>       add    ebx, CURSOR_POSN
                                <1>
                                <1>       ; 29/07/2016
00001809 F605[995F0100]7F       <1>       test    byte [p_crt_mode], 7Fh ; 83h or 3h
00001810 7427                   <1>       jz     short _sm_21 ; do not restore video pages
                                <1>
                                <1>       ;; restore video pages
00001812 BE00800900             <1>       mov    esi, 98000h ; 30/07/2016
00001817 BF00800B00             <1>       mov    edi, 0B8000h
0000181C 66B90020               <1>       mov    cx, 2000h ; 8K dwords (32K)
00001820 F3A5                   <1>       rep    movsd
                                <1>
                                <1>       ; restore cursor positions
00001822 BE[9E5F0100]           <1>       mov    esi, cursor_pposn
00001827 BF[1E520100]           <1>       mov    edi, CURSOR_POSN
                                <1>       ;mov   ecx, 4 ; restore all cursor positions (16 bytes)
0000182C B104                   <1>       mov    cl, 4
0000182E F3A5                   <1>       rep    movsd
                                <1>
00001830 F605[995F0100]80       <1>         test    byte [p_crt_mode], 80h
00001837 7420                   <1>       jz     short _sm_22 ; do not clear current video pages
                                <1> _sm_21:
                                <1>       ; clear video page
00001839 668B0D[9C5F0100]       <1>       mov    cx, [CRT_LEN] ; 4096
00001840 66D1E9                 <1>       shr    cx, 1 ; 2072
00001843 66B82007               <1>       mov    ax, 0720h
00001847 BF00800B00             <1>       mov    edi, 0B8000h ; [crt_base]
0000184C 66033D[1C520100]       <1>       add    di, [CRT_START]
00001853 F366AB                 <1>       rep    stosw  ; FILL THE REGEN BUFFER WITH BLANKS
                                <1>       ;
00001856 66890B                 <1>       mov    [ebx], cx ; reset cursor position
                                <1> _sm_22:
00001859 A2[995F0100]           <1>       mov    [p_crt_mode], al ; 0
                                <1> _sm_23:
                                <1>       ; al = video page number
                                <1>       ; [CRT_LEN] = length of regen buffer in bytes
0000185E E81E010000             <1>       call   _set_active_page
                                <1>
                                <1> ;-----      NORMAL RETURN FROM ALL VIDEO RETURNS
00001863 C3                     <1>       retn
                                <1>
                                <1> cursor_shape_fix:
                                <1>       ; 07/07/2016
                                <1>       ; (Cursor start and cursor end line values -6,7-
                                <1>       ; will be fixed depending on character height)
                                <1>       ;
                                <1>       ; derived from 'Plex86/Bochs VGABios' source code
                                <1>       ; vgabios-0.7a (2011)
                                <1>       ; by the LGPL VGABios developers Team (2001-2008)
                                <1>       ; 'vgabios.c', ' biosfn_set_cursor_shape (CH,CL)'
                                <1>       ;
                                <1>       ; INPUT ->
                                <1>       ;     AL = cursor start line (=6)
                                <1>       ;     AH = cursor end line (=7)
                                <1>       ; OUTPUT ->
                                <1>       ;     AL = cursor start line (=14)
                                <1>       ;     AH = cursor end line (=15)
                                <1>       ;
                                <1>       ;; if((modeset_ctl&0x01)&&(cheight>8)&&(CL<8)&&(CH<0x20))
                                <1>
                                <1>       ;test  byte [VGA_MODESET_CTL], 1 ; VGA active
                                <1>       ;jz    short csf_3
00001864 803D[C65E0000]08       <1>       cmp    byte [CHAR_HEIGHT], 8
0000186B 7649                   <1>       jna    short csf_3
0000186D 80FC08                 <1>       cmp    ah, 8
00001870 7344                   <1>       jnb    short csf_3
00001872 3C20                   <1>       cmp    al, 20h
00001874 7340                   <1>       jnb    short csf_3
                                <1>       ;
00001876 6650                   <1>       push   ax
                                <1>       ; {
                                <1>       ; if(CL!=(CH+1))
00001878 FEC0                   <1>       inc    al
0000187A 38C4                   <1>       cmp    ah, al   ; ah != al + 1
0000187C 740F                   <1>         je      short csf_1
                                <1>       ; CH = ((CH+1) * cheight / 8) -1;
0000187E 8A25[C65E0000]         <1>       mov    ah, [CHAR_HEIGHT]
00001884 F6E4                   <1>       mul    ah
00001886 C0E803                 <1>       shr    al, 3 ; / 8
00001889 FEC8                   <1>       dec    al ; - 1
0000188B EB0E                   <1>       jmp    short csf_2
                                <1> csf_1:
                                <1>       ; }
```

```
                                <1>       ; else       ; ah = al + 1
                                <1>       ; {
0000188D FEC4                   <1>       inc    ah     ; ah = ah + 1
                                <1>       ; CH = ((CL+1) * cheight / 8) - 2;
0000188F A0[C65E0000]           <1>       mov    al, [CHAR_HEIGHT]
00001894 F6E4                   <1>       mul    ah
00001896 C0E803                 <1>       shr    al, 3 ; / 8
00001899 2C02                   <1>       sub    al, 2 ; - 2
                                <1>       ; al = 14 (if [CHAR_HEIGHT] = 16)
                                <1> csf_2:
0000189B 880424                 <1>       mov    [esp], al
0000189E 8A642401               <1>       mov    ah, [esp+1]
                                <1>       ; CL = ((CL+1) * cheight / 8) - 1;
000018A2 FEC4                   <1>       inc    ah
000018A4 A0[C65E0000]           <1>       mov    al, [CHAR_HEIGHT]
000018A9 F6E4                   <1>       mul    ah
000018AB C0E803                 <1>       shr    al, 3 ; / 8
000018AE FEC8                   <1>       dec    al ; - 1
000018B0 88442401               <1>       mov    [esp+1], al
                                <1>       ; ah = 15 (if [CHAR_HEIGHT] = 16)
                                <1>       ;
000018B4 6658                   <1>       pop    ax
                                <1> csf_3:
000018B6 C3                     <1>       retn
                                <1>
                                <1> SET_CTYPE:
                                <1>       ; 12/09/2016
                                <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
000018B7 803D[C25E0000]07       <1>       cmp    byte [CRT_MODE], 7
000018BE 0F8790FCFFFF           <1>       ja     VIDEO_RETURN ; 12/09/2016
000018C4 E805000000             <1>       call   _set_ctype
000018C9 E986FCFFFF             <1>         jmp     VIDEO_RETURN
                                <1>
                                <1> _set_ctype:
                                <1>       ; 02/09/2014 (Retro UNIX 386 v1)
                                <1>       ;
                                <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                                <1>
                                <1>       ; (CH) = BITS 4-0 = START LINE FOR CURSOR
                                <1>       ;  ** HARDWARE WILL ALWAYS CAUSE BLINK
                                <1>       ;  ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING
                                <1>       ;     OR NO CURSOR AT ALL
                                <1>       ; (CL) = BITS 4-0 = END LINE FOR CURSOR
                                <1>
                                <1> ;--------------------------------------------
                                <1> ; SET_CTYPE
                                <1> ;     THIS ROUTINE SETS THE CURSOR VALUE
                                <1> ; INPUT
                                <1> ;     (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
                                <1> ; OUTPUT
                                <1> ;     NONE
                                <1> ;--------------------------------------------
                                <1>
                                <1>       ; 07/07/2016
                                <1>       ; Fixing cursor start and stop line depending on
                                <1>       ; current character height (=16)
                                <1>       ; (Note: Default/initial values are 6 and 7.
                                <1>       ; If set values are 6 (start) & 7 (stop) and
                                <1>       ; [CHAR_HEIGHT] = 16 :
                                <1>       ; After fixing, start line will be 14, stop line
                                <1>       ; will be 15.)
000018CE 6689C8                 <1>       mov    ax, cx
000018D1 86C4                   <1>       xchg   al, ah
                                <1>       ; AL = start line, AH = stop line
000018D3 E88CFFFFFF             <1>       call   cursor_shape_fix
                                <1>       ; AL = start line (fixed), AH = stop line (fixed)
000018D8 6689C1                 <1>       mov    cx, ax
000018DB 86E9                   <1>       xchg   ch, cl
                                <1>       ; CH = start line (fixed), CL = stop line (fixed)
                                <1>       ;
000018DD B40A                   <1>       mov    ah, 10 ; 6845 register for cursor set
000018DF 66890D[DB5E0000]       <1>       mov    [CURSOR_MODE], cx ; save in data area
                                <1>       ;call  m16    ; output cx register
                                <1>       ;retn
000018E6 E988040000             <1>         jmp     m16
                                <1>
                                <1> SET_CPOS:
                                <1>       ; 12/09/2016
                                <1>       ; 07/07/2016
                                <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
000018EB 80FF07                 <1>       cmp    bh, 7 ; video page > 7 ; 07/07/2016
000018EE 0F8760FCFFFF           <1>       ja     VIDEO_RETURN
                                <1>       ;
000018F4 803D[C25E0000]07       <1>       cmp    byte [CRT_MODE], 7
000018FB 770A                   <1>       ja     short vga_set_cpos ; 12/09/2016
000018FD E846040000             <1>       call   _set_cpos
00001902 E94DFCFFFF             <1>         jmp     VIDEO_RETURN
                                <1>
                                <1> vga_set_cpos:
                                <1>       ; 12/09/2016
                                <1>       ; 09/07/2016
                                <1>       ; set cursor position
                                <1>       ; NOTE: Hardware cursor position will not be set
                                <1>       ;   in any VGA modes (>7)
                                <1>       ;   But, cursor position will be saved into
                                <1>       ;   [CURSOR_POSN].
                                <1>       ;   TRDOS 386 (TRDOS v2.0) uses only one page
                                <1>       ;   (page 0) for all graphics modes.
                                <1>
00001907 668915[1E520100]       <1>       mov    [CURSOR_POSN], dx ; save cursor pos for pg 0
                                <1>       ; 04/08/2016
                                <1>       ;mov   bh, [ACTIVE_PAGE] ; = 0
```

```
                                <1>       ;call  _set_cpos
0000190E E941FCFFFF             <1>       jmp     VIDEO_RETURN
                                <1>
                                <1> READ_CURSOR:
                                <1>       ; 12/09/2016
                                <1>       ; 07/07/2016
                                <1>       ; 12/05/2016
                                <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>       ;
                                <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                                <1>
                                <1> ;--------------------------------------------------
                                <1> ; READ_CURSOR
                                <1> ;     THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
                                <1> ;     845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
                                <1> ; INPUT
                                <1> ;     BH - PAGE OF CURSOR
                                <1> ; OUTPUT
                                <1> ;     DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION
                                <1> ;     CX - CURRENT CURSOR MODE
                                <1> ;--------------------------------------------------
                                <1>
                                <1>       ; BH = Video page number (0 to 7)
                                <1>
                                <1>       ; 07/07/2016
00001913 80FF07                 <1>       cmp    bh, 7 ; video page > 7 (invalid)
00001916 7606                   <1>       jna    short read_cursor_1
                                <1>       ; invalid video page (input)
00001918 31C9                   <1>       xor    ecx, ecx ; 0
0000191A 31D2                   <1>       xor    edx, edx ; 0
0000191C EB15                   <1>       jmp    short read_cursor_2
                                <1> read_cursor_1:
                                <1>       ; 12/09/2016
0000191E 803D[C25E0000]07       <1>       cmp    byte [CRT_MODE], 7 ; vga mode
00001925 7727                   <1>       ja     short vga_get_cpos
                                <1>       ;
00001927 E815000000             <1>       call   get_cpos
0000192C 0FB70D[DB5E0000]       <1>       movzx  ecx, word [CURSOR_MODE]
                                <1> read_cursor_2:
00001933 5D                     <1>       pop    ebp
00001934 5F                     <1>       pop    edi
00001935 5E                     <1>       pop    esi
00001936 5B                     <1>       pop    ebx
00001937 58                     <1>       pop    eax  ; DISCARD SAVED CX AND DX
00001938 58                     <1>       pop    eax
00001939 A1[8C5F0100]           <1>       mov    eax, [video_eax] ; 12/05/2016
                                <1>       ;;15/01/2017
                                <1>       ;;mov  byte [intflg], 0 ; 07/01/2017
0000193E 1F                     <1>       pop    ds
0000193F 07                     <1>       pop    es
00001940 CF                     <1>       iretd
                                <1>
                                <1> get_cpos:
                                <1>       ; 12/05/2016
                                <1>       ; 16/01/2016
                                <1>       ; BH = Video page number (0 to 7)
                                <1>       ;
00001941 D0E7                   <1>       shl    bh, 1 ; WORD OFFSET
00001943 0FB6F7                 <1>       movzx  esi, bh
00001946 0FB796[1E520100]       <1>       movzx  edx, word [esi+CURSOR_POSN]
0000194D C3                     <1>       retn
                                <1>
                                <1> vga_get_cpos:
                                <1>       ; 12/09/2016
                                <1>       ; get cursor position (vga)
0000194E 0FB715[1E520100]       <1>       movzx  edx, word [CURSOR_POSN] ; cursor pos for pg 0
00001955 31C9                   <1>       xor    ecx, ecx ; Cursor Mode = 0 (invalid)
00001957 EBDA                   <1>       jmp     short read_cursor_2
                                <1>
                                <1> ACT_DISP_PAGE:
                                <1>       ; 07/07/2016
                                <1>       ; 26/06/2016
                                <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>       ;
                                <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                                <1>       ;
                                <1> ;--------------------------------------------------
                                <1> ; ACT_DISP_PAGE
                                <1> ;     THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
                                <1> ;     THE FULL USE OF THE MEMORY SET ASIDE FOR THE VIDEO ATTACHMENT
                                <1> ; INPUT
                                <1> ;     AL HAS THE NEW ACTIVE DISPLAY PAGE
                                <1> ; OUTPUT
                                <1> ;     THE 6845 IS RESET TO DISPLAY THAT PAGE
                                <1> ;--------------------------------------------------
                                <1>       ; 07/07/2016
00001959 3C07                   <1>       cmp    al, 7  ; > 7 = invalid video page number
0000195B 0F87F3FBFFFF           <1>       ja     VIDEO_RETURN
00001961 803D[C25E0000]03       <1>         cmp     byte [CRT_MODE], 3
00001968 7408                   <1>       je     short adp_1
0000196A 20C0                   <1>       and    al, al
0000196C 0F85E2FBFFFF           <1>         jnz     VIDEO_RETURN
                                <1>       ;sub   al, al ; 0 ; force to page 0
                                <1> adp_1:
00001972 E805000000             <1>       call   set_active_page
00001977 E9D8FBFFFF             <1>         jmp     VIDEO_RETURN
                                <1>
                                <1> set_active_page:   ; tty_sw
                                <1>       ; 26/07/2016
                                <1>       ; 26/06/2016
                                <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>       ; 30/06/2015
```

```
                                <1>       ; 04/03/2014  (act_disp_page --> tty_sw)
                                <1>       ; 10/12/2013
                                <1>       ; 04/12/2013
                                <1>       ;
0000197C A2[2E520100]           <1>       mov   [ACTIVE_PAGE], al ; save active page value ; [ptty]
                                <1> _set_active_page:
                                <1>       ; 27/06/2015
00001981 0FB6D8                 <1>       movzx  ebx, al
                                <1>       ;
00001984 6698                   <1>       cbw   ; 07/09/2014 (ah=0)
00001986 66F725[9C5F0100]       <1>       mul   word [CRT_LEN]  ; get saved length of regen buffer
                                <1>                             ; display page times regen length
                                <1>       ; 10/12/2013
0000198D 66A3[1C520100]         <1>       mov   [CRT_START], ax ; save start address for later
00001993 6689C1                 <1>       mov   cx, ax ; start address to cx
                                <1> _M16:
                                <1>       ;sar   cx, 1
00001996 66D1E9                 <1>       shr   cx, 1  ; divide by 2 for 6845 handling
00001999 B40C                   <1>       mov   ah, 12 ; 6845 register for start address
0000199B E8D3030000             <1>       call  m16
                                <1>       ;sal   bx, 1
                                <1>       ; 01/09/2014
000019A0 D0E3                   <1>       shl   bl, 1  ; *2 for word offset
000019A2 81C3[1E520100]         <1>       add   ebx, CURSOR_POSN
000019A8 668B13                 <1>       mov   dx, [ebx] ; get cursor for this page
                                <1>       ; 16/01/2016
                                <1>       ;call  m18
                                <1>       ;retn
000019AB E9AF030000             <1>       jmp   m18
                                <1>
                                <1> position:
                                <1>       ; 24/06/2016
                                <1>       ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; 27/06/2015
                                <1>       ; 02/09/2014
                                <1>       ; 30/08/2014 (Retro UNIX 386 v1)
                                <1>       ; 04/12/2013 (Retro UNIX 8086 v1)
                                <1>       ;
                                <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                                <1>       ;
                                <1> ;--------------------------------------
                                <1> ; POSITION
                                <1> ;     THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
                                <1> ;     OF A CHARACTER IN THE ALPHA MODE
                                <1> ; INPUT
                                <1> ;     AX = ROW, COLUMN POSITION
                                <1> ; OUTPUT
                                <1> ;     AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
                                <1> ;--------------------------------------
                                <1>
                                <1>       ; DX = ROW, COLUMN POSITION
000019B0 0FB605[C45E0000]       <1>       movzx  eax, byte [CRT_COLS] ; 24/06/2016
000019B7 F6E6                   <1>       mul   dh      ; row value
000019B9 30F6                   <1>       xor   dh, dh   ; 0
000019BB 6601D0                 <1>       add   ax, dx  ; add column value to the result
000019BE 66D1E0                 <1>       shl   ax, 1  ; * 2 for attribute bytes
                                <1>             ; EAX = AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
000019C1 C3                     <1>       retn
                                <1>
                                <1> find_position:
                                <1>       ; 24/06/2016
                                <1>       ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; 27/06/2015
                                <1>       ; 07/09/2014
                                <1>       ; 02/09/2014
                                <1>       ; 30/08/2014 (Retro UNIX 386 v1)
                                <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                                <1>
000019C2 0FB6CF                 <1>       movzx  ecx, bh ; video page number
000019C5 89CE                   <1>       mov   esi, ecx
000019C7 66D1E6                 <1>       shl   si, 1
000019CA 668B96[1E520100]       <1>       mov   dx, [esi+CURSOR_POSN]
000019D1 740C                   <1>       jz    short p21
000019D3 6631F6                 <1>       xor   si, si
                                <1> p20:
000019D6 660335[9C5F0100]       <1>       add   si, [CRT_LEN] ; 24/06/2016
                                <1>       ;add   si, 80*25*2 ; add length of buffer for one page
000019DD E2F7                   <1>       loop  p20
                                <1> p21:
000019DF 6621D2                 <1>       and   dx, dx
000019E2 7407                   <1>       jz    short p22
000019E4 E8C7FFFFFF             <1>       call  position ; determine location in regen in page
000019E9 01C6                   <1>       add   esi, eax ; add location to start of regen page
                                <1> p22:
                                <1>       ;mov   dx, [addr_6845] ; get base address of active display
                                <1>       ;mov   dx, 03D4h ; I/O address of color card
                                <1>       ;add   dx, 6  ; point at status port
000019EB 66BADA03               <1>       mov   dx, 03DAh ; status port
                                <1>       ; cx = 0
000019EF C3                     <1>       retn
                                <1>
                                <1> SCROLL_UP:
                                <1>       ; 07/07/2016
                                <1>       ; 26/06/2016
                                <1>       ; 12/05/2016
                                <1>       ; 30/01/2016
                                <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>       ; 07/09/2014
                                <1>       ; 02/09/2014
                                <1>       ; 01/09/2014 (Retro UNIX 386 v1 - beginning)
                                <1>       ; 04/04/2014
                                <1>       ; 04/12/2013
```

```
                        <1>       ;
                        <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                        <1>       ;
                        <1> ;--------------------------------------------
                        <1> ; SCROLL UP
                        <1> ;     THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
                        <1> ;     ON THE SCREEN
                        <1> ; INPUT
                        <1> ;     (AH) = CURRENT CRT MODE
                        <1> ;     (AL) = NUMBER OF ROWS TO SCROLL
                        <1> ;     (CX) = ROW/COLUMN OF UPPER LEFT CORNER
                        <1> ;     (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
                        <1> ;     (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
                        <1> ;     (DS) = DATA SEGMENT
                        <1> ;     (ES) = REGEN BUFFER SEGMENT
                        <1> ; OUTPUT
                        <1> ;     NONE -- THE REGEN BUFFER IS MODIFIED
                        <1> ;-----------------------------------------
                        <1>
                        <1>       ; 07/07/2016
000019F0 38F5           <1>       cmp   ch, dh
000019F2 0F875CFBFFFF   <1>       ja    VIDEO_RETURN
000019F8 38D1           <1>       cmp   cl, dl
000019FA 0F8754FBFFFF   <1>       ja    VIDEO_RETURN
                        <1>       ;
00001A00 E805000000     <1>       call  _scroll_up
00001A05 E94AFBFFFF     <1>         jmp     VIDEO_RETURN
                        <1>
                        <1> _scroll_up:  ; from 'write_tty'
                        <1>       ;
                        <1>       ; cl = left upper column
                        <1>       ; ch = left upper row
                        <1>       ; dl = right lower column
                        <1>       ; dh = right lower row
                        <1>       ;
                        <1>       ; al = line count
                        <1>       ; bl = attribute to be used on blanked line
                        <1>       ; bh = video page number (0 to 7)
                        <1>
00001A0A E896000000     <1>       call  test_line_count ; 16/01/2016
                        <1>
00001A0F 8A25[C25E0000] <1>       mov   ah, [CRT_MODE] ; current video mode
                        <1>       ;cmp  ah, 4
                        <1>       ;jb   short n0
                        <1>       ;cmp  byte [CRT_MODE], 4
00001A15 80FC04         <1>         cmp ah, 4 ; 07/07/2016
00001A18 0F8320050000   <1>       jnb     GRAPHICS_UP ; 26/06/2016
                        <1>
                        <1>       ;cmp  ah, 7 ; TEST FOR BW CARD
                        <1>       ;jne  GRAPHICS_UP
                        <1> n0:
                        <1>       ; 07/07/2016
00001A1E 80FF07         <1>       cmp   bh, 7 ; video page number
00001A21 7606           <1>       jna   short n1
00001A23 8A3D[2E520100] <1>       mov   bh, [ACTIVE_PAGE]
                        <1> n1:
00001A29 88DC           <1>       mov   ah, bl ; attribute
00001A2B 6650           <1>       push  ax ; *
                        <1>       ;mov  esi, [CRT_BASE]
00001A2D BE00800B00     <1>         mov     esi, 0B8000h
00001A32 3A3D[2E520100] <1>         cmp     bh, [ACTIVE_PAGE]
00001A38 750B           <1>       jne   short n2
                        <1>       ;
00001A3A 66A1[1C520100] <1>         mov     ax, [CRT_START]
00001A40 6601C6         <1>         add     si, ax
00001A43 EB11           <1>         jmp     short n4
                        <1> n2:
00001A45 20FF           <1>         and     bh, bh
00001A47 740D           <1>       jz    short n4
00001A49 88F8           <1>       mov   al, bh
                        <1> n3:
00001A4B 660335[9C5F0100] <1>       add  si, [CRT_LEN]
00001A52 FEC8           <1>         dec  al
00001A54 75F5           <1>       jnz   short n3
                        <1> n4:
00001A56 E85D000000     <1>       call  scroll_position ; 16/01/2016
00001A5B 7420           <1>         jz      short n6
                        <1>
00001A5D 01CE           <1>         add     esi, ecx ; from address for scroll
00001A5F 88F5           <1>       mov   ch, dh  ; #rows in block
00001A61 28C5           <1>       sub   ch, al ; #rows to be moved
                        <1> n5:
00001A63 E894000000     <1>       call  n10 ; 16/01/2016
                        <1>
00001A68 51             <1>         push ecx
00001A69 0FB60D[C45E0000] <1>       movzx ecx, byte [CRT_COLS]
00001A70 00C9           <1>       add   cl, cl
00001A72 01CE           <1>         add  esi, ecx  ; next line
00001A74 01CF           <1>         add  edi, ecx
00001A76 59             <1>       pop   ecx
                        <1>
00001A77 FECD           <1>       dec   ch      ; count of lines to move
00001A79 75E8           <1>       jnz   short n5 ; row loop
                        <1>       ; ch = 0
00001A7B 88C6           <1>       mov   dh, al  ; #rows
                        <1> n6:
                        <1>       ; attribute in ah
00001A7D B020           <1>       mov   al, ' '       ; fill with blanks
                        <1> n7:
00001A7F E885000000     <1>       call  n11 ; 16/01/2016
                        <1>
00001A84 8A0D[C45E0000] <1>       mov   cl, [CRT_COLS]
```

```
1292 00001A8A 00C9              <1>       add    cl, cl
1293 00001A8C 01CF              <1>         add  edi, ecx
1294                            <1>
1295 00001A8E FECE              <1>       dec    dh
1296 00001A90 75ED              <1>       jnz    short n7
1297                            <1> n16:
1298 00001A92 3A3D[2E520100]    <1>       cmp    bh, [ACTIVE_PAGE]
1299 00001A98 750A              <1>       jne    short n8
1300                            <1>
1301                            <1>       ;cmp   byte [CRT_MODE], 7  ; is this the black and white card
1302                            <1>       ;je    short n8            ; if so, skip the mode reset
1303                            <1>
1304 00001A9A A0[C35E0000]      <1>       mov    al, [CRT_MODE_SET] ; get the value of mode set
1305 00001A9F 66BAD803          <1>       mov    dx, 03D8h ; always set color card port
1306 00001AA3 EE                <1>       out    dx, al
1307                            <1> n8:
1308 00001AA4 C3                <1>       retn
1309                            <1>
1310                            <1> test_line_count:
1311                            <1>       ; 12/05/2016
1312                            <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
1313                            <1>       ; 07/09/2014 (scroll_up)
1314 00001AA5 08C0              <1>       or     al, al
1315 00001AA7 740E              <1>       jz     short al_set2
1316 00001AA9 6652              <1>       push   dx
1317 00001AAB 28EE              <1>       sub    dh, ch  ; subtract upper row from lower row number
1318 00001AAD FEC6              <1>       inc    dh     ; adjust difference by 1
1319 00001AAF 38C6              <1>       cmp    dh, al       ; line count = amount of rows in window?
1320 00001AB1 7502              <1>       jne    short al_set1 ; if not the we're all set
1321 00001AB3 30C0              <1>       xor    al, al ; otherwise set al to zero
1322                            <1> al_set1:
1323 00001AB5 665A              <1>       pop    dx
1324                            <1> al_set2:
1325 00001AB7 C3                <1>       retn
1326                            <1>
1327                            <1> scroll_position:
1328                            <1>       ; 26/06/2016
1329                            <1>       ; 30/01/2016
1330                            <1>         ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
1331                            <1>       ; 07/09/2014 (scroll_up)
1332                            <1>
1333 00001AB8 6652              <1>       push   dx
1334 00001ABA 6689CA            <1>       mov    dx, cx ; now, upper left position in DX
1335 00001ABD E8EEFEFFFF        <1>       call   position
1336 00001AC2 01C6              <1>       add    esi, eax
1337 00001AC4 89F7              <1>       mov    edi, esi
1338 00001AC6 665A              <1>       pop    dx     ; lower right position in DX
1339 00001AC8 6629CA            <1>       sub    dx, cx
1340 00001ACB FEC6              <1>       inc    dh     ; dh = #rows
1341 00001ACD FEC2              <1>       inc    dl     ; dl = #cols in block
1342 00001ACF 59                <1>       pop    ecx    ; return address
1343 00001AD0 6658              <1>       pop    ax     ; * ; al = line count, ah = attribute
1344 00001AD2 51                <1>       push   ecx    ; return address
1345 00001AD3 0FB7C8            <1>       movzx  ecx, ax
1346 00001AD6 8A25[C45E0000]    <1>       mov    ah, [CRT_COLS]
1347 00001ADC F6E4              <1>       mul    ah     ; determine offset to from address
1348 00001ADE 6601C0            <1>       add    ax, ax  ; *2 for attribute byte
1349                            <1>       ;
1350 00001AE1 6650              <1>       push   ax     ; offset
1351 00001AE3 6652              <1>       push   dx
1352                            <1>       ;
1353                            <1>       ; 04/04/2014
1354 00001AE5 66BADA03          <1>       mov    dx, 3DAh ; guaranteed to be color card here
1355                            <1> n9:                       ; wait_display_enable
1356 00001AE9 EC                <1>         in      al, dx   ; get port
1357 00001AEA A808              <1>       test   al, RVRT ; wait for vertical retrace
1358 00001AEC 74FB              <1>       jz     short n9 ; wait_display_enable
1359 00001AEE B025              <1>       mov    al, 25h
1360 00001AF0 B2D8              <1>       mov    dl, 0D8h ; address control port
1361 00001AF2 EE                <1>       out    dx, al ; turn off video during vertical retrace
1362 00001AF3 665A              <1>       pop    dx     ; #rows, #cols
1363 00001AF5 6658              <1>              pop    ax     ; offset
1364 00001AF7 6691              <1>       xchg   ax, cx ;
1365                            <1>       ; ecx = offset, al = line count, ah = attribute
1366                            <1>       ;
1367 00001AF9 08C0              <1>       or     al, al
1368 00001AFB C3                <1>       retn
1369                            <1> n10:
1370                            <1>       ; Move rows
1371 00001AFC 88D1              <1>       mov    cl, dl ; get # of cols to move
1372 00001AFE 56                <1>       push   esi
1373 00001AFF 57                <1>       push   edi    ; save start address
1374                            <1> n10r:
1375 00001B00 66A5              <1>       movsw         ; move that line on screen
1376 00001B02 FEC9              <1>       dec    cl
1377 00001B04 75FA              <1>         jnz     short n10r
1378 00001B06 5F                <1>       pop    edi
1379 00001B07 5E                <1>       pop    esi    ; recover addresses
1380 00001B08 C3                <1>       retn
1381                            <1> n11:
1382                            <1>       ; Clear rows
1383                            <1>                      ; dh =  #rows
1384 00001B09 88D1              <1>         mov  cl, dl ; get # of cols to clear
1385 00001B0B 57                <1>         push    edi     ; save address
1386                            <1> n11r:
1387 00001B0C 66AB              <1>         stosw           ; store fill character
1388 00001B0E FEC9              <1>       dec    cl
1389 00001B10 75FA              <1>         jnz     short n11r
1390 00001B12 5F                <1>         pop     edi     ; recover address
1391 00001B13 C3                <1>       retn
1392                            <1>
1393                            <1> SCROLL_DOWN:
```

```
                        <1>       ; 07/07/2016
                        <1>       ; 27/06/2016
                        <1>       ; 26/06/2016
                        <1>       ; 12/05/2016
                        <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                        <1>       ;
                        <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                        <1>
                        <1> ;-----------------------------------------
                        <1> ; SCROLL DOWN
                        <1> ;     THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED
                        <1> ;     BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES
                        <1> ;     WITH A DEFINED CHARACTER
                        <1> ; INPUT
                        <1> ;     (AH) = CURRENT CRT MODE
                        <1> ;     (AL) = NUMBER OF LINES TO SCROLL
                        <1> ;     (CX) = UPPER LEFT CORNER OF RECION
                        <1> ;     (DX) = LOWER RIGHT CORNER OF REGION
                        <1> ;     (BH) = FILL CHARACTER
                        <1> ;     (DS) = DATA SEGMENT
                        <1> ;     (ES) = REGEN SEGMENT
                        <1> ; OUTPUT
                        <1> ;     NONE -- SCREEN IS SCROLLED
                        <1> ;-----------------------------------------
                        <1>
                        <1>       ; 07/07/2016
00001B14 38F5           <1>       cmp   ch, dh
00001B16 0F8738FAFFFF   <1>       ja    VIDEO_RETURN
00001B1C 38D1           <1>       cmp   cl, dl
00001B1E 0F8730FAFFFF   <1>       ja    VIDEO_RETURN
                        <1>       ;
00001B24 E805000000     <1>       call  _scroll_down
00001B29 E926FAFFFF     <1>         jmp     VIDEO_RETURN
                        <1>
                        <1> _scroll_down: ; 27/06/2016
                        <1>
                        <1>       ; cl = left upper column
                        <1>       ; ch = left upper row
                        <1>       ; dl = right lower column
                        <1>       ; dh = right lower row
                        <1>       ;
                        <1>       ; al = line count
                        <1>       ; bl = attribute to be used on blanked line
                        <1>       ; bh = video page number (0 to 7)
                        <1>
                        <1>       ; !!!!
00001B2E FD             <1>       std           ; DIRECTION FOR SCROLL DOWN
                        <1>       ; !!!!
00001B2F E871FFFFFF     <1>       call  test_line_count ; 16/01/2016
                        <1>
00001B34 8A25[C25E0000] <1>       mov   ah, [CRT_MODE] ; current video mode
                        <1>       ;cmp  ah, 4
                        <1>       ;jb   short _n0
                        <1>       ;cmp  byte [CRT_MODE], 4
00001B3A 80FC04         <1>         cmp  ah, 4 ; 07/07/2016
00001B3D 0F83DF070000   <1>       jnb     GRAPHICS_DOWN ; 26/06/2016
                        <1>
                        <1>       ;cmp  ah, 7 ; TEST FOR BW CARD
                        <1>       ;jne  GRAPHICS_DOWN
                        <1> _n0:
                        <1>       ; 07/07/2016
00001B43 80FF07         <1>       cmp   bh, 7 ; video page number
00001B46 7606           <1>       jna   short n12
00001B48 8A3D[2E520100] <1>       mov   bh, [ACTIVE_PAGE]
                        <1>       ;
                        <1> n12:              ; CONTINUE_DOWN
00001B4E 88DC           <1>       mov   ah, bl
00001B50 6650           <1>       push  ax    ; * ; save attribute in ah
00001B52 6689D0         <1>       mov   ax, dx ; LOWER RIGHT CORNER
00001B55 E85EFFFFFF     <1>       call  scroll_position     ; GET REGEN LOCATION
00001B5A 741F           <1>       jz    short n14
00001B5C 29CE           <1>       sub   esi, ecx  ; SI IS FROM ADDRESS
00001B5E 88F5           <1>       mov   ch, dh  ; #rows in block
00001B60 28C5           <1>       sub   ch, al ; #rows to be moved
                        <1> n13:
00001B62 E895FFFFFF     <1>       call  n10   ; MOVE ONE ROW
                        <1>
00001B67 51             <1>       push  ecx
00001B68 8A0D[C45E0000] <1>       mov   cl, [CRT_COLS]
00001B6E 00C9           <1>       add   cl, cl
00001B70 29CE           <1>         sub  esi, ecx  ; next line
00001B72 29CF           <1>         sub  edi, ecx
00001B74 59             <1>         pop  ecx
                        <1>
00001B75 FECD           <1>       dec   ch      ; count of lines to move
00001B77 75E9           <1>       jnz   short n13 ; row loop
                        <1>       ; ch = 0
00001B79 88C6           <1>       mov   dh, al  ; #rows
                        <1> n14:
                        <1>       ; attribute in ah
00001B7B B020           <1>       mov   al, ' '       ; fill with blanks
                        <1> n15:
00001B7D E887FFFFFF     <1>       call  n11 ; 16/01/2016
                        <1>
00001B82 8A0D[C45E0000] <1>       mov   cl, [CRT_COLS]
00001B88 00C9           <1>       add   cl, cl
00001B8A 29CF           <1>         sub  edi, ecx
                        <1>
00001B8C FECE           <1>       dec   dh
00001B8E 75ED           <1>       jnz   short n15
                        <1>       ;
00001B90 E9FDFEFFFF     <1>       jmp   n16 ; 27/06/2016
```

```
1496                                    <1>
1497                                    <1> ;       cmp     bh, [ACTIVE_PAGE]
1498                                    <1> ;       jne     short n16
1499                                    <1> ;
1500                                    <1> ;       ;cmp    byte [CRT_MODE], 7  ; is this the black and white card
1501                                    <1> ;       ;je     short n16           ; if so, skip the mode reset
1502                                    <1> ;
1503                                    <1> ;       mov     al, [CRT_MODE_SET] ; get the value of mode set
1504                                    <1> ;       mov     dx, 03D8h ; always set color card port
1505                                    <1> ;       out     dx, al
1506                                    <1> ;n16:
1507                                    <1> ;       ; !!!!
1508                                    <1> ;       cld             ; Clear direction flag !
1509                                    <1> ;       ; !!!!
1510                                    <1> ;       retn
1511                                    <1>
1512                                    <1> READ_AC_CURRENT:
1513                                    <1>         ; 08/07/2016
1514                                    <1>         ; 26/06/2016
1515                                    <1>         ; 12/05/2016
1516                                    <1>         ; 18/01/2016
1517                                    <1>         ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
1518                                    <1>         ;
1519                                    <1>         ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
1520                                    <1>         ;
1521                                    <1>         ; 08/07/2016
1522 00001B95 803D[C25E0000]07         <1>           cmp     byte [CRT_MODE], 7 ; 6!?
1523 00001B9C 7607                     <1>         jna     short read_ac_c
1524 00001B9E 31C0                     <1>         xor     eax, eax
1525 00001BA0 E9B4F9FFFF               <1>           jmp     _video_return
1526                                    <1> read_ac_c:
1527 00001BA5 E805000000               <1>         call    _read_ac_current
1528                                    <1>         ; 12/05/2016
1529                                    <1>           ;jmp     VIDEO_RETURN
1530 00001BAA E9AAF9FFFF               <1>         jmp     _video_return
1531                                    <1>
1532                                    <1> ;--------------------------------------------------------------------------
1533                                    <1> ; READ_AC_CURRENT                                   :
1534                                    <1> ;       THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE CURRENT    :
1535                                    <1> ;       CURSOR POSITION AND RETURNS THEM TO THE CALLER                   :
1536                                    <1> ; INPUT                                                 :
1537                                    <1> ;       (AH) = CURRENT CRT MODE                         :
1538                                    <1> ;       (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )        :
1539                                    <1> ;       (DS) = DATA SEGMENT                         :
1540                                    <1> ;       (ES) = REGEN SEGMENT                            :
1541                                    <1> ; OUTPUT                                            :
1542                                    <1> ;       (AL) = CHARACTER READ                           :
1543                                    <1> ;       (AH) = ATTRIBUTE READ                           :
1544                                    <1> ;--------------------------------------------------------------------------
1545                                    <1>
1546                                    <1> _read_ac_current:
1547                                    <1>         ; 26/06/2016
1548                                    <1>         ; 12/05/2016
1549                                    <1>         ; 18/01/2016
1550                                    <1>
1551                                    <1>         ;mov   ah, [CRT_MODE] ; current video mode
1552                                    <1>         ;cmp   ah, 4
1553                                    <1>         ;jb    short p10
1554 00001BAF 803D[C25E0000]04         <1>         cmp    byte [CRT_MODE], 4
1555 00001BB6 0F83BB080000             <1>         jnb      GRAPHICS_READ ; 26/06/2016
1556                                    <1>
1557                                    <1>         ;cmp   ah, 7 ; TEST FOR BW CARD
1558                                    <1>         ;jne   GRAPHICS_READ
1559                                    <1> p10:
1560 00001BBC E801FEFFFF               <1>         call   find_position; GET REGEN LOCATION AND PORT ADDRESS
1561                                    <1>         ;
1562                                    <1>         ; esi = regen location
1563                                    <1>         ; dx = status port
1564                                    <1>         ;
1565 00001BC1 8A25[C25E0000]           <1>         mov    ah, [CRT_MODE]
1566 00001BC7 80EC02                   <1>         sub    ah, 2
1567 00001BCA D0EC                     <1>         shr    ah, 1
1568 00001BCC 7515                     <1>         jnz    short p13
1569                                    <1>
1570                                    <1>         ; WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1571                                    <1> p11:
1572 00001BCE FB                       <1>         sti            ; enable interrupts first
1573 00001BCF 3A3D[2E520100]           <1>         cmp     bh, [ACTIVE_PAGE]
1574 00001BD5 750C                     <1>         jne    short p13
1575 00001BD7 FA                       <1>         cli            ; block interrupts for single loop
1576 00001BD8 EC                       <1>         in     al, dx ; get status from the adapter
1577 00001BD9 A801                     <1>         test   al, RHRZ ; is horizontal retrace low
1578 00001BDB 75F1                     <1>         jnz    short p11 ; wait until it is
1579                                    <1> p12:                ;  wait for either retrace high
1580 00001BDD EC                       <1>         in     al, dx ; get status again
1581 00001BDE A809                     <1>         test   al, RVRT+RHRZ ; is horizontal or vertical retrace high
1582 00001BE0 74FB                     <1>         jz     short p12 ; wait until either retrace active
1583 00001BE2 FB                       <1>         sti
1584                                    <1> p13:
1585 00001BE3 81C600800B00             <1>         add    esi, 0B8000h
1586 00001BE9 668B06                   <1>         mov    ax, [esi]
1587                                    <1>
1588 00001BEC C3                       <1>         retn   ; 18/01/2016
1589                                    <1>
1590                                    <1> WRITE_AC_CURRENT:
1591                                    <1>         ; 08/07/2016
1592                                    <1>         ; 26/06/2016
1593                                    <1>         ; 24/06/2016
1594                                    <1>         ; 12/05/2016
1595                                    <1>         ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
1596                                    <1>         ;
1597                                    <1>         ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
```

```
1598                            <1>       ;
1599                            <1> ;-----------------------------------------------------------
1600                            <1> ; WRITE_AC_CURRENT                                :
1601                            <1> ;     THTS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER           :
1602                            <1> ;     AT THE CURRENT CURSOR POSITION                      :
1603                            <1> ; INPUT                                                   :
1604                            <1> ;     (AH) = CURRENT CRT MODE                             :
1605                            <1> ;     (BH) = DISPLAY PAGE                           :
1606                            <1> ;     (CX) = COUNT OF CHARACTERS TO WRITE                 :
1607                            <1> ;     (AL) = CHAR TO WRITE                                :
1608                            <1> ;     (BL) = ATTRIBUTE OF CHAR TO WRITE                   :
1609                            <1> ;     (DS) = DATA SEGMENT                           :
1610                            <1> ;     (ES) = REGEN SEGMENT                                :
1611                            <1> ; OUTPUT                                          :
1612                            <1> ;     DISPLAY REGEN BUFFER UPDATED                        :
1613                            <1> ;-----------------------------------------------------------
1614                            <1>
1615                            <1>       ; 08/07/2016
1616 00001BED 803D[C25E0000]07 <1>       cmp    byte [CRT_MODE], 7 ; 6!?
1617 00001BF4 760A             <1>       jna    short write_ac_c
1618                            <1>
1619 00001BF6 E8F20A0000       <1>       call   vga_write_char_attr
1620 00001BFB E954F9FFFF       <1>       jmp     VIDEO_RETURN
1621                            <1>
1622                            <1> write_ac_c:
1623 00001C00 E834000000       <1>       call   _write_c_current
1624                            <1>
1625 00001C05 0FB6F7           <1>       movzx  esi, bh ; video page number (0 to 7)
1626 00001C08 889E[CB5E0000]   <1>       mov    [esi+chr_attrib], bl ; color/attribute
1627                            <1>
1628 00001C0E E941F9FFFF       <1>         jmp     VIDEO_RETURN
1629                            <1>
1630                            <1> WRITE_C_CURRENT:
1631                            <1>       ; 08/07/2016
1632                            <1>       ; 26/06/2016
1633                            <1>       ; 12/05/2016
1634                            <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
1635                            <1>       ;
1636                            <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
1637                            <1>       ;
1638                            <1> ;-----------------------------------------------------------
1639                            <1> ; WRITE_C_CURRENT                                 :
1640                            <1> ;     THIS ROUTINE WRITES THE CHARACTER AT                :
1641                            <1> ;     THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED    :
1642                            <1> ; INPUT                                                   :
1643                            <1> ;     (AH) = CURRENT CRT MODE                             :
1644                            <1> ;     (BH) = DISPLAY PAGE                           :
1645                            <1> ;     (CX) = COUNT OF CHARACTERS TO WRITE                 :
1646                            <1> ;     (AL) = CHAR TO WRITE                                :
1647                            <1> ;     (DS) = DATA SEGMENT                           :
1648                            <1> ;     (ES) = REGEN SEGMENT                                :
1649                            <1> ; OUTPUT                                          :
1650                            <1> ;     DISPLAY REGEN BUFFER UPDATED                        :
1651                            <1> ;-----------------------------------------------------------
1652                            <1>
1653                            <1>       ; 08/07/2016
1654 00001C13 803D[C25E0000]07 <1>       cmp    byte [CRT_MODE], 7 ; 6!?
1655 00001C1A 760A             <1>       jna    short write_c_c
1656                            <1>
1657 00001C1C E8CC0A0000       <1>       call   vga_write_char_only
1658 00001C21 E92EF9FFFF       <1>       jmp     VIDEO_RETURN
1659                            <1>
1660                            <1> write_c_c:
1661                            <1>       ;and   bh, 7 ; video page number (<= 7)
1662 00001C26 0FB6F7           <1>       movzx  esi, bh
1663 00001C29 8A9E[CB5E0000]   <1>       mov    bl, [esi+chr_attrib]
1664                            <1>
1665 00001C2F E805000000       <1>       call   _write_c_current
1666 00001C34 E91BF9FFFF       <1>         jmp     VIDEO_RETURN
1667                            <1>
1668                            <1> _write_c_current:  ; from 'write_tty'
1669                            <1>       ; 26/06/2016
1670                            <1>       ; 24/06/2016
1671                            <1>       ; 12/05/2016
1672                            <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
1673                            <1>       ; 30/08/2014 (Retro UNIX 386 v1)
1674                            <1>       ; 18/01/2014
1675                            <1>       ; 04/12/2013
1676                            <1>       ;
1677                            <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
1678                            <1>
1679                            <1>       ;mov   ah, [CRT_MODE] ; current video mode
1680                            <1>       ;cmp   ah, 4
1681                            <1>       ;jb    short p40
1682 00001C39 803D[C25E0000]04 <1>       cmp    byte [CRT_MODE], 4
1683 00001C40 0F8381070000     <1>         jnb     GRAPHICS_WRITE ; 26/06/2016
1684                            <1>
1685                            <1>       ;cmp   ah, 7 ; TEST FOR BW CARD
1686                            <1>       ;jne   GRAPHICS_WRITE
1687                            <1> p40:
1688                            <1>       ; al = character
1689                            <1>       ; bl = color/attribute
1690                            <1>       ; bh = video page
1691                            <1>       ; cx = count of characters to write
1692 00001C46 6652              <1>       push   dx
1693 00001C48 88DC             <1>       mov    ah, bl  ; color/attribute (12/05/2016)
1694 00001C4A 6650             <1>       push   ax     ; save character & attribute/color
1695 00001C4C 6651             <1>       push   cx
1696 00001C4E E86FFDFFFF       <1>       call   find_position  ; get regen location and port address
1697 00001C53 6659             <1>       pop    cx
1698                            <1>       ; esi = regen location
1699                            <1>       ; dx = status port
```

```
1700                                    <1>       ;
1701 00001C55 81C600800B00              <1>       add    esi, 0B8000h ; 30/08/2014 (crt_base)
1702                                    <1>       ;
1703 00001C5B 8A25[C25E0000]            <1>       mov    ah, [CRT_MODE]
1704 00001C61 80EC02                    <1>       sub    ah, 2
1705 00001C64 D0EC                      <1>       shr    ah, 1
1706 00001C66 7519                      <1>       jnz    short p44    ; 26/06/2016
1707                                    <1>
1708                                    <1>       ; WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1709                                    <1> p41:
1710 00001C68 FB                        <1>       sti          ; enable interrupts first
1711 00001C69 3A3D[2E520100]            <1>       cmp     bh, [ACTIVE_PAGE]
1712 00001C6F 7510                      <1>       jne    short p44
1713 00001C71 FA                        <1>       cli          ; block interrupts for single loop
1714 00001C72 EC                        <1>       in     al, dx ; get status from the adapter
1715 00001C73 A808                      <1>       test   al, RVRT ; check for vertical retrace first
1716 00001C75 7509                      <1>       jnz    short p43 ; Do fast write now if vertical retrace
1717 00001C77 A801                      <1>       test   al, RHRZ ; is horizontal retrace low
1718 00001C79 75ED                      <1>       jnz    short p41 ; wait until it is
1719                                    <1> p42:              ;  wait for either retrace high
1720 00001C7B EC                        <1>       in     al, dx ; get status again
1721 00001C7C A809                      <1>       test   al, RVRT+RHRZ ; is horizontal or vertical retrace high
1722 00001C7E 74FB                      <1>       jz     short p42 ; wait until either retrace active
1723                                    <1> p43:
1724 00001C80 FB                        <1>       sti
1725                                    <1> p44:
1726 00001C81 668B0424                  <1>       mov    ax, [esp] ; restore the character (al) & attribute (ah)
1727 00001C85 668906                    <1>       mov    [esi], ax
1728                                    <1>
1729 00001C88 6649                      <1>       dec    cx
1730 00001C8A 7404                      <1>       jz     short p45
1731                                    <1>
1732 00001C8C 46                        <1>       inc    esi
1733 00001C8D 46                        <1>       inc    esi
1734 00001C8E EBD8                      <1>       jmp    short p41
1735                                    <1> p45:
1736 00001C90 6658                      <1>       pop    ax
1737 00001C92 665A                      <1>       pop    dx
1738 00001C94 C3                        <1>       retn
1739                                    <1>
1740                                    <1> ; 09/07/2016
1741                                    <1> ; 26/06/2016
1742                                    <1> ; 24/06/2016
1743                                    <1> ; 12/05/2016
1744                                    <1> ; 18/01/2016
1745                                    <1> ; 16/01/2016 - TRDOS 386 (TRDOS v2.0)
1746                                    <1> ; 30/06/2015
1747                                    <1> ; 27/06/2015
1748                                    <1> ; 11/03/2015
1749                                    <1> ; 02/09/2014
1750                                    <1> ; 30/08/2014
1751                                    <1> ; VIDEO FUNCTIONS
1752                                    <1> ; (write_tty - Retro UNIX 8086 v1 - U9.ASM, 01/02/2014)
1753                                    <1>
1754                                    <1> WRITE_TTY:
1755                                    <1>       ; 09/07/2016
1756                                    <1>       ; 01/07/2016
1757                                    <1>       ; 26/06/2016
1758                                    <1>       ; 24/06/2016
1759                                    <1>       ; 13/05/2016
1760                                    <1>       ; 12/05/2016
1761                                    <1>       ; 30/01/2016
1762                                    <1>       ; 18/01/2016
1763                                    <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
1764                                    <1>       ; 13/08/2015
1765                                    <1>       ; 02/09/2014
1766                                    <1>       ; 30/08/2014 (Retro UNIX 386 v1 - beginning)
1767                                    <1>       ; 01/02/2014 (Retro UNIX 8086 v1 - last update)
1768                                    <1>       ; 03/12/2013 (Retro UNIX 8086 v1 - beginning)
1769                                    <1>       ; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)
1770                                    <1>       ;
1771                                    <1>       ; INPUT -> AL = Character to be written
1772                                    <1>       ;         BL = Color (Forecolor, Backcolor)
1773                                    <1>       ;         BH = Video Page (0 to 7)
1774                                    <1>
1775                                    <1>       ; 09/07/2016
1776 00001C95 803D[C25E0000]07          <1>         cmp     byte [CRT_MODE], 7
1777 00001C9C 760A                      <1>       jna    short write_tty_cga
1778                                    <1>
1779 00001C9E E8290D0000                <1>       call   vga_write_teletype
1780 00001CA3 E9ACF8FFFF                <1>       jmp    VIDEO_RETURN
1781                                    <1>
1782                                    <1> write_tty_cga:
1783                                    <1>       ; 13/05/2016
1784                                    <1>       ;call  _write_tty
1785                                    <1>       ; 01/07/2016
1786 00001CA8 E818000000                <1>       call   _write_tty_m3
1787 00001CAD E9A2F8FFFF                <1>       jmp    VIDEO_RETURN
1788                                    <1>
1789                                    <1> RVRT  equ    00001000b    ; VIDEO VERTICAL RETRACE BIT
1790                                    <1> RHRZ  equ    00000001b    ; VIDEO HORIZONTAL RETRACE BIT
1791                                    <1>
1792                                    <1> ; Derived from "WRITE_TTY" procedure of IBM "pc-at" rombios source code
1793                                    <1> ; (06/10/1985), 'video.asm', INT 10H, VIDEO_IO
1794                                    <1> ;
1795                                    <1> ; 06/10/85  VIDEO DISPLAY BIOS
1796                                    <1> ;
1797                                    <1> ;--- WRITE_TTY ----------------------------------------------------------------
1798                                    <1> ;                                                                        :
1799                                    <1> ;   THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE                :
1800                                    <1> ;   VIDEO CARDS. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT              :
1801                                    <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> ; 08/07/2016
                        <1> ; 26/06/2016
                        <1> ; 24/06/2016
                        <1> _write_tty:  ; 13/05/2016
00001CB2 FA             <1>       cli
                        <1>       ;
                        <1>       ; 01/09/2014
00001CB3 803D[C25E0000]03   <1>       cmp    byte [CRT_MODE], 3
00001CBA 7409           <1>       je     short _write_tty_m3
                        <1>       ;
                        <1> set_mode_3:
00001CBC 53             <1>       push   ebx
00001CBD 50             <1>       push   eax
00001CBE E8A2F8FFFF     <1>       call   _set_mode
00001CC3 58             <1>       pop    eax
00001CC4 5B             <1>       pop    ebx
                        <1>       ;
                        <1> _write_tty_m3: ; 24/06/2016 (m3 -> _write_tty_m3)
00001CC5 0FB6F7         <1>       movzx  esi, bh ; 12/05/2016
00001CC8 66D1E6         <1>       shl    si, 1
00001CCB 81C6[1E520100]     <1>       add    esi, CURSOR_POSN
00001CD1 668B16         <1>       mov    dx, [esi]
                        <1>       ;
                        <1>       ; dx now has the current cursor position
                        <1>       ;
00001CD4 3C0D           <1>       cmp    al, 0Dh      ; CR   ; is it carriage return or control character
00001CD6 7636           <1>       jbe    short u8
                        <1>       ;
                        <1>       ; write the char to the screen
                        <1> u0:
                        <1>       ; al = character
                        <1>       ; bl = attribute/color
                        <1>       ; bh = video page number (0 to 7)
                        <1>       ;
00001CD8 66B90100       <1>       mov    cx, 1  ; 24/06/2016
                        <1>       ; cx = count of characters to write
                        <1>       ;
00001CDC E858FFFFFF     <1>       call   _write_c_current ; 16/01/2015
                        <1>       ;
                        <1>       ; position the cursor for next char
00001CE1 FEC2           <1>       inc    dl           ; next column
00001CE3 3A15[C45E0000]     <1>       cmp    dl, [CRT_COLS]  ; test for column overflow
00001CE9 755D           <1>         jne     _set_cpos
00001CEB B200           <1>       mov    dl, 0        ; column = 0
                        <1> u10:                      ; (line feed found)
00001CED 80FE18         <1>       cmp    dh, 25-1     ; check for last row
00001CF0 7218           <1>       jb     short u6
                        <1>       ;
                        <1>       ; scroll required
                        <1> u1:
                        <1>       ; SET CURSOR POSITION (04/12/2013)
00001CF2 E851000000     <1>       call   _set_cpos
                        <1>       ;
                        <1>       ; determine value to fill with during scroll
                        <1> u2:
                        <1>       ; bh = video page number
                        <1>       ;
00001CF7 E8B3FEFFFF     <1>       call   _read_ac_current ; 18/01/2016
                        <1>       ;
                        <1>       ; al = character, ah = attribute
                        <1>       ; bh = video page number
                        <1> u3:
                        <1>       ;;mov  ax, 0601h    ; scroll one line
                        <1>       ;;sub  cx, cx       ; upper left corner
                        <1>       ;;mov  dh, 25-1     ; lower right row
                        <1>       ;;;mov dl, [CRT_COLS]
                        <1>       ;mov   dl, 80       ; lower right column
                        <1>       ;;dec  dl
                        <1>       ;;mov  dl, 79
                        <1>
                        <1>       ;;call scroll_up    ; 04/12/2013
                        <1>       ;;; 11/03/2015
                        <1>       ; 02/09/2014
                        <1>       ;;;mov cx, [crt_ulc] ; Upper left corner  (0000h)
                        <1>       ;;;mov dx, [crt_lrc] ; Lower right corner (184Fh)
                        <1>       ; 11/03/2015
00001CFC 6629C9         <1>       sub    cx, cx
00001CFF 66BA4F18       <1>       mov    dx, 184Fh ; dl = 79 (column), dh = 24 (row)
                        <1>       ;
00001D03 B001           <1>       mov    al, 1        ; scroll 1 line up
                        <1>              ; ah = attribute
                        <1>       ;mov   bl, al ; 12/05/2016
00001D05 E900FDFFFF     <1>       jmp    _scroll_up   ; 16/01/2016
                        <1> ;u4:
```

```
1904                                    <1>       ;;int  10h          ; video-call return
1905                                    <1>                           ; scroll up the screen
1906                                    <1>                           ; tty return
1907                                    <1> ;u5:
1908                                    <1>       ;retn               ; return to the caller
1909                                    <1>
1910                                    <1> u6:                       ; set-cursor-inc
1911 00001D0A FEC6                     <1>       inc    dh           ; next row
1912                                    <1>                           ; set cursor
1913                                    <1> ;u7:
1914                                    <1>       ;;mov  ah, 02h
1915                                    <1>       ;;jmp  short u4     ; establish the new cursor
1916                                    <1>       ;call  _set_cpos
1917                                    <1>       ;jmp   short u5
1918 00001D0C EB3A                     <1>       jmp     _set_cpos
1919                                    <1>
1920                                    <1>       ; check for control characters
1921                                    <1> u8:
1922 00001D0E 7436                     <1>       je     short u9
1923 00001D10 3C0A                     <1>       cmp    al, 0Ah            ; is it a line feed (0Ah)
1924 00001D12 74D9                     <1>       je     short u10
1925 00001D14 3C07                     <1>       cmp    al, 07h      ; is it a bell
1926 00001D16 747A                     <1>       je     short u11
1927 00001D18 3C08                     <1>       cmp    al, 08h            ; is it a backspace
1928                                    <1>       ;jne   short u0
1929 00001D1A 7422                     <1>       je     short bs     ; 12/12/2013
1930                                    <1>       ; 12/12/2013 (tab stop)
1931 00001D1C 3C09                     <1>       cmp    al, 09h            ; is it a tab stop
1932 00001D1E 75B8                     <1>       jne    short u0
1933 00001D20 88D0                     <1>       mov    al, dl
1934 00001D22 6698                     <1>       cbw
1935 00001D24 B108                     <1>       mov    cl, 8
1936 00001D26 F6F1                     <1>       div    cl
1937 00001D28 28E1                     <1>       sub    cl, ah
1938                                    <1> ts:
1939                                    <1>       ; 02/09/2014
1940                                    <1>       ; 01/09/2014
1941 00001D2A B020                     <1>       mov    al, 20h
1942                                    <1> tsloop:
1943 00001D2C 6651                     <1>       push   cx
1944 00001D2E 6650                     <1>       push   ax
1945                                    <1>       ;mov   bh, [ACTIVE_PAGE]
1946 00001D30 E890FFFFFF               <1>       call   _write_tty_m3 ; 24/06/2016 (m3 -> _write_tty_m3)
1947 00001D35 6658                     <1>       pop    ax  ; ah = attribute/color
1948 00001D37 6659                     <1>       pop    cx
1949 00001D39 FEC9                     <1>       dec    cl
1950 00001D3B 75EF                     <1>       jnz    short tsloop
1951 00001D3D C3                       <1>       retn
1952                                    <1> bs:
1953                                    <1>       ; back space found
1954                                    <1>
1955 00001D3E 08D2                     <1>       or     dl, dl              ; is it already at start of line
1956                                    <1>       ;je    short u7     ; set_cursor
1957 00001D40 7406                     <1>       jz     short _set_cpos
1958 00001D42 664A                     <1>       dec    dx                  ; no -- just move it back
1959                                    <1>       ;jmp   short u7
1960 00001D44 EB02                     <1>       jmp    short _set_cpos
1961                                    <1>
1962                                    <1>       ; carriage return found
1963                                    <1> u9:
1964 00001D46 B200                     <1>       mov    dl, 0        ; move to first column
1965                                    <1>       ;jmp   short u7
1966                                    <1>       ;jmp   short _set_cpos ; 30/01/2016
1967                                    <1>
1968                                    <1>       ; line feed found
1969                                    <1> ;u10:
1970                                    <1> ;     cmp    dh, 25-1     ; bottom of screen
1971                                    <1> ;     jne    short u6     ; no, just set the cursor
1972                                    <1> ;       jmp     u1              ; yes, scroll the screen
1973                                    <1>
1974                                    <1> _set_cpos:
1975                                    <1>       ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
1976                                    <1>       ; 27/06/2015
1977                                    <1>       ; 01/09/2014
1978                                    <1>       ; 30/08/2014 (Retro UNIX 386 v1)
1979                                    <1>       ;
1980                                    <1>       ; 04/12/2013 - 12/12/2013 (Retro UNIX 8086 v1)
1981                                    <1>       ;
1982                                    <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
1983                                    <1>       ;
1984                                    <1> ;--------------------------------------------
1985                                    <1> ; SET_CPOS
1986                                    <1> ;     THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
1987                                    <1> ;     NEW X-Y VALUES PASSED
1988                                    <1> ; INPUT
1989                                    <1> ;     DX - ROW,COLUMN OF NEW CURSOR
1990                                    <1> ;     BH - DISPLAY PAGE OF CURSOR
1991                                    <1> ; OUTPUT
1992                                    <1> ;     CURSOR ID SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
1993                                    <1> ;--------------------------------------------
1994                                    <1>       ;
1995 00001D48 BE[1E520100]             <1>       mov    esi, CURSOR_POSN
1996 00001D4D 0FB6C7                   <1>         movzx   eax, bh   ; BH = video page number
1997                                    <1> ;     or     al, al
1998                                    <1> ;     jz     short _set_cpos_0
1999 00001D50 D0E0                     <1>         shl     al, 1   ; word offset
2000 00001D52 01C6                     <1>         add     esi, eax
2001                                    <1> ;_set_cpos_0:
2002 00001D54 668916                   <1>       mov    [esi], dx ; save the pointer
2003 00001D57 383D[2E520100]           <1>       cmp    [ACTIVE_PAGE], bh
2004 00001D5D 7532                     <1>       jne    short m17
2005                                    <1>       ;call  m18    ; CURSOR SET
```

```
                                <1> ;m17:               ; SET_CPOS_RETURN
                                <1>         ; 01/09/2014
                                <1> ;       retn
                                <1>                     ; DX  = row/column
                                <1> m18:
00001D5F E84CFCFFFF             <1>         call    position ; determine location in regen buffer
00001D64 668B0D[1C520100]       <1>         mov     cx, [CRT_START]
00001D6B 6601C1                 <1>         add     cx, ax  ; add char position in regen buffer
                                <1>                         ; to the start address (offset) for this page
00001D6E 66D1E9                 <1>         shr     cx, 1  ; divide by 2 for char only count
00001D71 B40E                   <1>         mov     ah, 14 ; register number for cursor
                                <1>         ;call   m16    ; output value to the 6845
                                <1>         ;retn
                                <1>
                                <1>         ;-----  THIS ROUTINE OUTPUTS THE CX REGISTER
                                <1>         ;       TO THE 6845 REGISTERS NAMED IN (AH)
                                <1> m16:
00001D73 FA                     <1>         cli
                                <1>         ;mov    dx, [addr_6845] ; address register
00001D74 66BAD403               <1>         mov     dx, 03D4h ; I/O address of color card
00001D78 88E0                   <1>         mov     al, ah ; get value
00001D7A EE                     <1>         out     dx, al ; register set
00001D7B 6642                   <1>         inc     dx     ; data register
00001D7D EB00                   <1>         jmp     $+2    ; i/o delay
00001D7F 88E8                   <1>         mov     al, ch ; data
00001D81 EE                     <1>         out     dx, al
00001D82 664A                   <1>         dec     dx
00001D84 88E0                   <1>         mov     al, ah
00001D86 FEC0                   <1>         inc     al     ; point to other data register
00001D88 EE                     <1>         out     dx, al ; set for second register
00001D89 6642                   <1>         inc     dx
00001D8B EB00                   <1>         jmp     $+2    ; i/o delay
00001D8D 88C8                   <1>         mov     al, cl ; second data value
00001D8F EE                     <1>         out     dx, al
00001D90 FB                     <1>         sti
                                <1> m17:
00001D91 C3                     <1>         retn
                                <1>
                                <1> beeper:
                                <1>         ; 04/08/2016
                                <1>         ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>         ; 30/08/2014 (Retro UNIX 386 v1)
                                <1>         ; 18/01/2014
                                <1>         ; 03/12/2013
                                <1>         ; bell found
                                <1> u11:
00001D92 FB                     <1>         sti
00001D93 3A3D[2E520100]         <1>         cmp     bh, [ACTIVE_PAGE]
00001D99 7551                   <1>         jne     short u12     ; Do not sound the beep
                                <1>                               ; if it is not written on the active page
                                <1> beeper_gfx: ; 04/08/2016
00001D9B 66B93305               <1>         mov     cx, 1331      ; divisor for 896 hz tone
00001D9F B31F                   <1>         mov     bl, 31        ; set count for 31/64 second for beep
                                <1>         ;call   beep          ; sound the pod bell
                                <1>         ;jmp    short u5      ; tty_return
                                <1>         ;retn
                                <1>
                                <1> TIMER equ   040h                ; 8254 TIMER - BASE ADDRESS
                                <1> PORT_B      equ   061h          ; PORT B READ/WRITE DIAGNOSTIC REGISTER
                                <1> GATE2 equ   00000001b     ; TIMER 2 INPUT CATE CLOCK BIT
                                <1> SPK2  equ   00000010b     ; SPEAKER OUTPUT DATA ENABLE BIT
                                <1>
                                <1> beep:
                                <1>         ; 07/02/2015
                                <1>         ; 30/08/2014 (Retro UNIX 386 v1)
                                <1>         ; 18/01/2014
                                <1>         ; 03/12/2013
                                <1>         ;
                                <1>         ; TEST4.ASM - 06/10/85  POST AND BIOS UTILITY ROUTINES
                                <1>         ;
                                <1>         ; ROUTINE TO SOUND THE BEEPER USING TIMER 2 FOR TONE
                                <1>         ;
                                <1>         ; ENTRY:
                                <1>         ;    (BL) = DURATION COUNTER ( 1 FOR 1/64 SECOND )
                                <1>         ;    (CX) = FREQUENCY DIVISOR (1193180/FREQUENCY) (1331 FOR 886 HZ)
                                <1>         ; EXIT:                          :
                                <1>         ;    (AX),(BL),(CX) MODIFIED.
                                <1>
00001DA1 9C                     <1>         pushf  ; 18/01/2014 ; save interrupt status
00001DA2 FA                     <1>         cli                 ; block interrupts during update
00001DA3 B0B6                   <1>         mov    al, 10110110b; select timer 2, lsb, msb binary
00001DA5 E643                   <1>         out    TIMER+3, al  ; write timer mode register
00001DA7 EB00                   <1>         jmp    $+2          ; I/O delay
00001DA9 88C8                   <1>         mov    al, cl       ; divisor for hz (low)
00001DAB E642                   <1>         out    TIMER+2,AL   ; write timer 2 count - lsb
00001DAD EB00                   <1>         jmp    $+2          ; I/O delay
00001DAF 88E8                   <1>         mov    al, ch       ; divisor for hz (high)
00001DB1 E642                   <1>         out    TIMER+2, al  ; write timer 2 count - msb
00001DB3 E461                   <1>         in     al, PORT_B   ; get current setting of port
00001DB5 88C4                   <1>         mov    ah, al       ; save that setting
00001DB7 0C03                   <1>         or     al, GATE2+SPK2     ; gate timer 2 and turn speaker on
00001DB9 E661                   <1>         out    PORT_B, al   ; and restore interrupt status
                                <1>         ;popf  ; 18/01/2014
00001DBB FB                     <1>         sti
                                <1> g7:                         ; 1/64 second per count (bl)
00001DBC B90B040000             <1>         mov    ecx, 1035    ; delay count for 1/64 of a second
00001DC1 E827000000             <1>         call   waitf        ; go to beep delay 1/64 count
00001DC6 FECB                   <1>         dec    bl           ; (bl) length count expired?
00001DC8 75F2                   <1>         jnz    short g7     ; no - continue beeping speaker
                                <1>         ;
                                <1>         ;pushf              ; save interrupt status
00001DCA FA                     <1>         cli    ; 18/01/2014 ; block interrupts during update
```

```
2108 00001DCB E461              <1>       in     al, PORT_B   ; get current port value
2109                            <1>         ;or      al, not (GATE2+SPK2) ; isolate current speaker bits in case
2110 00001DCD 0CFC              <1>         or      al, ~(GATE2+SPK2)
2111 00001DCF 20C4              <1>         and  ah, al       ; someone turned them off during beep
2112 00001DD1 88E0              <1>       mov    al, ah       ; recover value of port
2113                            <1>         ;or      al, not (GATE2+SPK2) ; force speaker data off
2114 00001DD3 0CFC              <1>       or     al, ~(GATE2+SPK2) ; isolate current speaker bits in case
2115 00001DD5 E661              <1>       out    PORT_B, al   ; and stop speaker timer
2116                            <1>       ;popf               ; restore interrupt flag state
2117 00001DD7 FB                <1>       sti
2118 00001DD8 B90B040000        <1>       mov    ecx, 1035    ; force 1/64 second delay (short)
2119 00001DDD E80B000000        <1>       call   waitf        ; minimum delay between all beeps
2120                            <1>       ;pushf              ; save interrupt status
2121 00001DE2 FA                <1>       cli                 ; block interrupts during update
2122 00001DE3 E461              <1>       in     al, PORT_B   ; get current port value in case
2123 00001DE5 2403              <1>       and    al, GATE2+SPK2      ; someone turned them on
2124 00001DE7 08E0              <1>       or     al, ah       ; recover value of port_b
2125 00001DE9 E661              <1>       out    PORT_B, al   ; restore speaker status
2126 00001DEB 9D                <1>       popf                ; restore interrupt flag state
2127                            <1> u12:
2128 00001DEC C3                <1>       retn
2129                            <1>
2130                            <1> REFRESH_BIT equ    00010000b    ; REFRESH TEST BIT
2131                            <1>
2132                            <1> WAITF:
2133                            <1> waitf:
2134                            <1>       ; 30/08/2014 (Retro UNIX 386 v1)
2135                            <1>       ; 03/12/2013
2136                            <1>       ;
2137                            <1> ;     push ax                   ; save work register (ah)
2138                            <1> ;waitf1:
2139                            <1>                           ; use timer 1 output bits
2140                            <1> ;     in     al, PORT_B   ; read current counter output status
2141                            <1> ;     and    al, REFRESH_BIT     ; mask for refresh determine bit
2142                            <1> ;     cmp    al, ah       ; did it just change
2143                            <1> ;     je     short waitf1 ; wait for a change in output line
2144                            <1> ;     ;
2145                            <1> ;     mov    ah, al       ; save new lflag state
2146                            <1> ;     loop   waitf1       ; decrement half cycles till count end
2147                            <1> ;     ;
2148                            <1> ;     pop    ax           ; restore (ah)
2149                            <1> ;     retn                ; return (cx)=0
2150                            <1>
2151                            <1> ; 06/02/2015 (unix386.s <-- dsectrm2.s)
2152                            <1> ; 17/12/2014 (dsectrm2.s)
2153                            <1> ; WAITF
2154                            <1> ; /// IBM PC-XT Model 286 System BIOS Source Code - Test 4 - 06/10/85 ///
2155                            <1> ;
2156                            <1> ;---WAITF----------------------------------------------------------------------
2157                            <1> ;     FIXED TIME WAIT ROUTINE (HARDWARE CONTROLLED - NOT PROCESSOR)
2158                            <1> ; ENTRY:
2159                            <1> ;     (CX) = COUNT OF 15.085737 MICROSECOND INTERVALS TO WAIT
2160                            <1> ;            MEMORY REFRESH TIMER 1 OUTPUT USED AS REFERENCE
2161                            <1> ; EXIT:
2162                            <1> ;                    AFTER (CX) TIME COUNT (PLUS OR MINUS 16 MICROSECONDS)
2163                            <1> ;     (CX) = 0
2164                            <1> ;-----------------------------------------------------------------------------
2165                            <1>
2166                            <1> ; Refresh period: 30 micro seconds (15-80 us)
2167                            <1> ; (16/12/2014 - AWARDBIOS 1999 - ATORGS.ASM, WAIT_REFRESH)
2168                            <1>
2169                            <1> ;WAITF:                              ; DELAY FOR (CX)*15.085737 US
2170 00001DED 6650              <1>       PUSH   AX                 ; SAVE WORK REGISTER (AH)
2171                            <1>       ; 16/12/2014
2172                            <1>       ;shr   cx, 1              ; convert to count of 30 micro seconds
2173 00001DEF D1E9              <1>       shr    ecx, 1 ; 21/02/2015
2174                            <1> ;17/12/2014
2175                            <1> ;WAITF1:
2176                            <1> ;     IN     AL, PORT_B   ;061h  ; READ CURRENT COUNTER OUTPUT STATUS
2177                            <1> ;     AND    AL, REFRESH_BIT     ;00010000b ; MASK FOR REFRESH DETERMINE BIT
2178                            <1> ;     CMP    AL, AH              ; DID IT JUST CHANGE
2179                            <1> ;     JE     short WAITF1        ; WAIT FOR A CHANGE IN OUTPUT LINE
2180                            <1> ;     MOV    AH, AL              ; SAVE NEW FLAG STATE
2181                            <1> ;     LOOP   WAITF1              ; DECREMENT HALF CYCLES TILL COUNT END
2182                            <1>       ;
2183                            <1>       ; 17/12/2014
2184                            <1>       ;
2185                            <1>       ; Modification from 'WAIT_REFRESH' procedure of AWARD BIOS - 1999
2186                            <1>       ;
2187                            <1> ;WAIT_REFRESH:  Uses port 61, bit 4 to have CPU speed independent waiting.
2188                            <1> ;     INPUT:  CX = number of refresh periods to wait
2189                            <1> ;                  (refresh periods = 1 per 30 microseconds on most machines)
2190                            <1> WR_STATE_0:
2191 00001DF1 E461              <1>       IN     AL,PORT_B          ; IN AL,SYS1
2192 00001DF3 A810              <1>       TEST   AL,010H
2193 00001DF5 74FA              <1>       JZ     SHORT WR_STATE_0
2194                            <1> WR_STATE_1:
2195 00001DF7 E461              <1>       IN     AL,PORT_B          ; IN AL,SYS1
2196 00001DF9 A810              <1>       TEST   AL,010H
2197 00001DFB 75FA              <1>       JNZ    SHORT WR_STATE_1
2198 00001DFD E2F2              <1>         LOOP    WR_STATE_0
2199                            <1>       ;
2200 00001DFF 6658              <1>       POP    AX                 ; RESTORE (AH)
2201 00001E01 C3                <1>       RETn                      ; (CX) = 0
2202                            <1>
2203                            <1> ; 09/07/2016
2204                            <1> ; 01/07/2016
2205                            <1> ; 24/06/2016
2206                            <1> ; 23/06/2016 - TRDOS 386 (TRDOS v2.0)
2207                            <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
2208                            <1> ;-----------------------------------------------------------------------------
2209                            <1> ; WRITE_STRING                                                        :
```

```
2210                            <1> ;     THIS ROUTINE WRITES A STRING OF CHARACTERS TO THE CRT.                  :
2211                            <1> ; INPUT                                                                 :
2212                            <1> ;     (AL) = WRITE STRING COMMAND  0 - 3                                  :
2213                            <1> ;     (BH) = DISPLAY PAGE (ACTIVE PAGE)                                   :
2214                            <1> ;     (CX) = COUNT OF CHARACTERS TO WRITE, IF (CX) = 0 THEN RETURN        :
2215                            <1> ;     (DX) = CURSOR POSITION FOR START OF STRING WRITE                    :
2216                            <1> ;     (BL) = ATTRIBUTE OF CHARACTER TO WRITE IF (AL) = 0  OR     (AL) = 1  :
2217                            <1> ;     (eBP) = SOURCE STRING OFFSET                                        :
2218                            <1> ; OUTPUT                                                                :
2219                            <1> ;     NONE                                                              :
2220                            <1> ;-----------------------------------------------------------------------------
2221                            <1>
2222                            <1> ; AL = 00h: Assign all characters the attribute in BL; do not update cursor
2223                            <1> ; AL = 01h: Assign all characters the attribute in BL; update cursor
2224                            <1> ; AL = 02h: Use attributes in string; do not update cursor
2225                            <1> ; AL = 03h: Use attributes in string; update cursor
2226                            <1>
2227                            <1> WRITE_STRING:
2228                            <1>       ; 12/09/2016
2229                            <1>       ; 09/07/2016
2230                            <1>       ;cmp   byte [CRT_MODE], 7 ; 6?!
2231                            <1>       ;ja    VIDEO_RETURN       ; not a valid function for VGA modes
2232                            <1>       ;
2233 00001E02 A2[985F0100]     <1>       mov    [w_str_cmd], al          ; save (AL) command
2234 00001E07 3C04             <1>       CMP    AL, 4              ; TEST FOR INVALID WRITE STRING OPTION
2235 00001E09 0F8345F7FFFF     <1>       JNB    VIDEO_RETURN       ; IF OPTION INVALID THEN RETURN
2236                            <1>
2237                            <1>         ;JCXZ   VIDEO_RETURN           ; IF ZERO LENGTH STRING THEN RETURN
2238                            <1>
2239 00001E0F 67E35E           <1>         jcxz    P55                ; 01/07/2016
2240                            <1>
2241                            <1>
2242                            <1>       ; 01/07/2016
2243                            <1>       ;and   ecx, 0FFFFh
2244                            <1>       ; ECX = byte count
2245                            <1>       ;push  ecx
2246 00001E12 89EE             <1>       mov    esi, ebp ; user buffer
2247 00001E14 BF00000700       <1>       mov    edi, Cluster_Buffer  ; system buffer
2248 00001E19 E8B2CA0000       <1>       call   transfer_from_user_buffer
2249                            <1>       ;pop   ecx
2250 00001E1E 0F8230F7FFFF     <1>       jc     VIDEO_RETURN
2251                            <1>       ; ecx = transfer (byte) count = character count
2252 00001E24 BD00000700       <1>       mov    ebp, Cluster_Buffer
2253                            <1>       ; 12/09/2016
2254 00001E29 803D[C25E0000]07 <1>       cmp    byte [CRT_MODE], 7 ; 6?!
2255 00001E30 0F879F000000     <1>       ja     vga_write_string
2256                            <1>       ;
2257 00001E36 0FB6F7           <1>       movzx  esi, bh                  ; GET CURRENT CURSOR PAGE
2258 00001E39 66D1E6           <1>       SAL    SI,1                ; CONVERT TO PAGE OFFSET  (SI= PAGE)
2259                            <1>       ; *****
2260 00001E3C 66FFB6[1E520100] <1>       PUSH   word [eSI+CURSOR_POSN]    ; SAVE CURRENT CURSOR POSITION IN STACK
2261                            <1>
2262                            <1>       ;MOV   AX,0200H            ; SET NEW CURSOR POSITION
2263                            <1>       ;INT   10H
2264                            <1> P50next:
2265 00001E43 53               <1>       push   ebx ; ****
2266 00001E44 51               <1>       push   ecx ; ***
2267 00001E45 56               <1>       push   esi ; **
2268 00001E46 52               <1>       push   edx ; *
2269 00001E47 E8FCFEFFFF       <1>       call   _set_cpos
2270                            <1> P50:
2271 00001E4C 8A4500           <1>       MOV    AL, [eBP]           ; GET CHARACTER FROM INPUT STRING
2272 00001E4F 45               <1>       INC    eBP                 ; BUMP POINTER TO CHARACTER
2273                            <1>
2274                            <1> ;-----      TEST FOR SPECIAL CHARACTER'S
2275                            <1>
2276 00001E50 3C08             <1>       CMP    AL, 08H                  ; IS IT A BACKSPACE
2277 00001E52 740C             <1>       JE     short P51           ; BACK_SPACE
2278 00001E54 3C0D             <1>       CMP    AL, 0Dh ; CR        ; IS IT CARRIAGE RETURN
2279 00001E56 7408             <1>       JE     short P51           ; CAR_RET
2280 00001E58 3C0A             <1>       CMP    AL, 0Ah ; LF        ; IS IT A LINE FEED
2281 00001E5A 7404             <1>       JE     short P51           ; LINE_FEED
2282 00001E5C 3C07             <1>       CMP    AL, 07h                  ; IS IT A BELL
2283 00001E5E 7515             <1>       JNE    short P52           ; IF NOT THEN DO WRITE CHARACTER
2284                            <1> P51:
2285                            <1>       ;MOV   AH,0EH              ; TTY_CHARACTER_WRITE
2286                            <1>       ;INT   10H                 ; WRITE TTY CHARACTER TO THE CRT
2287                            <1>
2288 00001E60 E860FEFFFF       <1>       call   _write_tty_m3
2289                            <1>
2290 00001E65 5A               <1>       pop    edx ; *
2291 00001E66 5E               <1>       pop    esi ; **
2292                            <1>
2293 00001E67 668B96[1E520100] <1>       MOV    DX, [eSI+CURSOR_POSN]     ; GET CURRENT CURSOR POSITION
2294 00001E6E EB46             <1>       JMP    SHORT P54           ; SET CURSOR POSITION AND CONTINUE
2295                            <1> P55:
2296 00001E70 E9DFF6FFFF       <1>       JMP    VIDEO_RETURN
2297                            <1> P52:
2298 00001E75 66B90100         <1>       MOV    CX, 1               ; SET CHARACTER WRITE AMOUNT TO ONE
2299 00001E79 803D[985F0100]02 <1>       CMP    byte [w_str_cmd], 2 ; IS THE ATTRIBUTE IN THE STRING
2300 00001E80 7204             <1>       JB     short P53           ; IF NOT THEN SKIP
2301 00001E82 8A5D00           <1>       MOV    BL, [eBP]           ; ELSE GET NEW ATTRIBUTE
2302 00001E85 45               <1>       INC    eBP                 ; BUMP STRING POINTER
2303                            <1> P53:
2304                            <1>       ;MOV   AH,09H              ; GOT_CHARACTER
2305                            <1>       ;INT   10H                 ; WRITE CHARACTER TO THE CRT
2306                            <1>
2307 00001E86 E8AEFDFFFF       <1>       call   _write_c_current
2308                            <1>
2309 00001E8B 5A               <1>       pop    edx ; *
2310                            <1>
2311 00001E8C 0FB6F7           <1>       movzx  esi, bh ; video page number (0 to 7)
```

```
2312 00001E8F 889E[CB5E0000]      <1>       mov   [esi+chr_attrib], bl ; color/attribute
2313                              <1>
2314 00001E95 FEC2                <1>       INC   DL                   ; INCREMENT COLUMN COUNTER
2315 00001E97 3A15[C45E0000]      <1>       CMP   DL, [CRT_COLS]             ; IF COLS ARE WITHIN RANGE FOR THIS MODE
2316 00001E9D 7217                <1>       JB    short P54            ;    THEN GO TO COLUMNS SET
2317 00001E9F FEC6                <1>       INC   DH                   ; BUMP ROW COUNTER BY ONE
2318 00001EA1 28D2                <1>       SUB   DL, DL               ; SET COLUMN COUNTER TO ZERO
2319 00001EA3 80FE19              <1>       CMP   DH, 25               ; IF ROWS ARE LESS THAN 25 THEN
2320 00001EA6 720E                <1>       JB    short P54            ; GO TO ROWS_COLUMNS_SET
2321                              <1>
2322 00001EA8 66B80A0E            <1>       MOV   AX,0E0AH             ; ELSE SCROLL SCREEN
2323                              <1>       ;INT  10H                  ; RESET ROW COUNTER TO 24
2324                              <1>
2325 00001EAC E814FEFFFF          <1>       call  _write_tty_m3
2326                              <1>
2327 00001EB1 66BA0018            <1>       mov   dx, 1800h            ; Column = 0, Row = 24
2328 00001EB5 5E                  <1>       pop   esi ; **
2329                              <1> P54:
2330                              <1>                                  ; ROW_COLUMNS_SET
2331                              <1>       ;MOV  AX,0200H             ; SET NEW CURSOR POSITION COMMAND
2332                              <1>       ;INT  10H                  ; ESTABLISH NEW CURSOR POSITION
2333                              <1>
2334 00001EB6 59                  <1>       pop   ecx ; ***
2335 00001EB7 5B                  <1>       pop   ebx ; ****
2336                              <1>
2337                              <1>       ;LOOP P50                  ; DO IT ONCE MORE UNTIL (CX) = ZERO
2338 00001EB8 6649                <1>       dec   cx
2339 00001EBA 7587                <1>       jnz   short P50next
2340                              <1>
2341 00001EBC 665A                <1>       POP   DX  ; *****          ; RESTORE OLD CURSOR COORDINATES
2342                              <1>
2343 00001EBE F605[985F0100]01    <1>       test  byte [w_str_cmd], 1 ; IF CURSOR WAS NOT TO BE MOVED
2344 00001EC5 0F8589F6FFFF        <1>       JNZ   VIDEO_RETURN         ; THEN EXIT WITHOUT RESETTING OLD VALUE
2345                              <1>
2346                              <1>       ;MOV  AX,0200H             ; ELSE RESTORE OLD CURSOR POSITION
2347                              <1>       ;INT  10H
2348                              <1>                                  ; DONE - EXIT WRITE STRING
2349 00001ECB E878FEFFFF          <1>       call  _set_cpos
2350 00001ED0 E97FF6FFFF          <1>       JMP   VIDEO_RETURN         ; RETURN TO CALLER
2351                              <1>
2352                              <1> vga_write_string:
2353                              <1>       ; 12/09/2016 - TRDOS 386 (TRDOS v2.0)
2354                              <1>       ;
2355                              <1>       ; derived from 'Plex86/Bochs VGABios' source code
2356                              <1>       ; vgabios-0.7a (2011)
2357                              <1>       ; by the LGPL VGABios developers Team (2001-2008)
2358                              <1>       ; 'vgabios.c', ' biosfn_write_string'
2359                              <1>
2360                              <1>       ; INPUT                                                    :
2361                              <1>       ;     (AL) = WRITE STRING COMMAND  0 - 3                       :
2362                              <1>       ;     (BH) = DISPLAY PAGE (ACTIVE PAGE)                        :
2363                              <1>       ;     (CX) = COUNT OF CHARACTERS TO WRITE, IF (CX) = 0 THEN RETURN       :
2364                              <1>       ;     (DX) = CURSOR POSITION FOR START OF STRING WRITE          :
2365                              <1>       ;     (BL) = ATTRIBUTE OF CHARACTER TO WRITE IF (AL) = 0  OR     (AL) = 1  :
2366                              <1>       ;     (eBP) = SOURCE STRING OFFSET                             :
2367                              <1>       ; OUTPUT                                                   :
2368                              <1>       ;     NONE                                                 :
2369                              <1>       ;-------------------------------------------------------------------------;
2370                              <1>
2371                              <1>       ; AL = 00h: Assign all characters the attribute in BL; do not update cursor
2372                              <1>       ; AL = 01h: Assign all characters the attribute in BL; update cursor
2373                              <1>       ; AL = 02h: Use attributes in string; do not update cursor
2374                              <1>       ; AL = 03h: Use attributes in string; update cursor
2375                              <1>
2376                              <1>       ; biosfn_write_string(GET_AL(),GET_BH(),GET_BL(),CX,GET_DH(),GET_DL(),ES,BP);
2377                              <1>       ; static void biosfn_write_string (flag,page,attr,count,row,col,seg,offset)
2378                              <1>
2379                              <1>       ; // Read curs info for the page
2380                              <1>       ; biosfn_get_cursor_pos(page,&dummy,&oldcurs);
2381                              <1>       ; bh = video page = 0
2382                              <1>       ;movzx esi, word [CURSOR_POSN] ; current cursor position for video page 0
2383                              <1>
2384                              <1>       ; // if row=0xff special case : use current cursor position
2385                              <1>       ; if(row==0xff)
2386                              <1>       ;  {col=oldcurs&0x00ff;
2387                              <1>       ;   row=(oldcurs&0xff00)>>8;
2388                              <1>       ;  }
2389                              <1>
2390                              <1>       ;mov   al, [w_str_cmd]
2391                              <1>
2392 00001ED5 80FEFF              <1>       cmp   dh, 0FFh
2393 00001ED8 7407                <1>       je    short vga_wstr_1 ; user current cursor position
2394                              <1> vga_wstr_0:
2395                              <1>       ; set cursor position
2396 00001EDA 668915[1E520100]    <1>       mov   [CURSOR_POSN], dx ; save cursor pos for pg 0
2397                              <1> vga_wstr_1:
2398 00001EE1 66FF35[1E520100]    <1>       push  word [CURSOR_POSN] ; *
2399                              <1>
2400                              <1>       ; ebp = string offset in system buffer (user buffer was copied to)
2401                              <1>
2402                              <1>       ; while(count--!=0)
2403                              <1>       ;  {
2404                              <1>       ;   car=read_byte(seg,offset++);
2405                              <1>       ;   if((flag&0x02)!=0)
2406                              <1>       ;    attr=read_byte(seg,offset++);
2407                              <1>       ;    biosfn_write_teletype(car,page,attr,WITH_ATTR);
2408                              <1>       ;  }
2409                              <1>
2410                              <1>       ;push  eax ; **
2411                              <1>       ;test  al, 2
2412 00001EE8 F605[985F0100]02    <1>       test  byte [w_str_cmd], 2
2413 00001EEF 751D                <1>       jnz   short vga_wstr_3
```

```
2414 00001EF1 881D[2F520100]      <1>       mov   [ccolor], bl
2415                              <1> vga_wstr_2:
2416 00001EF7 51                  <1>       push  ecx
2417 00001EF8 8A4500              <1>       mov   al, [ebp]
2418 00001EFB E8CC0A0000          <1>       call  vga_write_teletype
2419 00001F00 59                  <1>       pop   ecx
2420 00001F01 6649                <1>       dec   cx
2421 00001F03 741E                <1>       jz    short vga_wstr_4
2422 00001F05 45                  <1>       inc   ebp
2423 00001F06 8A1D[2F520100]      <1>       mov   bl, [ccolor]
2424 00001F0C EBE9                <1>       jmp   short vga_wstr_2
2425                              <1> vga_wstr_3:
2426 00001F0E 51                  <1>       push  ecx
2427 00001F0F 8A4500              <1>       mov   al, [ebp]
2428 00001F12 45                  <1>       inc   ebp
2429 00001F13 8A5D00              <1>       mov   bl, [ebp]
2430 00001F16 E8B10A0000          <1>       call  vga_write_teletype
2431 00001F1B 59                  <1>       pop   ecx
2432 00001F1C 6649                <1>       dec   cx
2433 00001F1E 7403                <1>       jz    short vga_wstr_4
2434 00001F20 45                  <1>       inc   ebp
2435 00001F21 EBEB                <1>       jmp   short vga_wstr_3
2436                              <1> vga_wstr_4:
2437                              <1>       ; // Set back curs pos
2438                              <1>       ; if((flag&0x01)==0)
2439                              <1>       ;  biosfn_set_cursor_pos(page,oldcurs);
2440                              <1>       ; }
2441                              <1>       ;pop   eax ; **
2442 00001F23 665A                <1>       pop   dx ; word [CURSOR_POSN] ; *
2443                              <1>       ;test  al, 1
2444 00001F25 F605[985F0100]01    <1>       test  byte [w_str_cmd], 1
2445 00001F2C 0F8522F6FFFF        <1>       jnz   VIDEO_RETURN
2446 00001F32 668915[1E520100]    <1>       mov   [CURSOR_POSN], dx
2447 00001F39 E916F6FFFF          <1>       JMP   VIDEO_RETURN
2448                              <1>
2449                              <1> ; 07/07/2016
2450                              <1> ; 27/06/2016 - TRDOS 386 (TRDOS v2.0)
2451                              <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
2452                              <1> ;--------------------------------------------------------
2453                              <1> ;  SCROLL UP
2454                              <1> ;   THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
2455                              <1> ; ENTRY ---
2456                              <1> ;  CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
2457                              <1> ;  DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
2458                              <1> ;   BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
2459                              <1> ;  BH = FILL VALUE FOR BLANKED LINES
2460                              <1> ;  AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
2461                              <1> ;  DS = DATA SEGMENT
2462                              <1> ;  ES = REGEN SEGMENT
2463                              <1> ; EXIT --
2464                              <1> ;  NOTHING, THE SCREEN IS SCROLLED
2465                              <1> ;--------------------------------------------------------
2466                              <1>
2467                              <1>       ; cl = upper left column
2468                              <1>       ; ch = upper left row
2469                              <1>       ; dl = lower rigth column
2470                              <1>       ; dh = lower right row
2471                              <1>       ;
2472                              <1>       ; al = line count (AL=0 means blank entire fields)
2473                              <1>       ; bl = fill value for blanked lines
2474                              <1>       ; bh = unused
2475                              <1>
2476                              <1> GRAPHICS_UP:
2477                              <1>       ; 07/07/2016
2478                              <1>       ;AH = Current video mode, [CRT_MODE]
2479 00001F3E 80FC07              <1>       cmp   ah, 7
2480 00001F41 7766                <1>       ja    short vga_graphics_up
2481                              <1>       ;je   n0
2482                              <1>
2483 00001F43 88C7                <1>       MOV   bh, al              ; save line count in BH
2484 00001F45 6689C8              <1>       MOV   AX, CX              ; GET UPPER LEFT POSITION INTO AX REG
2485                              <1>
2486                              <1> ;-----      USE CHARACTER SUBROUTINE FOR POSITIONING
2487                              <1> ;-----      ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
2488                              <1>
2489 00001F48 E8D9050000          <1>       CALL  GRAPH_POSN
2490 00001F4D 0FB7F8              <1>       MOVzx eDI, AX                   ; SAVE RESULT AS DESTINATION ADDRESS
2491                              <1>
2492                              <1> ;-----      DETERMINE SIZE OF WINDOW
2493                              <1>
2494 00001F50 6629CA              <1>       SUB   DX, CX
2495 00001F53 6681C20101          <1>         ADD     DX, 101h              ; ADJUST VALUES
2496 00001F58 C0E602              <1>       SAL   DH, 2               ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
2497                              <1>                                 ; AND EVEN/ODD ROWS
2498                              <1> ;-----      DETERMINE CRT MODE
2499                              <1>
2500 00001F5B 803D[C25E0000]06    <1>       CMP   byte [CRT_MODE], 6  ; TEST FOR MEDIUM RES
2501 00001F62 7305                <1>         JNC     short _R7_              ; FIND_SOURCE
2502                              <1>
2503                              <1> ;-----      MEDIUM RES UP
2504 00001F64 D0E2                <1>       SAL   DL, 1               ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
2505 00001F66 66D1E7              <1>       SAL   DI, 1               ; OFFSET *2 SINCE 2 BYTES/CHAR
2506                              <1>
2507                              <1> ;-----      DETERMINE THE SOURCE ADDRESS IN THE BUFFER
2508                              <1> _R7_:                                   ; FIND_SOURCE
2509 00001F69 81C700800B00        <1>       add   edi, 0B8000h
2510 00001F6F C0E702              <1>       sal   bh, 2               ; multiply number of lines by 4
2511 00001F72 7431                <1>         JZ      short _R11              ; IF ZERO, THEN BLANK ENTIRE FIELD
2512 00001F74 B050                <1>       MOV   AL, 80              ; 80 BYTES/ROW
2513 00001F76 F6E7                <1>       mul   bh                  ; determine offset to source
2514 00001F78 0FB7F0              <1>       movzx esi, ax                   ; offset to source
2515 00001F7B 01FE                <1>       add   eSI, eDI            ; SET UP SOURCE
```

```
2516 00001F7D 88F4              <1>       MOV    AH, DH              ; NUMBER OF ROWS IN FIELD
2517 00001F7F 28FC              <1>       sub    ah, bh              ; determine number to move
2518                            <1>
2519                            <1> ;-----      LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
2520                            <1> _R8:                                   ; ROW_LOOP
2521 00001F81 E812040000        <1>         CALL    _R17                   ; MOVE ONE ROW
2522 00001F86 6681EEB01F        <1>       SUB    SI, 2000h-80        ; MOVE TO NEXT ROW
2523 00001F8B 6681EFB01F        <1>       SUB    DI, 2000h-80
2524 00001F90 FECC              <1>       DEC    AH                  ; NUMBER OF ROWS TO MOVE
2525 00001F92 75ED              <1>         JNZ     short _R8              ; CONTINUE TILL ALL MOVED
2526                            <1>
2527                            <1> ;-----      FILL IN THE VACATED LINE(S)
2528                            <1> _R9:                                   ; CLEAR ENTRY
2529 00001F94 88D8              <1>       mov    al, bl              ; attribute to fill with
2530                            <1> _R10_:
2531 00001F96 E819040000        <1>         CALL    _R18                   ; CLEAR THAT ROW
2532 00001F9B 6681EFB01F        <1>       SUB    DI, 2000h-80        ; POINT TO NEXT LINE
2533 00001FA0 FECF              <1>       dec    bh                  ; number of lines to fill
2534 00001FA2 75F2              <1>         JNZ     short _R10_            ; CLEAR LOOP
2535 00001FA4 C3                <1>       retn                       ; EVERYYHING DONE
2536                            <1>
2537                            <1> _R11:                                  ; BLANK_FIELD
2538 00001FA5 88F7              <1>       mov    bh, dh              ; set blank count to everything in field
2539 00001FA7 EBEB              <1>         JMP     short _R9              ; CLEAR THE FIELD
2540                            <1>
2541                            <1> vga_graphics_up:
2542                            <1>       ; 08/08/2016
2543                            <1>       ; 07/08/2016
2544                            <1>       ; 04/08/2016
2545                            <1>       ; 01/08/2016
2546                            <1>       ; 31/07/2016
2547                            <1>       ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
2548                            <1>       ;
2549                            <1>       ; derived from 'Plex86/Bochs VGABios' source code
2550                            <1>       ; vgabios-0.7a (2011)
2551                            <1>       ; by the LGPL VGABios developers Team (2001-2008)
2552                            <1>       ; 'vgabios.c', 'biosfn_scroll'
2553                            <1>       ;
2554                            <1>
2555                            <1>       ; cl = upper left column
2556                            <1>       ; ch = upper left row
2557                            <1>       ; dl = lower rigth column
2558                            <1>       ; dh = lower right row
2559                            <1>       ;
2560                            <1>       ; al = line count (AL=0 means blank entire fields)
2561                            <1>       ; bl = fill value for blanked lines
2562                            <1>       ; bh = unused
2563                            <1>       ;
2564                            <1>       ; ah = [CRT_MODE], current video mode
2565                            <1>
2566 00001FA9 88C7              <1>       mov    bh, al ; 31/07/2016
2567 00001FAB BE[E65E0000]      <1>       mov    esi, vga_g_modes
2568 00001FB0 89F7              <1>       mov    edi, esi
2569 00001FB2 83C708            <1>       add    edi, vga_g_mode_count
2570                            <1> vga_g_up_0:
2571 00001FB5 AC                <1>       lodsb
2572 00001FB6 38E0              <1>       cmp    al, ah ; [CRT_MODE]
2573 00001FB8 7405              <1>       je     short vga_g_up_1
2574 00001FBA 39FE              <1>       cmp    esi, edi
2575 00001FBC 72F7              <1>       jb     short vga_g_up_0
2576                            <1>       ;xor   bh, bh ; 31/07/2016)
2577 00001FBE C3                <1>       retn   ; nothing to do
2578                            <1> vga_g_up_1:
2579 00001FBF 88F8              <1>       mov    al, bh ; 31/07/2016
2580 00001FC1 83C64F            <1>       add    esi, vga_g_memmodel - (vga_g_modes + 1)
2581                            <1>       ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
2582                            <1>
2583                            <1>       ; if(rlr>=nbrows)rlr=nbrows-1;
2584                            <1>       ; if(clr>=nbcols)clr=nbcols-1;
2585                            <1>       ; if(nblines>nbrows)nblines=0;
2586                            <1>       ; cols=clr-cul+1;
2587                            <1>
2588 00001FC4 3A35[CA5E0000]      <1>       cmp    dh, [VGA_ROWS]
2589 00001FCA 7208              <1>       jb     short vga_g_up_2
2590 00001FCC 8A35[CA5E0000]    <1>       mov    dh, [VGA_ROWS]
2591 00001FD2 FECE              <1>       dec    dh
2592                            <1> vga_g_up_2:
2593 00001FD4 3A15[C45E0000]    <1>       cmp    dl, [CRT_COLS]  ; = [VGA_COLS]
2594 00001FDA 7208              <1>       jb     short vga_g_up_3
2595 00001FDC 8A15[C45E0000]    <1>       mov    dl, [CRT_COLS]
2596 00001FE2 FECA              <1>       dec    dl
2597                            <1> vga_g_up_3:
2598 00001FE4 3A05[CA5E0000]    <1>       cmp    al, [VGA_ROWS]
2599 00001FEA 7602              <1>       jna    short vga_g_up_4
2600 00001FEC 28C0              <1>       sub    al, al ; 0
2601                            <1> vga_g_up_4:
2602 00001FEE 88D7              <1>       mov    bh, dl ; clr
2603 00001FF0 28CF              <1>       sub    bh, cl ; cul
2604 00001FF2 FEC7              <1>       inc    bh ; cols = clr-cul+1
2605                            <1>
2606 00001FF4 20C0              <1>       and    al, al ; nblines = 0
2607 00001FF6 755D              <1>       jnz    short vga_g_up_6
2608 00001FF8 20ED              <1>       and    ch, ch ; rul = 0
2609 00001FFA 7559              <1>       jnz    short vga_g_up_6
2610 00001FFC 20C9              <1>       and    cl, cl ; cul = 0
2611 00001FFE 7555              <1>       jnz    short vga_g_up_6
2612                            <1>
2613 00002000 6650              <1>       push   ax
2614 00002002 A0[CA5E0000]      <1>       mov    al, [VGA_ROWS]
2615 00002007 FEC8              <1>       dec    al
2616 00002009 38C6              <1>       cmp    dh, al ; rlr = nbrows-1
2617 0000200B 7546              <1>       jne    short vga_g_up_5
```

```
2618 0000200D A0[C45E0000]        <1>         mov     al, [CRT_COLS]  ; = VGA_COLS
2619 00002012 FEC8                <1>       dec   al
2620 00002014 38C2                <1>       cmp   dl, al ; clr = nbcols-1
2621 00002016 753B                <1>       jne   short vga_g_up_5
2622 00002018 6658                <1>       pop   ax
2623                              <1>
2624 0000201A 66B80502            <1>       mov   ax, 0205h
2625 0000201E 66BACE03            <1>       mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
2626 00002022 66EF                <1>       out   dx, ax
2627 00002024 A0[CA5E0000]        <1>       mov   al, [VGA_ROWS]
2628 00002029 8A25[C45E0000]      <1>       mov   ah, [CRT_COLS] ; = [VGA_COLS]
2629 0000202F F6E4                <1>       mul   ah
2630 00002031 0FB7D0              <1>       movzx edx, ax
2631                              <1>       ; 08/08/2016
2632 00002034 0FB605[C65E0000]    <1>       movzx eax, byte [CHAR_HEIGHT]
2633 0000203B F7E2                <1>       mul   edx
2634                              <1>       ; eax = byte count
2635 0000203D 89C1                <1>       mov   ecx, eax
2636                              <1>       ;; 07/08/2016
2637                              <1>       ;shl   dx, 3 ; * 8 ; * [CHAR_HEIGHT]
2638                              <1>       ;mov   ecx, edx
2639 0000203F 88D8                <1>       mov   al, bl ; fill value for blanked lines
2640 00002041 BF00000A00          <1>       mov   edi, 0A0000h
2641 00002046 F3AA                <1>       rep   stosb
2642                              <1>
2643 00002048 66B80500            <1>       mov   ax, 5
2644 0000204C 66BACE03            <1>       mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
2645 00002050 66EF                <1>       out   dx, ax ; 0005h
2646                              <1>
2647 00002052 C3                  <1>       retn
2648                              <1>
2649                              <1> vga_g_up_5:
2650 00002053 6658                <1>       pop   ax
2651                              <1>
2652                              <1> vga_g_up_6:
2653                              <1>       ; [ESI] = VGA memory model number for current video mode
2654                              <1>         ;
2655                              <1>         ; LINEAR8 equ 5
2656                              <1>         ; PLANAR4 equ 4
2657                              <1>         ; PLANAR1 equ 3
2658                              <1>
2659 00002055 803E04              <1>       cmp   byte [esi], PLANAR4
2660 00002058 7424                <1>       je    short vga_g_up_planar
2661 0000205A 803E03              <1>       cmp   byte [esi], PLANAR1
2662 0000205D 741F                <1>       je    short vga_g_up_planar
2663                              <1> vga_g_up_linear8:
2664                              <1>       ; 07/07/2016 (TEMPORARY)
2665                              <1>       ;
2666                              <1>       ; cl = upper left column ; cul
2667                              <1>       ; ch = upper left row ; rul
2668                              <1>       ; dl = lower rigth column ; clr
2669                              <1>       ; dh = lower right row ; rlr
2670                              <1>
2671                              <1> vga_g_up_l0:
2672                              <1>       ;{for(i=rul;i<=rlr;i++)
2673                              <1>       ; if((i+nblines>rlr)||(nblines==0))
2674 0000205F 08C0                <1>       or    al, al
2675 00002061 7414                <1>       jz    short vga_g_up_l2
2676 00002063 88C4                <1>       mov   ah, al
2677 00002065 00EC                <1>       add   ah, ch ; i+nblines
2678                              <1>       ;jc   short vga_g_up_l2
2679 00002067 38F4                <1>       cmp   ah, dh
2680 00002069 770C                <1>       ja    short vga_g_up_l2
2681                              <1>       ; else
2682                              <1>       ;  vgamem_copy_pl4(cul,i+nblines,i,cols,nbcols,cheight);
2683 0000206B E8F2000000          <1>       call  vgamem_copy_l8
2684                              <1> vga_g_up_l1:
2685 00002070 FEC5                <1>       inc   ch
2686 00002072 38F5                <1>       cmp   ch, dh
2687 00002074 76E9                <1>       jna   short vga_g_up_l0
2688 00002076 C3                  <1>       retn
2689                              <1> vga_g_up_l2:
2690                              <1>       ; vgamem_fill_pl4(cul,i,cols,nbcols,cheight,attr);
2691 00002077 E850010000          <1>       call  vgamem_fill_l8
2692 0000207C EBF2                <1>       jmp   short vga_g_up_l1
2693                              <1>
2694                              <1> vga_g_up_planar:
2695                              <1>       ; cl = upper left column ; cul
2696                              <1>       ; ch = upper left row ; rul
2697                              <1>       ; dl = lower rigth column ; clr
2698                              <1>       ; dh = lower right row ; rlr
2699                              <1> vga_g_up_pl0:
2700                              <1>       ;{for(i=rul;i<=rlr;i++)
2701                              <1>       ; if((i+nblines>rlr)||(nblines==0))
2702 0000207E 20C0                <1>       and   al, al
2703 00002080 7414                <1>       jz    short vga_g_up_pl2
2704 00002082 88C4                <1>       mov   ah, al
2705 00002084 00EC                <1>       add   ah, ch ; i+nblines
2706                              <1>       ;jc   short vga_g_up_pl2
2707 00002086 38F4                <1>       cmp   ah, dh
2708 00002088 770C                <1>       ja    short vga_g_up_pl2
2709                              <1>       ; else
2710                              <1>       ;  vgamem_copy_pl4(cul,i+nblines,i,cols,nbcols,cheight);
2711 0000208A E80E000000          <1>       call  vgamem_copy_pl4
2712                              <1> vga_g_up_pl1:
2713 0000208F FEC5                <1>       inc   ch
2714 00002091 38F5                <1>       cmp   ch, dh
2715 00002093 76E9                <1>       jna   short vga_g_up_pl0
2716 00002095 C3                  <1>       retn
2717                              <1> vga_g_up_pl2:
2718                              <1>       ; vgamem_fill_pl4(cul,i,cols,nbcols,cheight,attr);
2719 00002096 E870000000          <1>       call  vgamem_fill_pl4
```

```
2720 0000209B EBF2              <1>       jmp    short vga_g_up_pl1
2721                            <1>
2722                            <1> vgamem_copy_pl4:
2723                            <1>       ; 08/08/2016
2724                            <1>       ; 07/08/2016
2725                            <1>       ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
2726                            <1>       ;
2727                            <1>       ; derived from 'Plex86/Bochs VGABios' source code
2728                            <1>       ; vgabios-0.7a (2011)
2729                            <1>       ; by the LGPL VGABios developers Team (2001-2008)
2730                            <1>       ; 'vgabios.c', 'vgamem_copy_pl4'
2731                            <1>       ;
2732                            <1>       ; vgamem_copy_pl4(xstart,ysrc,ydest,cols,nbcols,cheight)
2733                            <1>       ; cl = xstart, ah = ysrc (i+nblines), ch = ydest (i),
2734                            <1>       ; bh = cols, [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight
2735                            <1>
2736                            <1>       ; src=ysrc*cheight*nbcols+xstart;
2737                            <1>       ; dest=ydest*cheight*nbcols+xstart;
2738                            <1>
2739 0000209D 52                <1>       push   edx
2740 0000209E 50                <1>       push   eax
2741                            <1>
2742                            <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0105)
2743 0000209F 66B80501          <1>       mov    ax, 0105h
2744 000020A3 66BACE03          <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
2745 000020A7 66EF              <1>       out    dx, ax
2746                            <1>
2747                            <1>       ; 07/08/2016
2748                            <1>       ;mov     ah, [esp+1]
2749                            <1>       ;movzx edx, ah ; ysrc
2750 000020A9 0FB6542401        <1>       movzx  edx, byte [esp+1]
2751                            <1>       ; 08/08/2016
2752 000020AE 0FB605[C65E0000]  <1>       movzx  eax, byte [CHAR_HEIGHT]
2753 000020B5 8A25[C45E0000]    <1>       mov    ah, [CRT_COLS] ; nbcols
2754 000020BB F6E4              <1>       mul    ah
2755                            <1>       ;; 07/08/2016
2756                            <1>       ;movzx eax, byte [CRT_COLS]
2757                            <1>       ;shl   ax, 3 ; * 8 ; * [CHAR_HEIGHT]
2758 000020BD 50                <1>       push   eax ; cheight * nbcols
2759 000020BE F7E2              <1>       mul    edx ; * ysrc
2760                            <1>       ; eax = ysrc * cheight * nbcols
2761                            <1>       ; edx = 0
2762 000020C0 88CA              <1>       mov    dl, cl ; edx = xstart
2763 000020C2 01D0              <1>       add    eax, edx
2764 000020C4 89C6              <1>       mov    esi, eax ; src
2765 000020C6 88EA              <1>       mov    dl, ch ; ydest
2766 000020C8 58                <1>       pop    eax ; cheight * nbcols
2767 000020C9 F7E2              <1>       mul    edx
2768                            <1>       ; eax = ydest * cheight * nbcols
2769 000020CB 88CA              <1>       mov    dl, cl ; edx = xstart
2770 000020CD 01D0              <1>       add    eax, edx
2771 000020CF 89C7              <1>       mov    edi, eax ; dest
2772                            <1>       ; esi = src
2773                            <1>       ; edi = dest
2774                            <1>       ; for(i=0;i<cheight;i++)
2775                            <1>       ; {
2776                            <1>       ;  memcpyb(0xa000,dest+i*nbcols,0xa000,src+i*nbcols,cols);
2777                            <1>       ; }
2778 000020D1 51                <1>       push   ecx
2779 000020D2 B900000A00        <1>       mov    ecx, 0A0000h
2780 000020D7 01CE              <1>       add    esi, ecx
2781 000020D9 01CF              <1>       add    edi, ecx
2782                            <1>       ; 08/08/2016
2783 000020DB 8A35[C65E0000]    <1>       mov    dh, [CHAR_HEIGHT]
2784                            <1>       ;; 07/08/2016
2785                            <1>       ;mov   dh, 8 ; 07/08/2016
2786 000020E1 28D2              <1>       sub    dl, dl ; i
2787                            <1> vgamem_copy_pl4_0:
2788 000020E3 56                <1>       push   esi
2789 000020E4 57                <1>       push   edi
2790 000020E5 0FB605[C45E0000]  <1>       movzx  eax, byte [CRT_COLS]
2791 000020EC F6E2              <1>       mul    dl
2792                            <1>       ; eax = i * nbcols
2793 000020EE 01C7              <1>       add    edi, eax ; dest+i*nbcols
2794 000020F0 01C6              <1>       add    esi, eax
2795 000020F2 0FB6CF            <1>       movzx  ecx, bh ; cols
2796 000020F5 F3A4              <1>       rep    movsb
2797 000020F7 5F                <1>       pop    edi
2798 000020F8 5E                <1>       pop    esi
2799 000020F9 FECE              <1>       dec    dh
2800 000020FB 75E6              <1>       jnz    short vgamem_copy_pl4_0
2801                            <1> vgamem_copy_pl4_1:
2802 000020FD 59                <1>       pop    ecx
2803                            <1>
2804                            <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0005);
2805 000020FE 66B80500          <1>       mov    ax, 0005h
2806 00002102 66BACE03          <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
2807 00002106 66EF              <1>       out    dx, ax
2808                            <1>
2809 00002108 58                <1>       pop    eax
2810 00002109 5A                <1>       pop    edx
2811                            <1>
2812 0000210A C3                <1>       retn
2813                            <1>
2814                            <1> vgamem_fill_pl4:
2815                            <1>       ; 08/08/2016
2816                            <1>       ; 07/08/2016
2817                            <1>       ; 04/08/2016
2818                            <1>       ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
2819                            <1>       ;
2820                            <1>       ; derived from 'Plex86/Bochs VGABios' source code
2821                            <1>       ; vgabios-0.7a (2011)
```

```
2822                                 <1>       ; by the LGPL VGABios developers Team (2001-2008)
2823                                 <1>       ; 'vgabios.c', 'vgamem_fill_pl4'
2824                                 <1>       ;
2825                                 <1>       ; vgamem_fill_pl4(xstart,ystart,cols,nbcols,cheight,attr)
2826                                 <1>       ; cl = xstart, edi = ch = ystart, bh = cols,
2827                                 <1>       ; [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight, attr = 0
2828                                 <1>
2829                                 <1>       ; dest=ystart*cheight*nbcols+xstart;
2830 0000210B 52                    <1>       push   edx
2831 0000210C 50                    <1>       push   eax
2832                                 <1>
2833                                 <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0205)
2834 0000210D 66B80502              <1>       mov    ax, 0205h
2835 00002111 66BACE03              <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
2836 00002115 66EF                  <1>       out    dx, ax
2837                                 <1>
2838                                 <1>              ; 08/08/2016
2839 00002117 0FB605[C65E0000]      <1>       movzx   eax, byte [CHAR_HEIGHT]
2840 0000211E F6E5                  <1>       mul    ch
2841                                 <1>       ;; 07/08/2016
2842                                 <1>       ;movzx eax, ch
2843                                 <1>       ;shl   ax, 3 ; * 8 ; * [CHAR_HEIGHT]
2844 00002120 0FB615[C45E0000]      <1>       movzx  edx, byte [CRT_COLS] ; = [VGA_COLS]
2845 00002127 F7E2                  <1>       mul    edx
2846                                 <1>       ; edx  = 0
2847 00002129 88CA                  <1>       mov    dl, cl
2848 0000212B 01D0                  <1>       add    eax, edx
2849 0000212D 89C7                  <1>       mov    edi, eax
2850                                 <1>       ; edi = dest
2851                                 <1>       ; for(i=0;i<cheight;i++)
2852                                 <1>       ; {
2853                                 <1>       ;  memsetb(0xa000,dest+i*nbcols,attr,cols);
2854                                 <1>       ; }
2855 0000212F 81C700000A00            <1>       add    edi, 0A0000h
2856 00002135 51                    <1>       push   ecx
2857                                 <1>       ; 08/08/2016
2858 00002136 8A35[C65E0000]        <1>       mov    dh, [CHAR_HEIGHT]
2859                                 <1>       ;; 07/08/2016
2860                                 <1>       ;mov   dh, 8 ; 07/08/2016
2861 0000213C 28D2                  <1>       sub    dl, dl ; i
2862                                 <1> vgamem_fill_pl4_0:
2863 0000213E 57                    <1>       push   edi
2864 0000213F 0FB605[C45E0000]      <1>       movzx  eax, byte [CRT_COLS]
2865 00002146 F6E2                  <1>       mul    dl
2866                                 <1>       ; eax = i * nbcols
2867 00002148 01C7                  <1>       add    edi, eax ; dest+i*nbcols
2868 0000214A 88D8                  <1>       mov    al, bl ; attr ; 04/08/2016
2869 0000214C 0FB6CF                <1>       movzx  ecx, bh ; cols
2870 0000214F F3AA                  <1>       rep    stosb
2871 00002151 5F                    <1>       pop    edi
2872 00002152 75EA                  <1>       jnz    short vgamem_fill_pl4_0
2873                                 <1> vgamem_fill_pl4_1:
2874 00002154 59                    <1>       pop    ecx
2875                                 <1>
2876                                 <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0005);
2877 00002155 66B80500              <1>       mov    ax, 0005h
2878 00002159 66BACE03              <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
2879 0000215D 66EF                  <1>       out    dx, ax
2880                                 <1>
2881 0000215F 58                    <1>       pop    eax
2882 00002160 5A                    <1>       pop    edx
2883                                 <1>
2884 00002161 C3                    <1>       retn
2885                                 <1>
2886                                 <1> vgamem_copy_l8:
2887                                 <1>       ; 08/08/2016
2888                                 <1>       ; 07/08/2016
2889                                 <1>       ; 06/08/2016
2890                                 <1>       ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
2891                                 <1>       ;
2892                                 <1>       ; TEMPORARY
2893                                 <1>       ;
2894                                 <1>       ; derived from 'Plex86/Bochs VGABios' source code
2895                                 <1>       ; vgabios-0.7a (2011)
2896                                 <1>       ; by the LGPL VGABios developers Team (2001-2008)
2897                                 <1>       ; 'vgabios.c', 'vgamem_copy_pl4'
2898                                 <1>       ;
2899                                 <1>       ; vgamem_copy_pl4(xstart,ysrc,ydest,cols,nbcols,cheight)
2900                                 <1>       ; cl = xstart, ah = ysrc (i+nblines), ch = ydest (i),
2901                                 <1>       ; bh = cols, [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight
2902                                 <1>
2903                                 <1>       ; src=ysrc*cheight*nbcols+xstart;
2904                                 <1>       ; dest=ydest*cheight*nbcols+xstart;
2905                                 <1>
2906 00002162 52                    <1>       push   edx
2907 00002163 50                    <1>       push   eax
2908                                 <1>
2909                                 <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0105)
2910                                 <1>       ;mov   ax, 0105h
2911                                 <1>       ;mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
2912                                 <1>       ;out   dx, ax
2913                                 <1>
2914                                 <1>       ;mov   ah, [esp+1]
2915                                 <1>
2916 00002164 0FB6D4                <1>       movzx  edx, ah ; ysrc
2917                                 <1>       ; 08/08/2016
2918 00002167 0FB605[C65E0000]      <1>       movzx  eax, byte [CHAR_HEIGHT]
2919 0000216E 8A25[C45E0000]        <1>       mov    ah, [CRT_COLS] ; nbcols
2920 00002174 F6E4                  <1>       mul    ah
2921                                 <1>       ;; 07/08/2016
2922                                 <1>       ;movzx eax, byte [CRT_COLS]
2923                                 <1>       ;shl   ax, 3 ; * 8 ; * [CHAR_HEIGHT]
```

```
2924 00002176 50                  <1>       push   eax ; cheight * nbcols
2925 00002177 F7E2                <1>       mul    edx ; * ysrc
2926                              <1>       ; eax = ysrc * cheight * nbcols
2927                              <1>       ; edx = 0
2928 00002179 88CA                <1>       mov    dl, cl ; edx = xstart
2929 0000217B 01D0                <1>       add    eax, edx
2930 0000217D 89C6                <1>       mov    esi, eax ; src
2931 0000217F 66C1E603            <1>       shl    si, 3 ; * 8 ; 06/08/2016
2932 00002183 88EA                <1>       mov    dl, ch ; ydest
2933 00002185 58                  <1>       pop    eax ; cheight * nbcols
2934 00002186 F7E2                <1>       mul    edx
2935                              <1>       ; eax = ydest * cheight * nbcols
2936 00002188 88CA                <1>       mov    dl, cl ; edx = xstart
2937 0000218A 01D0                <1>       add    eax, edx
2938 0000218C 89C7                <1>       mov    edi, eax ; dest
2939 0000218E 66C1E703            <1>       shl    di, 3 ; * 8 ; 06/08/2016
2940                              <1>       ; esi = src
2941                              <1>       ; edi = dest
2942                              <1>       ; for(i=0;i<cheight;i++)
2943                              <1>       ; {
2944                              <1>       ;  memcpyb(0xa000,dest+i*nbcols,0xa000,src+i*nbcols,cols);
2945                              <1>       ; }
2946 00002192 51                  <1>       push   ecx
2947 00002193 B900000A00          <1>       mov    ecx, 0A0000h
2948 00002198 01CE                <1>       add    esi, ecx
2949 0000219A 01CF                <1>       add    edi, ecx
2950                              <1>       ; 08/08/2016
2951 0000219C 8A35[C65E0000]      <1>       mov    dh, [CHAR_HEIGHT]
2952                              <1>       ;; 07/08/2016
2953                              <1>       ;mov   dh, 8 ; 07/08/2016
2954 000021A2 28D2                <1>       sub    dl, dl ; i
2955                              <1> vgamem_copy_l8_0:
2956 000021A4 56                  <1>       push   esi
2957 000021A5 57                  <1>       push   edi
2958 000021A6 0FB605[C45E0000]    <1>       movzx  eax, byte [CRT_COLS]
2959 000021AD F6E2                <1>       mul    dl
2960                              <1>       ; eax = i * nbcols
2961 000021AF 66C1E003            <1>       shl    ax, 3 ; * 8 ; 06/08/2016
2962 000021B3 01C7                <1>       add    edi, eax ; dest+i*nbcols
2963 000021B5 01C6                <1>       add    esi, eax
2964 000021B7 0FB6CF              <1>       movzx  ecx, bh ; cols
2965 000021BA 66C1E103            <1>       shl    cx, 3 ; * 8 ; 06/08/2016
2966 000021BE F3A4                <1>       rep    movsb
2967 000021C0 5F                  <1>       pop    edi
2968 000021C1 5E                  <1>       pop    esi
2969 000021C2 FEC2                <1>       inc    dl ; 06/08/2016
2970 000021C4 FECE                <1>       dec    dh
2971 000021C6 75DC                <1>       jnz    short vgamem_copy_l8_0
2972                              <1> vgamem_copy_l8_1:
2973 000021C8 59                  <1>       pop    ecx
2974                              <1>
2975                              <1>       ;; outw(VGAREG_GRDC_ADDRESS, 0x0005);
2976                              <1>       ;mov   ax, 0005h
2977                              <1>       ;mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
2978                              <1>       ;out   dx, ax
2979                              <1>
2980 000021C9 58                  <1>       pop    eax
2981 000021CA 5A                  <1>       pop    edx
2982                              <1>
2983 000021CB C3                  <1>       retn
2984                              <1>
2985                              <1> vgamem_fill_l8:
2986                              <1>       ; 08/08/2016
2987                              <1>       ; 07/08/2016
2988                              <1>       ; 06/08/2016
2989                              <1>       ; 04/08/2016
2990                              <1>       ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
2991                              <1>       ;
2992                              <1>       ; TEMPORARY
2993                              <1>       ;
2994                              <1>       ; derived from 'Plex86/Bochs VGABios' source code
2995                              <1>       ; vgabios-0.7a (2011)
2996                              <1>       ; by the LGPL VGABios developers Team (2001-2008)
2997                              <1>       ; 'vgabios.c', 'vgamem_fill_pl4'
2998                              <1>       ;
2999                              <1>       ; vgamem_fill_pl4(xstart,ystart,cols,nbcols,cheight,attr)
3000                              <1>       ; cl = xstart, edi = ch = ystart, bh = cols,
3001                              <1>       ; [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight, attr = 0
3002                              <1>
3003                              <1>       ; dest=ystart*cheight*nbcols+xstart;
3004 000021CC 52                  <1>       push   edx
3005 000021CD 50                  <1>       push   eax
3006                              <1>
3007                              <1>       ;; outw(VGAREG_GRDC_ADDRESS, 0x0205)
3008                              <1>       ;mov   ax, 0205h
3009                              <1>       ;mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
3010                              <1>       ;out   dx, ax
3011                              <1>
3012                              <1>         ; 08/08/2016
3013 000021CE 0FB605[C65E0000]    <1>       movzx   eax, byte [CHAR_HEIGHT]
3014 000021D5 F6E5                <1>       mul    ch
3015                              <1>       ;; 07/08/2016
3016                              <1>       ;movzx eax, ch
3017                              <1>       ;shl   ax, 3 ; * 8 ; * [CHAR_HEIGHT]
3018 000021D7 0FB615[C45E0000]    <1>       movzx  edx, byte [CRT_COLS] ; = [VGA_COLS]
3019 000021DE F7E2                <1>       mul    edx
3020                              <1>       ; edx  = 0
3021 000021E0 88CA                <1>       mov    dl, cl
3022 000021E2 01D0                <1>       add    eax, edx
3023 000021E4 89C7                <1>       mov    edi, eax
3024 000021E6 66C1E703            <1>       shl    di, 3 ; * 8 ; 06/08/2016
3025                              <1>       ; edi = dest
```

```
3026                            <1>       ; for(i=0;i<cheight;i++)
3027                            <1>       ; {
3028                            <1>       ;  memsetb(0xa000,dest+i*nbcols,attr,cols);
3029                            <1>       ; }
3030 000021EA 81C700000A00     <1>       add    edi, 0A0000h
3031 000021F0 51               <1>       push   ecx
3032                            <1>       ; 08/08/2016
3033 000021F1 8A35[C65E0000]   <1>       mov    dh, [CHAR_HEIGHT]
3034                            <1>       ;; 07/08/2016
3035                            <1>       ;mov   dh, 8 ; 07/08/2016
3036 000021F7 28D2             <1>       sub    dl, dl ; i
3037                            <1> vgamem_fill_l8_0:
3038 000021F9 57               <1>       push   edi
3039 000021FA 0FB605[C45E0000] <1>       movzx  eax, byte [CRT_COLS]
3040 00002201 F6E2             <1>       mul    dl
3041                            <1>       ; eax = i * nbcols
3042 00002203 66C1E003         <1>       shl    ax, 3 ; * 8 ; 06/08/2016
3043 00002207 01C7             <1>       add    edi, eax ; dest+i*nbcols
3044 00002209 88D8             <1>       mov    al, bl ; attr ; 04/08/2016
3045 0000220B 0FB6CF           <1>       movzx  ecx, bh ; cols
3046 0000220E 66C1E103         <1>       shl    cx, 3 ; * 8 ; 06/08/2016
3047 00002212 F3AA             <1>       rep    stosb
3048 00002214 5F               <1>       pop    edi
3049 00002215 FEC2             <1>       inc    dl ; 06/08/2016
3050 00002217 FECE             <1>       dec    dh
3051 00002219 75DE             <1>       jnz    short vgamem_fill_l8_0
3052                            <1> vgamem_fill_l8_1:
3053 0000221B 59               <1>       pop    ecx
3054                            <1>
3055                            <1>       ;; outw(VGAREG_GRDC_ADDRESS, 0x0005);
3056                            <1>       ;mov   ax, 0005h
3057                            <1>       ;mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
3058                            <1>       ;out   dx, ax
3059                            <1>
3060 0000221C 58               <1>       pop    eax
3061 0000221D 5A               <1>       pop    edx
3062                            <1>
3063 0000221E C3               <1>       retn
3064                            <1>
3065                            <1> vga_graphics_down:
3066                            <1>       ; 08/08/2016
3067                            <1>       ; 07/08/2016
3068                            <1>       ; 31/07/2016
3069                            <1>       ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
3070                            <1>       ;
3071                            <1>       ; derived from 'Plex86/Bochs VGABios' source code
3072                            <1>       ; vgabios-0.7a (2011)
3073                            <1>       ; by the LGPL VGABios developers Team (2001-2008)
3074                            <1>       ; 'vgabios.c', 'biosfn_scroll'
3075                            <1>       ;
3076                            <1>
3077                            <1>       ; cl = upper left column
3078                            <1>       ; ch = upper left row
3079                            <1>       ; dl = lower rigth column
3080                            <1>       ; dh = lower right row
3081                            <1>       ;
3082                            <1>       ; al = line count (AL=0 means blank entire fields)
3083                            <1>       ; bl = fill value for blanked lines
3084                            <1>       ; bh = unused
3085                            <1>       ;
3086                            <1>       ; ah = [CRT_MODE], current video mode
3087                            <1>
3088 0000221F FC               <1>         cld     ; !!! Clear direction flag !!!
3089                            <1>
3090 00002220 88C7             <1>       mov    bh, al ; 31/07/2016
3091                            <1>
3092 00002222 BE[DE5E0000]     <1>       mov    esi, vga_modes
3093 00002227 89F7             <1>       mov    edi, esi
3094 00002229 83C710           <1>       add    edi, vga_mode_count
3095                            <1> vga_g_down_0:
3096 0000222C AC               <1>       lodsb
3097 0000222D 38E0             <1>       cmp    al, ah ; [CRT_MODE]
3098 0000222F 7405             <1>       je     short vga_g_down_1
3099 00002231 39FE             <1>       cmp    esi, edi
3100 00002233 72F7             <1>       jb     short vga_g_down_0
3101                            <1>       ; xor  bh, bh ; 31/07/2016
3102 00002235 C3               <1>       retn   ; nothing to do
3103                            <1> vga_g_down_1:
3104 00002236 88F8             <1>       mov    al, bh ; 31/07/2016
3105 00002238 83C64F           <1>       add    esi, vga_memmodel - (vga_modes + 1)
3106                            <1>       ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
3107                            <1>
3108                            <1>       ; if(rlr>=nbrows)rlr=nbrows-1;
3109                            <1>       ; if(clr>=nbcols)clr=nbcols-1;
3110                            <1>       ; if(nblines>nbrows)nblines=0;
3111                            <1>       ; cols=clr-cul+1;
3112                            <1>
3113 0000223B 3A35[CA5E0000]   <1>       cmp    dh, [VGA_ROWS]
3114 00002241 7208             <1>       jb     short vga_g_down_2
3115 00002243 8A35[CA5E0000]   <1>       mov    dh, [VGA_ROWS]
3116 00002249 FECE             <1>       dec    dh
3117                            <1> vga_g_down_2:
3118 0000224B 3A15[C45E0000]   <1>       cmp    dl, [CRT_COLS]  ; = [VGA_COLS]
3119 00002251 7208             <1>       jb     short vga_g_down_3
3120 00002253 8A15[C45E0000]   <1>       mov    dl, [CRT_COLS]
3121 00002259 FECA             <1>       dec    dl
3122                            <1> vga_g_down_3:
3123 0000225B 3A05[CA5E0000]   <1>       cmp    al, [VGA_ROWS]
3124 00002261 7602             <1>       jna    short vga_g_down_4
3125 00002263 28C0             <1>       sub    al, al ; 0
3126                            <1> vga_g_down_4:
3127 00002265 88F7             <1>       mov    bh, dh ; clr
```

```
3128 00002267 28CF                <1>       sub    bh, cl ; cul
3129 00002269 FEC7                <1>       inc    bh ; cols = clr-cul+1
3130                              <1>
3131 0000226B 20C0                <1>       and    al, al ; nblines = 0
3132 0000226D 755B                <1>       jnz    short vga_g_down_6
3133 0000226F 20ED                <1>       and    ch, ch ; rul = 0
3134 00002271 7557                <1>       jnz    short vga_g_down_6
3135 00002273 20C9                <1>       and    cl, cl ; cul = 0
3136 00002275 7553                <1>       jnz    short vga_g_down_6
3137                              <1>
3138 00002277 6650                <1>       push   ax
3139 00002279 A0[CA5E0000]        <1>       mov    al, [VGA_ROWS]
3140 0000227E FEC8                <1>       dec    al
3141 00002280 38C6                <1>       cmp    dh, al ; rlr = nbrows-1
3142 00002282 7544                <1>       jne    short vga_g_down_5
3143 00002284 A0[C45E0000]        <1>         mov     al, [CRT_COLS]  ; = VGA_COLS
3144 00002289 FEC8                <1>       dec    al
3145 0000228B 38C2                <1>       cmp    dl, al ; clr = nbcols-1
3146 0000228D 7539                <1>       jne    short vga_g_down_5
3147 0000228F 6658                <1>       pop    ax
3148                              <1>
3149 00002291 66B80502            <1>       mov    ax, 0205h
3150 00002295 66BACE03            <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
3151 00002299 66EF                <1>       out    dx, ax
3152 0000229B A0[CA5E0000]        <1>       mov    al, [VGA_ROWS]
3153 000022A0 8A25[C45E0000]      <1>       mov    ah, [CRT_COLS] ; = [VGA_COLS]
3154 000022A6 F6E4                <1>       mul    ah
3155 000022A8 0FB7D0              <1>       movzx  edx, ax
3156                              <1>       ; 08/08/2016
3157 000022AB 0FB605[C65E0000]    <1>       movzx  eax, byte [CHAR_HEIGHT]
3158 000022B2 F7E2                <1>       mul    edx
3159                              <1>       ; eax = byte count
3160 000022B4 89C1                <1>       mov    ecx, eax
3161                              <1>       ;; 07/08/2016
3162                              <1>       ;shl   dx, 3 ; * 8 ; * [CHAR_HEIGHT]
3163                              <1>       ;mov   ecx, edx
3164 000022B6 88D8                <1>       mov    al, bl ; fill value for blanked lines
3165 000022B8 BF00000A00          <1>       mov    edi, 0A0000h
3166 000022BD F3AA                <1>       rep    stosb
3167                              <1>
3168 000022BF B005                <1>       mov    al, 5
3169 000022C1 66BACE03            <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
3170 000022C5 66EF                <1>       out    dx, ax ; 0005h
3171                              <1>
3172 000022C7 C3                  <1>       retn
3173                              <1>
3174                              <1> vga_g_down_5:
3175 000022C8 6658                <1>       pop    ax
3176                              <1>
3177                              <1> vga_g_down_6:
3178                              <1>       ; [ESI] = VGA memory model number for current video mode
3179                              <1>         ;
3180                              <1>         ; LINEAR8 equ 5
3181                              <1>         ; PLANAR4 equ 4
3182                              <1>         ; PLANAR1 equ 3
3183                              <1>
3184 000022CA 803E04              <1>       cmp    byte [esi], PLANAR4
3185 000022CD 742C                <1>       je     short vga_g_down_planar
3186 000022CF 803E03              <1>       cmp    byte [esi], PLANAR1
3187 000022D2 7427                <1>       je     short vga_g_down_planar
3188                              <1> vga_g_down_linear8:
3189                              <1>       ; 07/07/2016 (TEMPORARY)
3190                              <1>       ;
3191                              <1>       ; cl = upper left column ; cul
3192                              <1>       ; ch = upper left row ; rul
3193                              <1>       ; dl = lower rigth column ; clr
3194                              <1>       ; dh = lower right row ; rlr
3195                              <1>
3196                              <1> vga_g_down_l0:
3197                              <1>       ;{for(i=rlr;i>=rul;i--)
3198                              <1>         ; if((i<rul+nblines)||(nblines==0))
3199 000022D4 08C0                <1>       or     al, al
3200 000022D6 741C                <1>       jz     short vga_g_down_l2
3201 000022D8 88C4                <1>       mov    ah, al
3202 000022DA 00EC                <1>       add    ah, ch
3203                              <1>       ;jc    short vga_g_down_l2
3204 000022DC 86EE                <1>       xchg   ch, dh
3205 000022DE 38E5                <1>       cmp    ch, ah
3206 000022E0 7212                <1>       jb     short vga_g_down_l2
3207 000022E2 88EC                <1>       mov    ah, ch
3208 000022E4 28C4                <1>       sub    ah, al ; ah = i - nblines
3209                              <1>       ; else
3210                              <1>       ; vgamem_copy_pl4(cul,i,i-nblines,cols,nbcols,cheight);
3211 000022E6 E877FEFFFF          <1>       call   vgamem_copy_l8
3212                              <1> vga_g_down_l1:
3213 000022EB 86F5                <1>       xchg   dh, ch
3214 000022ED FECE                <1>       dec    dh
3215 000022EF 38EE                <1>       cmp    dh, ch
3216 000022F1 73E1                <1>       jnb    short vga_g_down_l0
3217 000022F3 C3                  <1>       retn
3218                              <1>
3219                              <1> vga_g_down_l2:
3220                              <1>       ; vgamem_fill_pl4(cul,i,cols,nbcols,cheight,attr);
3221 000022F4 E8D3FEFFFF          <1>       call   vgamem_fill_l8
3222 000022F9 EBF0                <1>       jmp    short vga_g_down_l1
3223                              <1>
3224                              <1> vga_g_down_planar:
3225                              <1>       ; cl = upper left column ; cul
3226                              <1>       ; ch = upper left row ; rul
3227                              <1>       ; dl = lower rigth column ; clr
3228                              <1>       ; dh = lower right row ; rlr
3229                              <1> vga_g_down_pl0:
```

```
                        <1>       ;{for(i=rlr;i>=rul;i--)
                        <1>         ; if((i<rul+nblines)||(nblines==0))
000022FB 08C0           <1>       or     al, al
000022FD 741C           <1>       jz     short vga_g_down_pl2
000022FF 88C4           <1>       mov    ah, al
00002301 00EC           <1>       add    ah, ch
                        <1>       ;jc    short vga_g_down_pl2
00002303 86EE           <1>       xchg   ch, dh
00002305 38E5           <1>       cmp    ch, ah
00002307 7212           <1>       jb     short vga_g_down_pl2
00002309 88EC           <1>       mov    ah, ch
0000230B 28C4           <1>       sub    ah, al ; ah = i - nblines
                        <1>       ; else
                        <1>       ; vgamem_copy_pl4(cul,i,i-nblines,cols,nbcols,cheight);
0000230D E88BFDFFFF     <1>       call   vgamem_copy_pl4
                        <1> vga_g_down_pl1:
00002312 86F5           <1>       xchg   dh, ch
00002314 FECE           <1>       dec    dh
00002316 38EE           <1>       cmp    dh, ch
00002318 73E1           <1>       jnb    short vga_g_down_pl0
0000231A C3             <1>       retn
                        <1>
                        <1> vga_g_down_pl2:
                        <1>       ; vgamem_fill_pl4(cul,i,cols,nbcols,cheight,attr);
0000231B E8EBFDFFFF     <1>       call   vgamem_fill_pl4
00002320 EBF0           <1>       jmp    short vga_g_down_pl1
                        <1>
                        <1> ; 07/07/2016
                        <1> ; 27/06/2016 - TRDOS 386 (TRDOS v2.0)
                        <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                        <1> ;--------------------------------------------------
                        <1> ; SCROLL DOWN
                        <1> ;  THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
                        <1> ; ENTRY --
                        <1> ;  CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
                        <1> ;  DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
                        <1> ;   BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
                        <1> ;  BH = FILL VALUE FOR BLANKED LINES
                        <1> ;  AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
                        <1> ;  DS = DATA SEGMENT
                        <1> ;  ES = REGEN SEGMENT
                        <1> ; EXIT --
                        <1> ;  NOTHING, THE SCREEN IS SCROLLED
                        <1> ;--------------------------------------------------
                        <1>
                        <1>       ; cl = upper left column
                        <1>       ; ch = upper left row
                        <1>       ; dl = lower rigth column
                        <1>       ; dh = lower right row
                        <1>       ;
                        <1>       ; al = line count (AL=0 means blank entire fields)
                        <1>       ; bl = fill value for blanked lines
                        <1>       ; bh = unused
                        <1>
                        <1> GRAPHICS_DOWN:
                        <1>       ; 07/07/2016
                        <1>       ;AH = Current video mode, [CRT_MODE]
                        <1>       ;STD                    ; SET DIRECTION
00002322 80FC07         <1>       cmp    ah, 7
00002325 0F87F4FEFFFF   <1>         ja      vga_graphics_down
                        <1>       ;je    _n0
                        <1>
0000232B 88C7           <1>       MOV    bh, al          ; save line count in BH
0000232D 6689D0         <1>       MOV    AX, DX          ; GET LOWER RIGHT POSITION INTO AX REG
                        <1>
                        <1> ;-----      USE CHARACTER SUBROUTINE FOR POSITIONING
                        <1> ;-----      ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
                        <1>
00002330 E8F1010000     <1>       CALL   GRAPH_POSN
00002335 0FB7F8         <1>       MOVzx  eDI, AX                  ; SAVE RESULT AS DESTINATION ADDRESS
                        <1>
                        <1> ;-----      DETERMINE SIZE OF WINDOW
                        <1>
00002338 6629CA         <1>       SUB    DX, CX
0000233B 6681C20101     <1>         ADD     DX, 101h              ; ADJUST VALUES
00002340 C0E602         <1>       SAL    DH, 2              ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
                        <1>                                 ; AND EVEN/ODD ROWS
                        <1>
                        <1> ;-----      DETERMINE CRT MODE
                        <1>
00002343 803D[C25E0000]06   <1>       CMP    byte [CRT_MODE], 6  ; TEST FOR MEDIUM RES
0000234A 7307           <1>         JNC     short _R12            ; FIND_SOURCE_DOWN
                        <1>
                        <1> ;-----      MEDIUM RES DOWN
0000234C D0E2           <1>       SAL    DL, 1              ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
0000234E 66D1E7         <1>       SAL    DI, 1              ; OFFSET *2 SINCE 2 BYTES/CHAR
00002351 6647           <1>       INC    DI                 ; POINT TO LAST BYTE
                        <1>
                        <1> ;-----      DETERMINE THE SOURCE ADDRESS IN THE BUFFER
                        <1>
                        <1> _R12:                                   ; FIND_SOURCE_DOWN
00002353 81C700800B00   <1>       add    edi, 0B8000h
00002359 6681C7F000     <1>       ADD    DI, 240                 ; POINT TO LAST ROW OF PIXELS
0000235E C0E702         <1>       sal    bh, 2              ; multiply number of lines by 4
00002361 74(06)         <1>       JZ     short 6                  ; IF ZERO, THEN BLANK ENTIRE FIELD
00002363 B050           <1>       MOV    AL, 80             ; 80 BYTES/ROW
00002365 F6E7           <1>       mul    bh                 ; determine offset to source
00002367 89FE           <1>       MOV    eSI, eDI           ; SET UP SOURCE
00002369 6629C6         <1>       SUB    SI, AX             ; SUBTRACT THE OFFSET
0000236C 88F4           <1>       MOV    AH, DH             ; NUMBER OF ROWS IN FIELD
0000236E 28FC           <1>       sub    ah, bh             ; determine number to move
                        <1>
```

```
                        <1> ;-----      LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
                        <1>
                        <1> _R13:                                       ; ROW_LOOP_DOWN
00002370 E823000000     <1>         CALL    _R17                        ; MOVE ONE ROW
00002375 6681EE5020     <1>       SUB    SI, 2000h+80         ; MOVE TO NEXT ROW
0000237A 6681EF5020     <1>       SUB    DI, 2000h+80
0000237F FECC           <1>       DEC    AH                   ; NUMBER OF ROWS TO MOVE
00002381 75ED           <1>         JNZ     short _R13                  ; CONTINUE TILL ALL MOVED
                        <1>
                        <1> ;-----      FILL IN THE VACATED LINE(S)
                        <1> _R14:                                       ; CLEAR_ENTRY_DOWN
00002383 88D8           <1>       mov    al, bl               ; attribute to fill with
                        <1> _R15_:                                      ; CLEAR_LOOP_DOWN
00002385 E82A000000     <1>       CALL   _R18                 ; CLEAR A ROW
0000238A 6681EF5020     <1>       SUB    DI, 2000h+80         ; POINT TO NEXT LINE
0000238F FECF           <1>       dec    bh                   ; number of lines to fill
00002391 75F2           <1>         JNZ     short _R15_                 ; CLEAR_LOOP_DOWN
                        <1>
00002393 C3             <1>       retn                        ; EVERYYHING DONE
                        <1>
                        <1> _R16:                                       ; BLANK_FIELD_DOWN
00002394 88F7           <1>       mov    bh, dh               ; set blank count to everything in field
00002396 EBEB           <1>         JMP     short _R14                  ; CLEAR THE FIELD
                        <1>
                        <1> ; 27/06/2016 - TRDOS 386 (TRDOS v2.0)
                        <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                        <1>
                        <1> ;-----      ROUTINE TO MOVE ONE ROW OF INFORMATION
                        <1>
                        <1> _R17:
00002398 0FB6CA         <1>       MOVzx  ecx, DL                  ; NUMBER OF BYTES IN THE ROW
0000239B 56             <1>       PUSH   eSI
0000239C 57             <1>       PUSH   eDI                  ; SAVE POINTERS
0000239D F3A4           <1>       REP    MOVSB                ; MOVE THE EVEN FIELD
0000239F 5F             <1>       POP    eDI
000023A0 5E             <1>       POP    eSI
000023A1 6681C60020     <1>       ADD    SI, 2000h
000023A6 6681C70020     <1>       ADD    DI, 2000h            ; POINT TO THE ODD FIELD
000023AB 56             <1>       PUSH   eSI
000023AC 57             <1>       PUSH   eDI                  ; SAVE THE POINTERS
000023AD 88D1           <1>       MOV    CL, DL               ; COUNT BACK
000023AF F3A4           <1>       REP    MOVSB                ; MOVE THE ODD FIELD
000023B1 5F             <1>       POP    eDI
000023B2 5E             <1>       POP    eSI                  ; POINTERS BACK
000023B3 C3             <1>       RETn                        ; RETURN TO CALLER
                        <1>
                        <1> ;-----      CLEAR A SINGLE ROW
                        <1>
                        <1> _R18:
000023B4 0FB6CA         <1>       MOVzx  ecx, DL                       ; NUMBER OF BYTES IN FIELD
000023B7 57             <1>       PUSH   eDI                  ; SAVE POINTER
000023B8 F3AA           <1>       REP    STOSB                ; STORE THE NEW VALUE
000023BA 5F             <1>       POP    eDI                  ; POINTER BACK
000023BB 6681C70020     <1>       ADD    DI, 2000h            ; POINT TO ODD FIELD
000023C0 57             <1>       PUSH   eDI
000023C1 88D1           <1>       MOV    CL, DL
000023C3 F3AA           <1>       REP    STOSB                ; FILL THE ODD FIELD
000023C5 5F             <1>       POP    eDI
000023C6 C3             <1>       RETn                        ; RETURN TO CALLER
                        <1>
                        <1> ; 04/07/2016
                        <1> ; 01/07/2016
                        <1> ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)
                        <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                        <1> ;-------------------------------------------------
                        <1> ; GRAPHICS WRITE
                        <1> ;  THIS ROUTINE WRITES THE ASCII CHARACTER TO THE CURRENT
                        <1> ;  POSITION ON THE SCREEN.
                        <1> ; ENTRY --
                        <1> ;  AL = CHARACTER TO WRITE
                        <1> ;  BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR
                        <1> ;     IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN BUFFER
                        <1> ;     (0 IS USED FOR THE BACKGROUND COLOR)
                        <1> ;  CX = NUMBER OF CHARS TO WRITE
                        <1> ;  DS = DATA SEGMENT
                        <1> ;  ES = REGEN SEGMENT
                        <1> ; EXIT --
                        <1> ;  NOTHING IS RETURNED
                        <1> ;
                        <1> ; GRAPHICS READ
                        <1> ;  THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT CURSOR
                        <1> ;  POSITION ON THE SCREEN BY MATCHING THE DOTS ON THE SCREEN TO THE
                        <1> ;  CHARACTER GENERATOR CODE POINTS
                        <1> ; ENTRY --
                        <1> ;  NONE (0 IS ASSUMED AS THE BACKGROUND COLOR)
                        <1> ; EXIT --
                        <1> ;  AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF NONE FOUND)
                        <1> ;
                        <1> ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE CONTAINED IN ROM
                        <1> ;  FOR THE 1ST 128 CHARS.  TO ACCESS CHARS IN THE SECOND HALF, THE USER
                        <1> ;  MUST INITIALIZE THE VECTOR AT INTERRUPT 1FH (LOCATION 0007CH) TO
                        <1> ;  POINT TO THE USER SUPPLIED TABLE OF GRAPHIC IMAGES (8X8 BOXES).
                        <1> ;  FAILURE TO DO SO WILL CAUSE IN STRANGE RESULTS
                        <1> ;-------------------------------------------------------
                        <1>
                        <1> GRAPHICS_WRITE:
000023C7 25FF000000     <1>       and    eax, 0FFh            ; ZERO TO HIGH OF CODE POINT
000023CC 50             <1>       PUSH   eAX                  ; SAVE CODE POINT VALUE
                        <1>
                        <1> ;-----      DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
                        <1>
000023CD E84D010000     <1>       CALL   S26                  ; FIND LOCATION IN REGEN BUFFER
```

```
3434 000023D2 89C7                <1>       MOV    eDI, eAX           ; REGEN POINTER IN DI
3435                              <1>
3436                              <1> ;-----      DETERMINE REGION TO GET CODE POINTS FROM
3437                              <1>
3438 000023D4 58                  <1>       POP    eAX                ; RECOVER CODE POINT
3439                              <1>
3440 000023D5 BE[7C260100]        <1>       MOV    eSI, CRT_CHAR_GEN  ; OFFSET OF IMAGES
3441                              <1>
3442                              <1> ;-----      DETERMINE GRAPHICS MODE IN OPERATION
3443                              <1>                                  ; DETERMINE_MODE
3444 000023DA 66C1E003            <1>       SAL    AX, 3              ; MULTIPLY CODE POINT VALUE BY 8
3445 000023DE 01C6                <1>       ADD    eSI, eAX           ; SI HAS OFFSET OF DESIRED CODES
3446                              <1>
3447 000023E0 803D[C25E0000]06    <1>       CMP    byte [CRT_MODE], 6
3448 000023E7 7231                <1>       JC     short S6           ; TEST FOR MEDIUM RESOLUTION MODE
3449                              <1>
3450                              <1> ;-----      HIGH RESOLUTION MODE
3451                              <1>
3452 000023E9 81C700800B00        <1>       add    edi, 0B8000h
3453                              <1> S1:                              ; HIGH_CHAR
3454 000023EF 57                  <1>       PUSH   eDI                ; SAVE REGEN POINTER
3455 000023F0 56                  <1>       PUSH   eSI                ; SAVE CODE POINTER
3456 000023F1 B604                <1>       MOV    DH, 4              ; NUMBER OF TIMES THROUGH LOOP
3457                              <1> S2:
3458 000023F3 AC                  <1>       LODSB                     ; GET BYTE FROM CODE POINTS
3459 000023F4 F6C380              <1>       TEST   BL, 80H                  ; SHOULD WE USE THE FUNCTION
3460 000023F7 7515                <1>       JNZ    short S5           ; TO PUT CHAR IN
3461 000023F9 AA                  <1>       STOSB                     ; STORE IN REGEN BUFFER
3462 000023FA AC                  <1>       LODSB
3463                              <1> S4:
3464 000023FB 8887FF1F0000        <1>       MOV    [eDI+2000H-1], AL ; STORE IN SECOND HALF
3465 00002401 83C74F              <1>       ADD    eDI, 79                  ; MOVE TO NEXT ROW IN REGEN
3466 00002404 FECE                <1>       DEC    DH                 ; DONE WITH LOOP
3467 00002406 75EB                <1>       JNZ    short S2
3468 00002408 5E                  <1>       POP    eSI
3469 00002409 5F                  <1>       POP    eDI                ; RECOVER REGEN POINTER
3470 0000240A 47                  <1>       INC    eDI                ; POINT TO NEXT CHAR POSITION
3471 0000240B E2E2                <1>       LOOP   S1                 ; MORE CHARS TO WRITE
3472 0000240D C3                  <1>       retn
3473                              <1>
3474                              <1> S5:
3475 0000240E 3207                <1>       XOR    AL, [eDI]          ; EXCLUSIVE OR WITH CURRENT
3476 00002410 AA                  <1>       STOSB                     ; STORE THE CODE POINT
3477 00002411 AC                  <1>       LODSB                     ; AGAIN FOR ODD FIELD
3478 00002412 3287FF1F0000        <1>       XOR    AL, [eDI+2000H-1]
3479 00002418 EBE1                <1>       JMP    short S4           ; BACK TO MAINSTREAM
3480                              <1>
3481                              <1> ;-----      MEDIUM RESOLUTION WRITE
3482                              <1> S6:                              ; MED_RES_WRITE
3483 0000241A 88DA                <1>       MOV    DL, BL             ; SAVE HIGH COLOR BIT
3484 0000241C 66D1E7              <1>       SAL    DI, 1              ; OFFSET*2 SINCE 2 BYTES/CHAR
3485                              <1>                                  ; EXPAND BL TO FULL WORD OF COLOR
3486 0000241F 80E303              <1>       AND    BL, 3              ; ISOLATE THE COLOR BITS ( LOW 2 BITS )
3487 00002422 B055                <1>       MOV    AL, 055H           ; GET BIT CONVERSION MULTIPLIER
3488 00002424 F6E3                <1>       MUL    BL                 ; EXPAND 2 COLOR BITS TO 4 REPLICATIONS
3489 00002426 88C3                <1>       MOV    BL, AL             ; PLACE BACK IN WORK REGISTER
3490 00002428 88C7                <1>       MOV    BH, AL             ; EXPAND TO 8 REPLICATIONS OF COLOR BITS
3491 0000242A 81C700800B00        <1>       add    edi, 0B8000h
3492                              <1> S7:                                   ; MED_CHAR
3493 00002430 57                  <1>       PUSH   eDI                ; SAVE REGEN POINTER
3494 00002431 56                  <1>       PUSH   eSI                ; SAVE THE CODE POINTER
3495 00002432 B604                <1>       MOV    DH, 4              ; NUMBER OF LOOPS
3496                              <1> S8:
3497 00002434 AC                  <1>       LODSB                     ; GET CODE POINT
3498 00002435 E8B3000000          <1>       CALL   S21                ; DOUBLE UP ALL THE BITS
3499 0000243A 6621D8              <1>       AND    AX, BX             ; CONVERT TO FOREGROUND COLOR ( 0 BACK )
3500 0000243D 86E0                <1>       XCHG   AH, AL             ; SWAP HIGH/LOW BYTES FOR WORD MOVE
3501 0000243F F6C280              <1>       TEST   DL, 80H                 ; IS THIS XOR FUNCTION
3502 00002442 7403                <1>       JZ     short S9           ; NO, STORE IT IN AS IS
3503 00002444 663307              <1>       XOR    AX, [eDI]          ; DO FUNCTION WITH LOW/HIGH
3504                              <1> S9:
3505 00002447 668907              <1>       MOV    [eDI], AX          ; STORE FIRST BYTE HIGH, SECOND LOW
3506 0000244A AC                  <1>       LODSB                     ; GET CODE POINT
3507 0000244B E89D000000          <1>       CALL   S21
3508 00002450 6621D8              <1>       AND    AX, BX             ; CONVERT TO COLOR
3509 00002453 86E0                <1>       XCHG   AH, AL             ; SWAP HIGH/LOW BYTES FOR WORD MOVE
3510 00002455 F6C280              <1>       TEST   DL, 80H                 ; AGAIN, IS THIS XOR FUNCTION
3511 00002458 7407                <1>       JZ     short _S10         ; NO, JUST STORE THE VALUES
3512 0000245A 66338700200000      <1>       XOR    AX, [eDI+2000H]         ; FUNCTION WITH FIRST HALF LOW
3513                              <1> _S10:
3514 00002461 66898700200000      <1>       MOV    [eDI+2000H], AX         ; STORE SECOND PORTION HIGH
3515 00002468 6683C750            <1>       ADD    DI, 80             ; POINT TO NEXT LOCATION
3516 0000246C FECE                <1>       DEC    DH
3517 0000246E 75C4                <1>       JNZ    short S8           ; KEEP GOING
3518 00002470 5E                  <1>       POP    eSI                ; RECOVER CODE POINTER
3519 00002471 5F                  <1>       POP    eDI                ; RECOVER REGEN POINTER
3520 00002472 47                  <1>       INC    eDI                ; POINT TO NEXT CHAR POSITION
3521 00002473 47                  <1>       INC    eDI
3522 00002474 E2BA                <1>       LOOP   S7                 ; MORE TO WRITE
3523 00002476 C3                  <1>       retn
3524                              <1>
3525                              <1> ; 04/07/2016
3526                              <1> ; 01/07/2016
3527                              <1> ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)
3528                              <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
3529                              <1> ;---------------------------------------
3530                              <1> ; GRAPHICS READ
3531                              <1> ;---------------------------------------
3532                              <1> GRAPHICS_READ:
3533 00002477 E8A3000000          <1>       CALL   S26                ; CONVERTED TO OFFSET IN REGEN
3534 0000247C 89C6                <1>       MOV    eSI, eAX           ; SAVE IN SI
3535 0000247E 81C600800B00        <1>       add    esi, 0B8000h       ; 01/07/2016
```

```
3536 00002484 83EC08              <1>       SUB    eSP, 8              ; ALLOCATE SPACE FOR THE READ CODE POINT
3537 00002487 89E5                <1>       MOV    eBP, eSP            ; POINTER TO SAVE AREA
3538                              <1>
3539                              <1> ;-----      DETERMINE GRAPHICS MODES
3540 00002489 B604                <1>       mov    dh, 4               ; number of passes ; 01/07/2016
3541 0000248B 803D[C25E0000]06    <1>       CMP    byte [CRT_MODE], 6
3542 00002492 7219                <1>       JC     short S12           ; MEDIUM RESOLUTION
3543                              <1>
3544                              <1> ;-----      HIGH RESOLUTION READ
3545                              <1> ;-----      GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
3546                              <1>       ;MOV   DH,4                ; NUMBER OF PASSES
3547                              <1> S11:
3548 00002494 8A06                <1>       MOV    AL, [eSI]           ; GET FIRST BYTE
3549 00002496 884500              <1>       MOV    [eBP], AL           ; SAVE IN STORAGE AREA
3550 00002499 45                  <1>       INC    eBP                 ; NEXT LOCATION
3551 0000249A 8A8600200000        <1>       MOV    AL, [eSI+2000H]           ; GET LOWER REGION BYTE
3552 000024A0 884500              <1>       MOV    [eBP], AL           ; ADJUST AND STORE
3553 000024A3 45                  <1>       INC    eBP
3554 000024A4 83C650              <1>       ADD    eSI, 80                   ; POINTER INTO REGEN
3555 000024A7 FECE                <1>       DEC    DH                  ; LOOP CONTROL
3556 000024A9 75E9                <1>       JNZ    short S11           ; DO IT SOME MORE
3557 000024AB EB1D                <1>       JMP    SHORT S14           ; GO MATCH THE SAVED CODE POINTS
3558                              <1>
3559                              <1> ;-----      MEDIUM RESOLUTION READ
3560                              <1> S12:
3561 000024AD 66D1E6              <1>       SAL    SI, 1               ; OFFSET*2 SINCE 2 BYTES/CHAR
3562                              <1>       ;MOV   DH, 4               ; NUMBER OF PASSES
3563                              <1> S13:
3564 000024B0 E84D000000          <1>       CALL   S23                 ; GET BYTES FROM REGEN INTO SINGLE SAVE
3565 000024B5 81C6FE1F0000        <1>       ADD    eSI, 2000H-2        ; GO TO LOWER REGION
3566 000024BB E842000000          <1>       CALL   S23                 ; GET THIS PAIR INTO SAVE
3567 000024C0 81EEB21F0000        <1>       SUB    eSI, 2000H-80+2           ; ADJUST POINTER BACK INTO UPPER
3568 000024C6 FECE                <1>       DEC    DH
3569 000024C8 75E6                <1>       JNZ    short S13           ; KEEP GOING UNTIL ALL 8 DONE
3570                              <1>
3571                              <1> ;-----      SAVE AREA HAS CHARACTER IN IT, MATCH IT
3572                              <1> S14:                           ; FIND_CHAR
3573 000024CA BF[7C260100]        <1>       MOV    eDI, CRT_CHAR_GEN   ; ESTABLISH ADDRESSING
3574 000024CF 83ED08              <1>       SUB    eBP, 8              ; ADJUST POINTER TO START OF SAVE AREA
3575 000024D2 89EE                <1>       MOV    eSI, eBP
3576                              <1> S15:
3577 000024D4 66B80001            <1>       mov    ax, 256                  ; NUMBER TO TEST AGAINST
3578                              <1> S16:
3579 000024D8 56                  <1>       PUSH   eSI                 ; SAVE SAVE AREA POINTER
3580 000024D9 57                  <1>       PUSH   eDI                 ; SAVE CODE POINTER
3581                              <1>       ;MOV   eCX, 4              ; NUMBER OF WORDS TO MATCH
3582                              <1>       ;REPE  CMPSW               ; COMPARE THE 8 BYTES AS WORDS
3583 000024DA A7                  <1>       cmpsd                      ; compare first 4 bytes
3584 000024DB 7501                <1>       jne    short S17           ;
3585 000024DD A7                  <1>       cmpsd                      ; compare last 4 bytes
3586                              <1> S17:
3587 000024DE 5F                  <1>       POP    eDI                 ; RECOVER THE POINTERS
3588 000024DF 5E                  <1>       POP    eSI
3589                              <1>       ;JZ    short S18           ; IF ZERO FLAG SET, THEN MATCH OCCURRED
3590 000024E0 7407                <1>       je     short S18
3591                              <1>       ;                          ; NO MATCH, MOVE ON TO NEXT
3592 000024E2 83C708              <1>       ADD    eDI, 8              ; NEXT CODE POINT
3593 000024E5 6648                <1>       dec    ax                  ; LOOP CONTROL
3594 000024E7 75EF                <1>       JNZ    short S16           ; DO ALL OF THEM
3595                              <1>
3596                              <1> ;-----      CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
3597                              <1> S18:
3598 000024E9 83C408              <1>       ADD    eSP, 8              ; READJUST THE STACK, THROW AWAY SAVE
3599 000024EC C3                  <1>       retn                       ; ALL DONE
3600                              <1>
3601                              <1> ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)
3602                              <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
3603                              <1> ;--------------------------------------------
3604                              <1> ; EXPAND BYTE
3605                              <1> ;  THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
3606                              <1> ;  OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
3607                              <1> ;  THE RESULT IS LEFT IN AX
3608                              <1> ;--------------------------------------------
3609                              <1> S21:
3610 000024ED 6651                <1>       PUSH   CX                  ; SAVE REGISTER
3611                              <1>       ;MOV   CX, 8               ; SHIFT COUNT REGISTER FOR ONE BYTE
3612 000024EF B108                <1>       mov    cl, 8
3613                              <1> S22:
3614 000024F1 D0C8                <1>       ROR    AL,1                ; SHIFT BITS, LOW BIT INTO CARRY FLAG
3615 000024F3 66D1DD              <1>       RCR    BP,1                ; MOVE CARRY FLAG (LOW BIT INTO RESULTS
3616 000024F6 66D1FD              <1>       SAR    BP,1                ; SIGN EXTEND HIGH BIT (DOUBLE IT)
3617                              <1>       ;LOOP  S22                 ; REPEAT FOR ALL 8 BITS
3618 000024F9 FEC9                <1>       dec    cl
3619 000024FB 75F4                <1>       jnz    short S22
3620 000024FD 6695                <1>       XCHG   AX, BP              ; MOVE RESULTS TO PARAMETER REGISTER
3621 000024FF 6659                <1>       POP    CX                  ; RECOVER REGISTER
3622 00002501 C3                  <1>       RETn                       ; ALL DONE
3623                              <1>
3624                              <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
3625                              <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
3626                              <1> ;--------------------------------------------------
3627                              <1> ; MED_READ_BYTE
3628                              <1> ; THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
3629                              <1> ;  COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
3630                              <1> ;  THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
3631                              <1> ;  POSITION IN THE SAVE AREA
3632                              <1> ; ENTRY --
3633                              <1> ;  SI,DS = POINTER TO REGEN AREA OF INTEREST
3634                              <1> ;  BX = EXPANDED FOREGROUND COLOR
3635                              <1> ;  BP = POINTER TO SAVE AREA
3636                              <1> ; EXIT --
3637                              <1> ;  SI AND BP ARE INCREMENTED
```

```
                        <1> ;--------------------------------------------------
                        <1> S23:
00002502 66AD           <1>       LODSW                   ; GET FIRST BYTE AND SECOND BYTES
00002504 86C4           <1>       XCHG   AL, AH           ; SWAP FOR COMPARE
00002506 66B900C0       <1>       MOV    CX, 0C000H       ; 2 BIT MASK TO TEST THE ENTRIES
0000250A B200           <1>       MOV    DL, 0            ; RESULT REGISTER
                        <1> S24:
0000250C 6685C8         <1>       TEST   AX, CX           ; IS THIS SECTION BACKCROUND?
0000250F 7401           <1>         JZ     short S25            ; IF ZERO, IT IS BACKGROUND (CARRY=0)
00002511 F9             <1>       STC                     ; WASN'T, SO SET CARRY
                        <1> S25:
00002512 D0D2           <1>       RCL    DL, 1            ; MOVE THAT BIT INTO THE RESULT
00002514 66C1E902       <1>       SHR    CX, 2            ; MOVE THE MASK TO THE RIGHT BY 2 BITS
00002518 73F2           <1>       JNC    short S24        ; DO IT AGAIN IF MASK DIDN'T FALL OUT
0000251A 885500         <1>       MOV    [eBP], DL        ; STORE RESULT IN SAVE AREA
0000251D 45             <1>       INC    eBP              ; ADJUST POINTER
0000251E C3             <1>       RETn                    ; ALL DONE
                        <1>
                        <1> ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)
                        <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                        <1> ;----------------------------------------
                        <1> ; V4_POSITION
                        <1> ;  THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN
                        <1> ;  THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET
                        <1> ;  INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.
                        <1> ;  FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST
                        <1> ;  BE DOUBLED.
                        <1> ; ENTRY -- NO REGISTERS,MEMORY LOCATION @CURSOR_POSN IS USED
                        <1> ; EXIT--
                        <1> ;  AX CONTAINS OFFSET INTO REGEN BUFFER
                        <1> ;----------------------------------------
                        <1> S26:
0000251F 0FB705[1E520100]   <1>       movzx  eax, word [CURSOR_POSN]   ; GET CURRENT CURSOR
                        <1> GRAPH_POSN:
00002526 53             <1>       PUSH   eBX              ; SAVE REGISTER
00002527 0FB6D8         <1>       movzx  ebx, al                ; SAVE A COPY OF CURRENT CURSOR
0000252A A0[C45E0000]   <1>       MOV    AL, [CRT_COLS]         ; GET BYTES PER COLUMN
0000252F F6E4           <1>       MUL    AH               ; MULTIPLY BY ROWS
00002531 66C1E002       <1>       SHL    AX, 2            ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
00002535 01D8           <1>       ADD    eAX, eBX         ; DETERMINE OFFSET
00002537 5B             <1>       POP    eBX              ; RECOVER POINTER
00002538 C3             <1>       RETn                    ; ALL DONE
                        <1>
                        <1> ; 09/07/2016
                        <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
                        <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                        <1> ;----------------------------------------------
                        <1> ; SET_COLOR
                        <1> ;     THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE OVERSCAN COLOR,
                        <1> ;     AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION GRAPHICS
                        <1> ; INPUT
                        <1> ;     (BH) HAS COLOR ID
                        <1> ;           IF BH=0, THE BACKGROUND COLOR VALUE IS SET
                        <1> ;                 FROM THE LOW BITS OF BL (0-31)
                        <1> ;           IF BH=1, THE PALETTE SELECTION IS MADE
                        <1> ;                 BASED ON THE LOW BIT OF BL:
                        <1> ;                       0 = GREEN, RED, YELLOW FOR COLORS 1,2,3
                        <1> ;                       1 = BLUE, CYAN, MAGENTA FOR COLORS 1,2,3
                        <1> ;     (BL) HAS THE COLOR VALUE TO BE USED
                        <1> ; OUTPUT
                        <1> ;     THE COLOR SELECTION IS UPDATED
                        <1> ;----------------------------------------------
                        <1> SET_COLOR:
00002539 803D[C25E0000]07   <1>         cmp     byte [CRT_MODE], 7      ; 09/07/2016
00002540 0F870EF0FFFF       <1>       ja     VIDEO_RETURN       ; nothing to do for VGA modes
                        <1>
                        <1>       ;MOV   DX, [ADDR_6845]           ; I/O PORT FOR PALETTE
                        <1>       ;mov   dx, 3D4h
                        <1>       ;ADD   DX,5              ; OVERSCAN PORT
00002546 66BAD903       <1>       mov    dx, 3D9h
0000254A A0[C55E0000]   <1>       MOV    AL, [CRT_PALETTE]   ; GET THE CURRENT PALETTE VALUE
0000254F 08FF           <1>       OR     BH, BH           ; IS THIS COLOR 0?
00002551 7512           <1>       JNZ    short M20        ; OUTPUT COLOR 1
                        <1>
                        <1> ;-----      HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
                        <1>
00002553 24E0           <1>       AND    AL, 0E0H         ; TURN OFF LOW 5 BITS OF CURRENT
00002555 80E31F         <1>       AND    BL, 01FH         ; TURN OFF HIGH 3 BITS OF INPUT VALUE
00002558 08D8           <1>       OR     AL, BL           ; PUT VALUE INTO REGISTER
                        <1> M19:                          ; OUTPUT THE PALETTE
0000255A EE             <1>       OUT    DX, AL           ; OUTPUT COLOR SELECTION TO 3D9 PORT
0000255B A2[C55E0000]   <1>       MOV    [CRT_PALETTE], AL   ; SAVE THE COLOR VALUE
00002560 E9EFEFFFFF     <1>       JMP    VIDEO_RETURN
                        <1>
                        <1> ;-----      HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED
                        <1>
                        <1> M20:
00002565 24DF           <1>       AND    AL, 0DFH         ; TURN OFF PALETTE SELECT BIT
00002567 D0EB           <1>       SHR    BL, 1            ; TEST THE LOW ORDER BIT OF BL
00002569 73EF           <1>       JNC    short M19        ; ALREADY DONE
0000256B 0C20           <1>       OR     AL, 20H               ; TURN ON PALETTE SELECT BIT
0000256D EBEB           <1>       JMP    short M19        ; GO DO IT
                        <1>
                        <1> ; 09/07/2016
                        <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
                        <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                        <1> ;----------------------------------------
                        <1> ; READ DOT -- WRITE DOT
                        <1> ; THESE ROUTINES WILL WRITE A DOT, OR READ THE
                        <1> ;  DOT AT THE INDICATED LOCATION
                        <1> ; ENTRY --
                        <1> ;   DX = ROW (0-199)     (THE ACTUAL VALUE DEPENDS ON THE MODE)
```

```
                        <1> ;   CX = COLUMN ( 0-639) ( THE VALUES ARE NOT RANGE CHECKED )
                        <1> ;   AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
                        <1> ;     REQUIRED FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
                        <1> ;     BIT 7 OF AL = 1 INDICATES XOR THE VALUE INTO THE LOCATION
                        <1> ;   DS = DATA SEGMENT
                        <1> ;   ES = REGEN SEGMENT
                        <1> ;
                        <1> ; EXIT
                        <1> ;     AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
                        <1> ;-------------------------------------------
                        <1>
                        <1> READ_DOT:
                        <1>       ; 09/07/2016
0000256F 8A25[C25E0000] <1>       mov    ah,  [CRT_MODE]
00002575 80FC07         <1>       cmp    ah, 7 ; 6!?
00002578 760A           <1>       jna    short read_dot_cga
                        <1>
0000257A E8CB030000     <1>       call   vga_read_pixel
                        <1>       ; al = pixel value
0000257F E9D5EFFFFF     <1>       jmp    _video_return
                        <1>
                        <1> read_dot_cga:
                        <1>       ;je    VIDEO_RETURN ; 7
00002584 80FC04         <1>       cmp    ah, 4 ; graphics ?
00002587 0F82C7EFFFFF   <1>       jb     VIDEO_RETURN ; no, text mode, nothing to do
                        <1>
0000258D E855000000     <1>       CALL   R3                  ; DETERMINE BYTE POSITION OF DOT
00002592 8A06           <1>       MOV    AL, [eSI]           ; GET THE BYTE
00002594 20E0           <1>       AND    AL, AH              ; MASK OFF THE OTHER BITS IN THE BYTE
00002596 D2E0           <1>       SHL    AL, CL              ; LEFT JUSTIFY THE VALUE
00002598 88F1           <1>       MOV    CL, DH              ; GET NUMBER OF BITS IN RESULT
0000259A D2C0           <1>       ROL    AL, CL              ; RIGHT JUSTIFY THE RESULT
                        <1>       ;JMP   VIDEO_RETURN        ; RETURN FROM VIDEO I/O
0000259C 0FB6C0         <1>       movzx  eax, al
0000259F E9B5EFFFFF     <1>       jmp    _video_return
                        <1>
                        <1> ; 09/07/2016
                        <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
                        <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                        <1>
                        <1> WRITE_DOT:
                        <1>       ; 09/07/2016
000025A4 8A25[C25E0000] <1>       mov    ah, [CRT_MODE]
000025AA 80FC07         <1>       cmp    ah, 7 ; 6!?
000025AD 760A           <1>       jna    short write_dot_cga
                        <1>
000025AF E805030000     <1>       call   vga_write_pixel
000025B4 E99BEFFFFF     <1>       jmp    VIDEO_RETURN
                        <1>
                        <1> write_dot_cga:
                        <1>       ;je    VIDEO_RETURN ; 7
000025B9 80FC04         <1>       cmp    ah, 4 ; graphics ?
000025BC 0F8292EFFFFF   <1>       jb     VIDEO_RETURN ; no, text mode, nothing to do
                        <1>
                        <1>       ;PUSH  AX                  ; SAVE DOT VALUE
000025C2 6650           <1>       PUSH   AX                  ; TWICE
000025C4 E81E000000     <1>       CALL   R3                  ; DETERMINE BYTE POSITION OF THE DOT
000025C9 D2E8           <1>       SHR    AL, CL              ; SHIFT TO SET UP THE BITS FOR OUTPUT
000025CB 20E0           <1>       AND    AL, AH              ; STRIP OFF THE OTHER BITS
000025CD 8A0E           <1>       MOV    CL, [eSI]           ; GET THE CURRENT BYTE
000025CF 665B           <1>       POP    BX                  ; RECOVER XOR FLAG
000025D1 F6C380         <1>       TEST   BL, 80H                   ; IS IT ON
000025D4 750D           <1>       JNZ    short R2            ; YES, XOR THE DOT
000025D6 F6D4           <1>       NOT    AH                  ; SET MASK TO REMOVE THE INDICATED BITS
000025D8 20E1           <1>       AND    CL, AH
000025DA 08C8           <1>       OR     AL, CL              ; OR IN THE NEW VALUE OF THOSE BITS
                        <1> R1:                               ; FINISH_DOT
000025DC 8806           <1>       MOV    [eSI], AL           ; RESTORE THE BYTE IN MEMORY
                        <1>       ;POP   AX
000025DE E971EFFFFF     <1>       JMP    VIDEO_RETURN        ; RETURN FROM VIDEO I/O
                        <1> R2:                               ; XOR_DOT
000025E3 30C8           <1>       XOR    AL, CL              ; EXCLUSIVE OR THE DOTS
000025E5 EBF5           <1>       JMP    short R1            ; FINISH UP THE WRITING
                        <1>
                        <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
                        <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                        <1>
                        <1> ;-------------------------------------------
                        <1> ; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
                        <1> ; INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
                        <1> ; ENTRY --
                        <1> ;  DX = ROW VALUE (0-199)
                        <1> ;  CX = COLUMN VALUE (0-639)
                        <1> ; EXIT --
                        <1> ;  SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
                        <1> ;  AH = MASK TO STRIP OFF THE BITS OF INTEREST
                        <1> ;  CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
                        <1> ;  DH = # BITS IN RESULT
                        <1> ;  BX = MODIFIED
                        <1> ;-------------------------------------------
                        <1> R3:
                        <1>
                        <1> ;-----      DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE BY 40
                        <1> ;-----       ( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW )
                        <1>
000025E7 0FB7F0         <1>       movzx  esi, ax                   ; WILL SAVE AL AND AH DURING OPERATION
000025EA B028           <1>       MOV    AL, 40
000025EC F6E2           <1>       MUL    DL                  ; AX= ADDRESS OF START OF INDICATED ROW
000025EE A808           <1>       TEST   AL, 08H             ; TEST FOR EVEN/ODD ROW CALCULATED
000025F0 7404           <1>       JZ     short R4            ; JUMP IF EVEN ROW
000025F2 6605D81F       <1>       ADD    AX, 2000H-40        ; OFFSET TO LOCATION OF ODD ROWS ADJUST
                        <1> R4:                               ; EVEN_ROW
```

```
3842 000025F6 6696               <1>       XCHG   SI, AX              ; MOVE POINTER TO (SI) AND RECOVER (AX)
3843 000025F8 81C600800B00       <1>       add    esi, 0B8000h
3844 000025FE 6689CA             <1>       MOV    DX, CX              ; COLUMN VALUE TO DX
3845                             <1>
3846                             <1> ;-----      DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
3847                             <1>
3848                             <1> ; SET UP THE REGISTERS ACCORDING TO THE MODE
3849                             <1> ; CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3 FOR HIGH/MED RES )
3850                             <1> ; CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2 FOR H/M )
3851                             <1> ; BL = MASK TO SELECT BITS FROM POINTED BYTE ( 80H/C0H FOR H/M )
3852                             <1> ; BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2 FOR H/M )
3853                             <1>
3854 00002601 66BBC002           <1>       MOV    BX, 2C0H
3855 00002605 66B90203           <1>       MOV    CX, 302H            ; SET PARMS FOR MED RES
3856 00002609 803D[C25E0000]06   <1>       CMP    byte [CRT_MODE], 6
3857 00002610 7208               <1>       JC     short R5            ; HANDLE IF MED RES
3858 00002612 66BB8001           <1>       MOV    BX, 180H
3859 00002616 66B90307           <1>       MOV    CX, 703H            ; SET PARMS FOR HIGH RES
3860                             <1>
3861                             <1> ;-----      DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
3862                             <1> R5:
3863 0000261A 20D5               <1>       AND    CH, DL              ; ADDRESS OF PEL WITHIN BYTE TO CH
3864                             <1>
3865                             <1> ;-----      DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
3866                             <1>
3867 0000261C 66D3EA             <1>       SHR    DX, CL              ; SHIFT BY CORRECT AMOUNT
3868 0000261F 6601D6             <1>       ADD    SI, DX              ; INCREMENT THE POINTER
3869 00002622 88FE               <1>       MOV    DH, BH              ; GET THE # OF BITS IN RESULT TO DH
3870                             <1>
3871                             <1> ;-----      MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
3872                             <1>
3873 00002624 28C9               <1>       SUB    CL, CL              ; ZERO INTO STORAGE LOCATION
3874                             <1> R6:
3875 00002626 D0C8               <1>       ROR    AL, 1               ; LEFT JUSTIFY VALUE IN AL (FOR WRITE)
3876 00002628 00E9               <1>       ADD    CL, CH              ; ADD IN THE BIT OFFSET VALUE
3877 0000262A FECF               <1>       DEC    BH                  ; LOOP CONTROL
3878 0000262C 75F8               <1>       JNZ    short R6            ; ON EXIT, CL HAS COUNT TO RESTORE BITS
3879 0000262E 88DC               <1>       MOV    AH, BL              ;  GET MASK TO AH
3880 00002630 D2EC               <1>       SHR    AH, CL              ;  MOVE THE MASK TO CORRECT LOCATION
3881 00002632 C3                 <1>       RETn                       ;  RETURN WITH EVERYTHING SET UP
3882                             <1>
3883                             <1> load_dac_palette:
3884                             <1>       ; 29/07/2016
3885                             <1>       ; 23/07/2016
3886                             <1>       ; 03/07/2016 (TRDOS 386 = TRDOS v2.0)
3887                             <1>       ; (set_mode_vga)
3888                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
3889                             <1>       ; vgabios-0.7a (2011)
3890                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
3891                             <1>       ; 'vgabios.c', 'load_dac_palette'
3892                             <1>       ;
3893                             <1>       ; Oracle VirtualBox 5.0.24 VGABios Source Code
3894                             <1>       ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
3895                             <1>       ;
3896                             <1>       ; INPUT -> AH = DAC selection number (3, 2 or 1)
3897                             <1>       ; OUTPUT -> ECX = 0, AX = 0
3898                             <1>       ; (Modifed registers: EAX, ECX, EDX, ESI)
3899                             <1>       ;
3900 00002633 66BAC803           <1>       mov    dx, 3C8h  ; VGAREG_DAC_WRITE_ADDRESS
3901 00002637 28C0               <1>       sub    al, al ; 0
3902 00002639 EE                 <1>       out    dx, al ; 0 ; color index, always 0 at the beginning
3903 0000263A 6642               <1>       inc    dx   ; 3C9h ; VGAREG_DAC_DATA
3904 0000263C B900010000         <1>       mov    ecx, 256   ; always 256*3 values
3905                             <1>       ;push  esi
3906 00002641 88E0               <1>       mov    al, ah
3907 00002643 B43F               <1>       mov    ah, 3Fh      ; 3Fh except DAC selection number 3
3908 00002645 3C02               <1>       cmp    al, 2
3909 00002647 7414               <1>       je     short l_dac_p_2
3910 00002649 7719               <1>       ja     short l_dac_p_3
3911 0000264B 20C0               <1>       and    al, al
3912 0000264D 7507               <1>       jnz    short l_dac_p_1
3913                             <1> l_dac_p_0:
3914 0000264F BE[3C210100]       <1>       mov    esi, palette0
3915 00002654 EB15               <1>       jmp    short l_dac_p_4
3916                             <1> l_dac_p_1:
3917 00002656 BE[FC210100]       <1>       mov    esi, palette1
3918 0000265B EB0E               <1>       jmp    short l_dac_p_4
3919                             <1> l_dac_p_2:
3920 0000265D BE[BC220100]       <1>       mov    esi, palette2
3921 00002662 EB07               <1>       jmp    short l_dac_p_4
3922                             <1> l_dac_p_3:
3923 00002664 B4FF               <1>       mov    ah, 0FFh ; dac registers
3924 00002666 BE[7C230100]       <1>       mov    esi, palette3
3925                             <1> l_dac_p_4:
3926 0000266B AC                 <1>       lodsb
3927 0000266C EE                 <1>       out    dx, al  ; Red
3928 0000266D AC                 <1>       lodsb
3929 0000266E EE                 <1>       out    dx, al ; Green
3930 0000266F AC                 <1>       lodsb
3931 00002670 EE                 <1>       out    dx, al ; Blue
3932 00002671 20E4               <1>       and    ah, ah
3933 00002673 7405               <1>       jz     short l_dac_p_5
3934 00002675 FECC               <1>       dec    ah
3935 00002677 E2F2               <1>       loop   l_dac_p_4
3936                             <1>       ;pop   esi
3937 00002679 C3                 <1>       retn
3938                             <1> l_dac_p_5:
3939                             <1>       ; 29/07/2016
3940 0000267A FEC9               <1>       dec    cl
3941 0000267C 7407               <1>       jz     short l_dac_p_7
3942                             <1>       ;
3943 0000267E 28C0               <1>       sub    al, al ; 0
```

```
                        <1> l_dac_p_6:
00002680 EE             <1>         out    dx, al ; outb(VGAREG_DAC_DATA,0);
00002681 EE             <1>         out    dx, al
00002682 EE             <1>         out    dx, al
00002683 E2FB           <1>         loop   l_dac_p_6
                        <1> l_dac_p_7:
                        <1>         ;pop   esi
00002685 C3             <1>         retn
                        <1>
                        <1> gray_scale_summing:
                        <1>         ; 03/07/2016 (TRDOS 386 = TRDOS v2.0)
                        <1>         ; (set_mode_vga)
                        <1>         ; derived from 'Plex86/Bochs VGABios' source code
                        <1>         ; vgabios-0.7a (2011)
                        <1>         ; by the LGPL VGABios developers Team (2001-2008)
                        <1>         ; 'vgabios.c', 'biosfn_perform_gray_scale_summing'
                        <1>         ;
                        <1>         ; Oracle VirtualBox 5.0.24 VGABios Source Code
                        <1>         ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
                        <1>         ;
                        <1>
                        <1>         ; INPUT -> EBX = Start address (color index <= 255)
                        <1>         ;         ECX = Count (<= 256)
                        <1>         ; OUTPUT -> (E)CX = 0
                        <1>         ; (Modifed registers: EAX, ECX, EDX, EBX)
                        <1>
00002686 66BADA03       <1>         mov    dx, 3DAh ; VGAREG_ACTL_RESET
0000268A EC             <1>         in     al, dx
0000268B 30C0           <1>         xor    al, al ; 0
0000268D 66BAC003       <1>         mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
00002691 EE             <1>         out    dx, al ; clear bit 5
                        <1>                    ; (while loading palette registers)
                        <1>         ; set read address and switch to read mode
                        <1> g_s_s_1:
00002692 66BAC703       <1>         mov    dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
00002696 88D8           <1>         mov    al, bl
00002698 EE             <1>         out    dx, al
                        <1>         ; get 6-bit wide RGB data values
                        <1>         ; intensity = (0.3*Red)+(0.59*Green)+(0.11*Blue)
                        <1>         ; i = ( ( 77*r + 151*g + 28*b ) + 0x80 ) >> 8;
00002699 66BAC903       <1>         mov    dx, 3C9h ; VGAREG_DAC_DATA
0000269D EC             <1>         in     al, dx ; red
0000269E B44D           <1>         mov    ah, 77 ; 0.3* Red
000026A0 F6E4           <1>         mul    ah
000026A2 6650           <1>         push   ax
000026A4 EC             <1>         in     al, dx ; green
000026A5 B497           <1>         mov    ah, 151  ; 0.59 * Green
000026A7 F6E4           <1>         mul    ah
000026A9 6650           <1>         push   ax
000026AB EC             <1>         in     al, dx ; blue
000026AC B41C           <1>         mov    ah, 28 ; 0.11 * Blue
000026AE F6E4           <1>         mul    ah
000026B0 665A           <1>         pop    dx
000026B2 6601D0         <1>         add    ax, dx
000026B5 665A           <1>         pop    dx
000026B7 6601D0         <1>         add    ax, dx
000026BA 66058000       <1>         add    ax, 80h
000026BE B03F           <1>         mov    al, 3Fh
000026C0 38C4           <1>         cmp    ah, al
000026C2 7602           <1>         jna    short g_s_s_2
000026C4 88C4           <1>         mov    ah, al
                        <1> g_s_s_2:
000026C6 66BAC803       <1>         mov    dx, 3C8h  ; VGAREG_DAC_WRITE_ADDRESS
000026CA 88D8           <1>         mov    al, bl ; color index
000026CC EE             <1>         out    dx, al
000026CD 88E0           <1>         mov    al, ah ; intensity
000026CF 6642           <1>         inc    dx ; 3C9h ; VGAREG_DAC_DATA
000026D1 EE             <1>         out    dx, al ; R (R=G=B)
000026D2 88E0           <1>         mov    al, ah ; intensity
000026D4 EE             <1>         out    dx, al ; G (R=G=B)
000026D5 88E0           <1>         mov    al, ah ; intensity
000026D7 EE             <1>         out    dx, al ; B (R=G=B)
000026D8 6649           <1>         dec    cx
000026DA 7404           <1>         jz     short g_s_s_3
000026DC FEC3           <1>         inc    bl    ; next color index value
000026DE EBB2           <1>         jmp    short g_s_s_1
                        <1> g_s_s_3:
000026E0 66BADA03       <1>         mov    dx, 3DAh ; VGAREG_ACTL_RESET
000026E4 EC             <1>         in     al, dx
000026E5 B020           <1>         mov    al, 20h
000026E7 66BAC003       <1>         mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
000026EB EE             <1>         out    dx, al ; 20h -> set bit 5
                        <1>                    ; (after loading palette regs)
000026EC C3             <1>         retn
                        <1>
                        <1> vga_write_char_attr:
                        <1> vga_write_char_only:
                        <1>         ; 08/07/2016 (TRDOS 386 = TRDOS v2.0)
                        <1>         ;
                        <1>         ; derived from 'Plex86/Bochs VGABios' source code
                        <1>         ; vgabios-0.7a (2011)
                        <1>         ; by the LGPL VGABios developers Team (2001-2008)
                        <1>         ; 'vgabios.c', 'biosfn_write_char_attr'
                        <1>         ; 'biosfn_write_char_only'
                        <1>
                        <1>         ; INPUT ->
                        <1>         ; [CRT_MODE] = current video mode (>7)
                        <1>         ; CX = Count of characters to write
                        <1>         ; AL = Character to write
                        <1>         ; BL = Color of character
                        <1>         ; OUTPUT ->
                        <1>         ; Regen buffer updated
```

```
                                <1>
000026ED 8A25[C25E0000]         <1>       mov    ah, [CRT_MODE]
000026F3 668B15[1E520100]       <1>       mov    dx, [CURSOR_POSN] ; cursor pos for page 0
                                <1>
000026FA BE[DE5E0000]           <1>       mov    esi, vga_modes
000026FF 89F7                   <1>       mov    edi, esi
00002701 83C710                 <1>       add    edi, vga_mode_count
                                <1> vga_wca_0:
00002704 AC                     <1>       lodsb
00002705 38E0                   <1>       cmp    al, ah ; [CRT_MODE]
00002707 7405                   <1>       je     short vga_wca_2
00002709 39FE                   <1>       cmp    esi, edi
0000270B 72F7                   <1>       jb     short vga_wca_0
                                <1> vga_wca_1:
0000270D C3                     <1>       retn   ; nothing to do
                                <1> vga_wca_2:
0000270E 83C64F                 <1>       add    esi, vga_memmodel - (vga_modes + 1)
                                <1>       ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
                                <1>
                                <1>       ; biosfn_write_char_attr (car,page,attr,count)
                                <1>       ; AL = car, page = 0, BL = attr, CX = count
00002711 803E04                 <1>       cmp    byte [esi], PLANAR4
00002714 741D                   <1>       je     short vga_wca_planar
00002716 803E03                 <1>       cmp    byte [esi], PLANAR1
00002719 7418                   <1>       je     short vga_wca_planar
                                <1> vga_wca_linear8:
                                <1>       ; while((count-->0) && (xcurs<nbcols))
                                <1>       ; CX = count
0000271B 6621C9                 <1>       and    cx, cx
0000271E 74ED                   <1>       jz     short vga_wca_1
00002720 3A15[C45E0000]         <1>       cmp    dl, [CRT_COLS]
00002726 73E5                   <1>       jnb    short vga_wca_1
                                <1>       ; write_gfx_char_lin(car,attr,xcurs,ycurs,nbcols);
                                <1>       ; AL = car, BL = attr, DL = xcurs, DH = ycurs,
                                <1>       ; [CRT_COLS] = nbcols
00002728 E81E000000             <1>       call   write_gfx_char_lin
0000272D 6649                   <1>       dec    cx ; count
0000272F FEC2                   <1>       inc    dl ; xcurs
00002731 EBE8                   <1>       jmp    short vga_wca_linear8
                                <1> vga_wca_planar:
                                <1>       ; while((count-->0) && (xcurs<nbcols))
                                <1>       ; CX = count
00002733 6621C9                 <1>       and    cx, cx
00002736 74D5                   <1>       jz     short vga_wca_1
00002738 3A15[C45E0000]         <1>       cmp    dl, [CRT_COLS]
0000273E 73CD                   <1>       jnb    short vga_wca_1
                                <1>       ; write_gfx_char_pl4(car,attr,xcurs,ycurs,nbcols,cheight);
                                <1>       ; AL = car, BL = attr, DL = xcurs, DH = ycurs,
                                <1>       ; [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight
00002740 E89D000000             <1>       call   write_gfx_char_pl4
00002745 6649                   <1>       dec    cx ; count
00002747 FEC2                   <1>       inc    dl ; xcurs
00002749 EBE8                   <1>       jmp    short vga_wca_planar
                                <1>
                                <1> write_gfx_char_lin:
                                <1>       ; 08/08/2016
                                <1>       ; 31/07/2016
                                <1>       ; 08/07/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>       ;
                                <1>       ; derived from 'Plex86/Bochs VGABios' source code
                                <1>       ; vgabios-0.7a (2011)
                                <1>       ; by the LGPL VGABios developers Team (2001-2008)
                                <1>       ; 'vgabios.c', 'write_gfx_char_lin'
                                <1>
                                <1>       ; write_gfx_char_lin(car,attr,xcurs,ycurs,nbcols)
                                <1>       ; INPUT ->
                                <1>       ; AL = car, BL = attr, DL = xcurs, DH = ycurs,
                                <1>       ; [CRT_COLS] = nbcols
                                <1>       ; OUTPUT ->
                                <1>       ; Regen buffer updated
                                <1>
0000274B 51                     <1>       push   ecx
0000274C 53                     <1>       push   ebx
0000274D 52                     <1>       push   edx
0000274E 50                     <1>       push   eax
                                <1>       ; addr=xcurs*8+ycurs*nbcols*64;
                                <1>       ; 08/08/2016
0000274F 0FB6F0                 <1>       movzx  esi, al ; car
00002752 0FB6C6                 <1>       movzx  eax, dh ; ycurs
00002755 8A25[C45E0000]         <1>       mov    ah, [CRT_COLS] ; nbcols
0000275B F6E4                   <1>       mul    ah
                                <1>       ;shl   ax, 6 ; * 64
0000275D 66C1E003               <1>       shl    ax, 3 ; * 8
                                <1>       ;sub   dh, dh
                                <1>       ;shl   dx, 3 ; xcurs * 8
                                <1>       ;movzx edi, dx
00002761 0FB6FA                 <1>       movzx  edi, dl
00002764 66C1E703               <1>       shl    di, 3 ; xcurs * 8
00002768 30F6                   <1>       xor    dh, dh
0000276A 8A15[C65E0000]         <1>       mov    dl, [CHAR_HEIGHT]
00002770 66F7E2                 <1>       mul    dx
                                <1>       ; eax = ycurs*nbcols*8*[CHAR_HEIGHT]
00002773 01C7                   <1>       add    edi, eax ; addr
00002775 81C700000A00           <1>       add    edi, 0A0000h
                                <1>       ;shl   si, 3 ; car * 8
0000277B 30E4                   <1>       xor    ah, ah
0000277D A0[C65E0000]           <1>       mov    al, [CHAR_HEIGHT]
00002782 66F7E6                 <1>       mul    si
00002785 6689C6                 <1>       mov    si, ax
                                <1>       ;; esi = src = car * 8
                                <1>       ; esi = src = car * [CHAR_HEIGHT]
                                <1>       ; i = 0
```

```
                                <1>       ;add   esi, vgafont8 ; fdata [src+i]
                                <1>       ; 08/08/2016
00002788 A1[AE5F0100]           <1>       mov    eax, [VGA_INT43H]
0000278D 3D[7C3C0100]           <1>       cmp    eax, vgafont16
00002792 740F                   <1>         je      short wgfxl_0
00002794 3D[7C2E0100]           <1>       cmp    eax, vgafont14
00002799 7408                   <1>       je     short wgfxl_0
0000279B 81C6[7C260100]         <1>       add    esi, vgafont8
000027A1 EB02                   <1>       jmp    short wgfxl_1
                                <1> wgfxl_0:
000027A3 01C6                   <1>       add    esi, eax
                                <1> wgfxl_1:
000027A5 28FF                   <1>       sub    bh, bh ; i = 0
                                <1> wgfxl_2:
                                <1>       ; for(i=0;i<8;i++)
000027A7 57                     <1>       push   edi ; addr
000027A8 0FB605[C45E0000]       <1>       movzx  eax, byte [CRT_COLS] ; nbcols
000027AF F6E7                   <1>       mul    bh ; nbcols*i
000027B1 66C1E003               <1>       shl    ax, 3 ; i*nbcols*8
                                <1>       ; dest=addr+i*nbcols*8;
000027B5 01C7                   <1>       add    edi, eax ; dest + j ; j = 0
000027B7 B180                   <1>       mov    cl, 80h ; mask = 0x80;
                                <1>       ; esi = fdata + src + i
                                <1>       ; for(j=0;j<8;j++)
000027B9 29D2                   <1>       sub    edx, edx ; j = 0
                                <1> wgfxl_3:
000027BB 8A06                   <1>       mov    al, [esi] ; al = fdata[src+i]
000027BD 20C8                   <1>       and    al, cl ; if (fdata[src+i] & mask)
000027BF 7402                   <1>       jz     short wgfxl_4  ; data = 0, zf = 1
000027C1 88D8                   <1>       mov    al, bl ; data = attr;
                                <1> wgfxl_4:
                                <1>       ; write_byte(0xa000,dest+j,data);
000027C3 AA                     <1>       stosb  ; dest + j (+ 0A0000h)
                                <1>       ;inc   dl ; j++
                                <1>       ;cmp   dl, 8
000027C4 80FA07                 <1>       cmp    dl, 7
000027C7 720E                   <1>       jb     short wgfxl_5
000027C9 5F                     <1>       pop    edi
                                <1>       ; 08/08/2016
                                <1>       ;cmp   bh, 7
                                <1>       ;jnb   short wgfxl_6
000027CA FEC7                   <1>       inc    bh ; i++
000027CC 3A3D[C65E0000]         <1>       cmp    bh, [CHAR_HEIGHT]
000027D2 7309                   <1>       jnb    short wgfxl_6
000027D4 46                     <1>       inc    esi
000027D5 EBD0                   <1>       jmp    short wgfxl_2
                                <1> wgfxl_5:
000027D7 D0E9                   <1>       shr    cl, 1 ; mask >>= 1;
000027D9 FEC2                   <1>       inc    dl ; j++
000027DB EBDE                   <1>         jmp     short wgfxl_3
                                <1> wgfxl_6:
000027DD 58                     <1>       pop    eax
000027DE 5A                     <1>       pop    edx
000027DF 5B                     <1>       pop    ebx
000027E0 59                     <1>       pop    ecx
000027E1 C3                     <1>       retn
                                <1>
                                <1> write_gfx_char_pl4:
                                <1>       ; 08/08/2016
                                <1>       ; 08/07/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>       ;
                                <1>       ; derived from 'Plex86/Bochs VGABios' source code
                                <1>       ; vgabios-0.7a (2011)
                                <1>       ; by the LGPL VGABios developers Team (2001-2008)
                                <1>       ; 'vgabios.c', 'write_gfx_char_pl4'
                                <1>
                                <1>       ; write_gfx_char_pl4(car,attr,xcurs,ycurs,nbcols,cheight)
                                <1>       ; INPUT ->
                                <1>       ; AL = car, BL = attr, DL = xcurs, DH = ycurs,
                                <1>       ; [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight
                                <1>       ; OUTPUT ->
                                <1>       ; Regen buffer updated
                                <1>
000027E2 51                     <1>       push   ecx
000027E3 53                     <1>       push   ebx
000027E4 52                     <1>       push   edx
000027E5 50                     <1>       push   eax
                                <1> wgfxpl_f0:
                                <1>       ; switch(cheight)
000027E6 8A25[C65E0000]         <1>       mov    ah, [CHAR_HEIGHT]
000027EC 80FC10                 <1>       cmp    ah, 16 ; case 16:
000027EF 7507                   <1>       jne    short wgfxpl_f1
                                <1>       ; fdata = &vgafont16;
000027F1 BE[7C3C0100]           <1>       mov    esi, vgafont16
000027F6 EB13                   <1>       jmp    short wgfxpl_f3
                                <1> wgfxpl_f1:
000027F8 80FC0E                 <1>       cmp    ah, 14 ; case 14:
000027FB 7507                   <1>       jne    short wgfxpl_f2
000027FD BE[7C2E0100]           <1>       mov    esi, vgafont14
00002802 EB07                   <1>       jmp    short wgfxpl_f3
                                <1> wgfxpl_f2:
                                <1>       ; default:
                                <1>       ;  fdata = &vgafont8;
00002804 BE[7C260100]           <1>       mov    esi, vgafont8
00002809 B408                   <1>       mov    ah, 8
                                <1> wgfxpl_f3:
                                <1>       ; al = car
0000280B F6E4                   <1>       mul    ah ; ah = cheight
0000280D 25FFFF0000             <1>       and    eax, 0FFFFh ; car * cheight
                                <1>       ; src = car * cheight;
00002812 01C6                   <1>       add    esi, eax ; esi = fdata[src+i]
                                <1>       ; addr=xcurs*8+ycurs*nbcols*64;
```

```
4250 00002814 88F0              <1>       mov    al, dh ; ycurs
4251 00002816 8A25[C45E0000]    <1>       mov    ah, [CRT_COLS] ; nbcols
4252 0000281C F6E4              <1>       mul    ah
4253                            <1>       ; 08/08/2016
4254                            <1>       ;shl   ax, 6 ; * 64
4255 0000281E 66C1E003          <1>       shl    ax, 3 ; * 8
4256                            <1>       ;sub   dh, dh ; 0
4257                            <1>       ;shl   dx, 3 ; xcurs * 8
4258                            <1>       ;movzx edi, dx
4259 00002822 0FB6FA            <1>       movzx  edi, dl
4260 00002825 66C1E703          <1>       shl    di, 3 ; xcurs * 8
4261 00002829 30F6              <1>       xor    dh, dh
4262 0000282B 8A15[C65E0000]    <1>       mov    dl, [CHAR_HEIGHT]
4263 00002831 66F7E2            <1>       mul    dx
4264                            <1>       ; eax = ycurs*nbcols*8*[CHAR_HEIGHT]
4265 00002834 01C7              <1>       add    edi, eax ; addr
4266 00002836 81C700000A00      <1>       add    edi, 0A0000h
4267                            <1>       ;
4268                            <1>       ; outw(VGAREG_SEQU_ADDRESS, 0x0f02);
4269                            <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0205);
4270 0000283C 66BAC403          <1>       mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
4271 00002840 66B8020F          <1>       mov    ax, 0F02h
4272 00002844 66EF              <1>       out    dx, ax
4273 00002846 66BACE03          <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
4274 0000284A 66B80502          <1>       mov    ax, 0205h
4275 0000284E 66EF              <1>       out    dx, ax
4276                            <1>       ;
4277 00002850 66BACE03          <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
4278 00002854 F6C380            <1>       test   bl, 80h ; if(attr&0x80)
4279 00002857 7406              <1>       jz     short wgfxpl_f4 ; else
4280                            <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x1803);
4281 00002859 66B80318          <1>       mov    ax, 1803h
4282 0000285D EB04              <1>       jmp    short wgfxpl_f5
4283                            <1> wgfxpl_f4:
4284                            <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0003);
4285 0000285F 66B80300          <1>       mov    ax, 0003h
4286                            <1> wgfxpl_f5:
4287 00002863 66EF              <1>       out    dx, ax
4288                            <1>       ;
4289 00002865 28FF              <1>       sub    bh, bh ; i = 0
4290                            <1> wgfxpl_0:
4291                            <1>       ; for(i=0;i<cheight;i++)
4292 00002867 57                <1>       push   edi ; addr
4293 00002868 0FB605[C45E0000]  <1>       movzx  eax, byte [CRT_COLS] ; nbcols
4294 0000286F F6E7              <1>       mul    bh ; nbcols*i
4295                            <1>       ; dest=addr+i*nbcols
4296 00002871 01C7              <1>       add    edi, eax ; dest
4297 00002873 B580              <1>       mov    ch, 80h ; mask = 0x80;
4298                            <1>       ; for(j=0;j<8;j++)
4299 00002875 28C9              <1>       sub    cl, cl ; j = 0
4300                            <1> wgfxpl_1:
4301 00002877 D2ED              <1>       shr    ch, cl ; mask=0x80>>j;
4302                            <1>       ;
4303                            <1>       ; outw(VGAREG_GRDC_ADDRESS, (mask << 8) | 0x08);
4304                            <1>       ; read_byte(0xa000,dest);
4305                            <1>       ;mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
4306 00002879 88EC              <1>       mov    ah, ch
4307 0000287B B008              <1>       mov    al, 8
4308 0000287D 66EF              <1>       out    dx, ax
4309 0000287F 8A07              <1>       mov    al, [edi] ; ? (io delay?)
4310                            <1>       ;
4311 00002881 28C0              <1>       sub    al, al ; attr = 0
4312                            <1>       ; if (fdata[src+i] & mask)
4313 00002883 842E              <1>       test   byte [esi], ch
4314 00002885 7404              <1>       jz     short wgfxpl_2  ; zf = 1
4315                            <1>       ; write_byte(0xa000,dest,attr&0x0f);
4316 00002887 88D8              <1>       mov    al, bl ; attr;
4317 00002889 240F              <1>       and    al, 0Fh      ; attr&0x0f
4318                            <1> wgfxpl_2:
4319                            <1>       ; write_byte(0xa000,dest,0x00);
4320 0000288B 8807              <1>       mov    [edi], al ; dest (+ 0A0000h)
4321 0000288D FEC1              <1>       inc    cl ; j++
4322 0000288F 80F908            <1>       cmp    cl, 8
4323 00002892 72E3              <1>       jb     short wgfxpl_1
4324 00002894 5F                <1>       pop    edi
4325                            <1>       ; 08/08/2016
4326                            <1>       ;cmp   bh, 7
4327                            <1>       ;jnb   short wgfxpl_3
4328 00002895 FEC7              <1>       inc    bh ; i++
4329 00002897 3A3D[C65E0000]    <1>       cmp    bh, [CHAR_HEIGHT]
4330 0000289D 7303              <1>       jnb    short wgfxpl_3
4331 0000289F 46                <1>       inc    esi
4332 000028A0 EBC5              <1>       jmp    short wgfxpl_0
4333                            <1> wgfxpl_3:
4334                            <1>       ;mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
4335 000028A2 66B808FF          <1>       mov    ax, 0FF08h
4336 000028A6 66EF              <1>       out    dx, ax
4337 000028A8 66B80500          <1>       mov    ax, 0005h
4338 000028AC 66EF              <1>       out    dx, ax
4339 000028AE 66B80300          <1>       mov    ax, 0003h
4340 000028B2 66EF              <1>       out    dx, ax
4341                            <1>       ;
4342 000028B4 58                <1>       pop    eax
4343 000028B5 5A                <1>       pop    edx
4344 000028B6 5B                <1>       pop    ebx
4345 000028B7 59                <1>       pop    ecx
4346 000028B8 C3                <1>       retn
4347                            <1>
4348                            <1> vga_write_pixel:
4349                            <1>       ; 09/07/2016 (TRDOS 386 = TRDOS v2.0)
4350                            <1>       ;
4351                            <1>       ; derived from 'Plex86/Bochs VGABios' source code
```

```
4352                                    <1>       ; vgabios-0.7a (2011)
4353                                    <1>       ; by the LGPL VGABios developers Team (2001-2008)
4354                                    <1>       ; 'vgabios.c', 'biosfn_write_pixel'
4355                                    <1>
4356                                    <1>       ; INPUT ->
4357                                    <1>       ;      DX = row (0-239)
4358                                    <1>       ;      CX = column (0-799)
4359                                    <1>       ;      AL = pixel value
4360                                    <1>       ;      (AH = [CRT_MODE])
4361                                    <1>       ; OUTPUT ->
4362                                    <1>       ;      none
4363                                    <1>
4364 000028B9 88C3                     <1>       mov    bl, al ; pixel value
4365                                    <1>       ;mov   ah, [CRT_MODE]
4366 000028BB BE[DE5E0000]             <1>       mov    esi, vga_modes
4367 000028C0 89F7                     <1>       mov    edi, esi
4368 000028C2 83C710                   <1>       add    edi, vga_mode_count
4369                                    <1> vga_wp_0:
4370 000028C5 AC                       <1>       lodsb
4371 000028C6 38E0                     <1>       cmp    al, ah ; [CRT_MODE]
4372 000028C8 7405                     <1>       je     short vga_wp_1
4373 000028CA 39FE                     <1>       cmp    esi, edi
4374 000028CC 72F7                     <1>       jb     short vga_wp_0
4375 000028CE C3                       <1>       retn   ; nothing to do
4376                                    <1> vga_wp_1:
4377 000028CF 83C64F                   <1>       add    esi, vga_memmodel - (vga_modes + 1)
4378                                    <1>       ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
4379 000028D2 BF00000A00               <1>       mov    edi, 0A0000h
4380                                    <1>       ;
4381 000028D7 803E04                   <1>       cmp    byte [esi], PLANAR4
4382 000028DA 741D                     <1>       je     short vga_wp_planar
4383 000028DC 803E03                   <1>       cmp    byte [esi], PLANAR1
4384 000028DF 7418                     <1>       je     short vga_wp_planar
4385                                    <1> vga_wp_linear8:
4386                                    <1>       ; addr=CX+DX*(read_word(BIOSMEM_SEG,BIOSMEM_NB_COLS)*8);
4387 000028E1 0FB605[C45E0000]         <1>       movzx  eax, byte [CRT_COLS] ; = [VGA_COLS] ; nbcols
4388 000028E8 66C1E003                 <1>       shl    ax, 3 ; * 8
4389 000028EC 66F7E2                   <1>       mul    dx
4390 000028EF 50                       <1>       push   eax
4391                                    <1>       ;mov   edi, 0A0000h
4392 000028F0 6601CF                   <1>       add    di, cx
4393 000028F3 58                       <1>       pop    eax
4394 000028F4 01C7                     <1>       add    edi, eax ; addr
4395                                    <1>       ; write_byte(0xa000,addr,AL);
4396 000028F6 881F                     <1>       mov    [edi], bl
4397 000028F8 C3                       <1>       retn
4398                                    <1> vga_wp_planar:
4399                                    <1>       ; addr = CX/8+DX*read_word(BIOSMEM_SEG,BIOSMEM_NB_COLS);
4400 000028F9 0FB7C1                   <1>       movzx  eax, cx
4401 000028FC 66C1E803                 <1>       shr    ax, 3 ; CX/8
4402 00002900 50                       <1>       push   eax
4403 00002901 28E4                     <1>       sub    ah, ah ; 0
4404 00002903 A0[C45E0000]             <1>       mov    al, [CRT_COLS] ; = [VGA_COLS] ; nbcols
4405 00002908 66F7E2                   <1>       mul    dx
4406                                    <1>       ;mov   edi, 0A0000h
4407 0000290B 6601C7                   <1>         add     di, ax
4408 0000290E 58                       <1>       pop    eax
4409 0000290F 01C7                     <1>       add    edi, eax ; addr
4410 00002911 80E107                   <1>       and    cl, 7
4411 00002914 B580                     <1>       mov    ch, 80h ; mask
4412 00002916 D2ED                     <1>       shr    ch, cl       ; mask = 0x80 >> (CX & 0x07);
4413                                    <1>
4414                                    <1>       ; outw(VGAREG_GRDC_ADDRESS, (mask << 8) | 0x08);
4415 00002918 66BACE03                 <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
4416 0000291C 88EC                     <1>       mov    ah, ch
4417 0000291E B008                     <1>       mov    al, 8
4418 00002920 66EF                     <1>       out    dx, ax
4419                                    <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0205);
4420 00002922 66B80502                 <1>       mov    ax, 0205h
4421 00002926 66EF                     <1>       out    dx, ax
4422                                    <1>       ; data = read_byte(0xa000,addr);
4423 00002928 8A07                     <1>       mov    al, [edi] ; (delay?)
4424                                    <1>       ; if (AL & 0x80)
4425                                    <1>       ; {
4426                                    <1>       ;  outw(VGAREG_GRDC_ADDRESS, 0x1803);
4427                                    <1>       ; }
4428 0000292A F6C380                   <1>       test   bl, 80h
4429 0000292D 7406                     <1>       jz     short vga_wp_2
4430 0000292F 66B80318                 <1>       mov    ax, 1803h
4431 00002933 66EF                     <1>       out    dx, ax
4432                                    <1> vga_wp_2:
4433                                    <1>       ; write_byte(0xa000,addr,AL);
4434 00002935 881F                     <1>       mov    [edi], bl
4435                                    <1>       ;
4436                                    <1>       ;mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
4437 00002937 66B808FF                 <1>       mov    ax, 0FF08h
4438 0000293B 66EF                     <1>       out    dx, ax
4439 0000293D 66B80500                 <1>       mov    ax, 0005h
4440 00002941 66EF                     <1>       out    dx, ax
4441 00002943 66B80300                 <1>       mov    ax, 0003h
4442 00002947 66EF                     <1>       out    dx, ax
4443                                    <1>       ;
4444 00002949 C3                       <1>       retn
4445                                    <1>
4446                                    <1> vga_read_pixel:
4447                                    <1>       ; 09/07/2016 (TRDOS 386 = TRDOS v2.0)
4448                                    <1>       ;
4449                                    <1>       ; derived from 'Plex86/Bochs VGABios' source code
4450                                    <1>       ; vgabios-0.7a (2011)
4451                                    <1>       ; by the LGPL VGABios developers Team (2001-2008)
4452                                    <1>       ; 'vgabios.c', 'biosfn_read_pixel'
4453                                    <1>
```

```
                                <1>       ; INPUT ->
                                <1>       ;      DX = row (0-239)
                                <1>       ;      CX = column (0-799)
                                <1>       ;      (AH = [CRT_MODE])
                                <1>       ; OUTPUT ->
                                <1>       ;      AL = pixel value
                                <1>
                                <1>       ;mov   ah, [CRT_MODE]
0000294A BE[DE5E0000]           <1>       mov    esi, vga_modes
0000294F 89F7                  <1>       mov    edi, esi
00002951 83C710                <1>       add    edi, vga_mode_count
                                <1> vga_rp_0:
00002954 AC                    <1>       lodsb
00002955 38E0                  <1>       cmp    al, ah ; [CRT_MODE]
00002957 7405                  <1>       je     short vga_rp_1
00002959 39FE                  <1>       cmp    esi, edi
0000295B 72F7                  <1>       jb     short vga_rp_0
0000295D C3                    <1>       retn   ; nothing to do
                                <1> vga_rp_1:
0000295E 83C64F                <1>       add    esi, vga_memmodel - (vga_modes + 1)
                                <1>       ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
00002961 BF00000A00            <1>       mov    edi, 0A0000h
                                <1>       ;
00002966 803E04                <1>       cmp    byte [esi], PLANAR4
00002969 741D                  <1>       je     short vga_rp_planar
0000296B 803E03                <1>       cmp    byte [esi], PLANAR1
0000296E 7418                  <1>       je     short vga_rp_planar
                                <1> vga_rp_linear8:
                                <1>       ; addr=CX+DX*(read_word(BIOSMEM_SEG,BIOSMEM_NB_COLS)*8);
00002970 0FB605[C45E0000]      <1>       movzx  eax, byte [CRT_COLS] ; = [VGA_COLS] ; nbcols
00002977 66C1E003              <1>       shl    ax, 3 ; * 8
0000297B 66F7E2                <1>       mul    dx
0000297E 50                    <1>       push   eax
                                <1>       ;mov   edi, 0A0000h
0000297F 6601CF                <1>       add    di, cx
00002982 58                    <1>       pop    eax
00002983 01C7                  <1>       add    edi, eax ; addr
                                <1>       ; attr=read_byte(0xa000,addr);
00002985 8A07                  <1>       mov    al, [edi] ; pixel value
00002987 C3                    <1>       retn
                                <1> vga_rp_planar:
                                <1>       ; addr = CX/8+DX*read_word(BIOSMEM_SEG,BIOSMEM_NB_COLS);
00002988 0FB7C1                <1>       movzx  eax, cx
0000298B 66C1E803              <1>       shr    ax, 3 ; CX/8
0000298F 50                    <1>       push   eax
00002990 28E4                  <1>       sub    ah, ah ; 0
00002992 A0[C45E0000]          <1>       mov    al, [CRT_COLS] ; = [VGA_COLS] ; nbcols
00002997 66F7E2                <1>       mul    dx
                                <1>       ;mov   edi, 0A0000h
0000299A 6601C7                <1>         add     di, ax
0000299D 58                    <1>       pop    eax
0000299E 01C7                  <1>       add    edi, eax ; addr
000029A0 80E107                <1>       and    cl, 7
000029A3 B580                  <1>       mov    ch, 80h ; mask
000029A5 D2ED                  <1>       shr    ch, cl       ; mask = 0x80 >> (CX & 0x07);
                                <1>       ; attr = 0x00;
000029A7 30DB                  <1>       xor    bl, bl ; attr = bl = 0,
000029A9 30C9                  <1>       xor    cl, cl ; i = cl = 0
                                <1>       ; for(i=0;i<4;i++)
                                <1>             ; {
                                <1>             ;  outw(VGAREG_GRDC_ADDRESS, (i << 8) | 0x04);
                                <1>             ;  data = read_byte(0xa000,addr) & mask;
                                <1>             ;  if (data > 0) attr |= (0x01 << i);
                                <1>             ; }
                                <1> vga_rp_2:
000029AB 88CC                  <1>       mov    ah, cl ; i << 8
000029AD B004                  <1>       mov    al, 4  ; | 0x04
000029AF 66BACE03              <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
000029B3 66EF                  <1>       out    dx, ax
                                <1>       ; data = read_byte(0xa000,addr) & mask;
000029B5 8A07                  <1>       mov    al, [edi]
000029B7 20E8                  <1>       and    al, ch ; & mask
                                <1>       ; if (data > 0) attr |= (0x01 << i);
000029B9 08C0                  <1>       or     al, al
000029BB 7408                  <1>       jz     short vga_rp_3 ; al = 0
000029BD B701                  <1>       mov    bh, 1
000029BF D2E7                  <1>       shl    bh, cl ; (0x01 << i)
000029C1 08FB                  <1>       or     bl, bh ; attr |= (0x01 << i)
000029C3 88D8                  <1>       mov    al, bl ; pixel value
                                <1> vga_rp_3:
000029C5 C3                    <1>       retn
                                <1>
                                <1> vga_beeper:
                                <1>       ; 04/08/2016  (TRDOS 386 = TRDOS v2.0)
000029C6 FB                    <1>       sti
                                <1>       ;mov   bh, [ACTIVE_PAGE]
000029C7 E9CFF3FFFF            <1>         jmp     beeper_gfx
                                <1>
                                <1> vga_write_teletype:
                                <1>       ; 06/08/2016
                                <1>       ; 04/08/2016
                                <1>       ; 01/08/2016
                                <1>       ; 31/07/2016
                                <1>       ; 09/07/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>       ;
                                <1>       ; derived from 'Plex86/Bochs VGABios' source code
                                <1>       ; vgabios-0.7a (2011)
                                <1>       ; by the LGPL VGABios developers Team (2001-2008)
                                <1>       ; 'vgabios.c', 'biosfn_write_teletype'
                                <1>       ; 'biosfn_write_char_only'
                                <1>
                                <1>       ; INPUT ->
```

```
                                <1>       ; [CRT_MODE] = current video mode (>7)
                                <1>       ; AL = Character to write
                                <1>       ; BL = Color of character
                                <1>       ; OUTPUT ->
                                <1>       ; Regen buffer updated
                                <1>
                                <1>       ; biosfn_write_teletype (car, page, attr, flag)
                                <1>       ; car = character (AL)
                                <1>       ; page = 0
                                <1>       ; attr = color (BL)
                                <1>       ; 'flag' not used
                                <1>
000029CC 8A25[C25E0000]         <1>       mov    ah, [CRT_MODE]
000029D2 88C7                   <1>       mov    bh, al ; character
000029D4 668B15[1E520100]       <1>       mov    dx, [CURSOR_POSN] ; cursor pos for page 0
                                <1>
000029DB BE[E65E0000]           <1>       mov    esi, vga_g_modes
000029E0 89F7                   <1>       mov    edi, esi
000029E2 83C708                 <1>       add    edi, vga_g_mode_count
                                <1> vga_wtty_0:
000029E5 AC                     <1>       lodsb
000029E6 38E0                   <1>       cmp    al, ah ; [CRT_MODE]
000029E8 7405                   <1>       je     short vga_wtty_2
000029EA 39FE                   <1>       cmp    esi, edi
000029EC 72F7                   <1>       jb     short vga_wtty_0
                                <1> vga_wtty_1:
000029EE C3                     <1>       retn   ; nothing to do
                                <1> vga_wtty_2:
000029EF 80FF07                 <1>       cmp    bh, 07h ; bell (beep)
000029F2 74D2                   <1>       je     short vga_beeper  ; u11
000029F4 80FF08                 <1>       cmp    bh, 08h ; backspace
000029F7 7508                   <1>       jne    short vga_wtty_3
                                <1>       ; if(xcurs>0)xcurs--;
000029F9 08D2                   <1>       or     dl, dl ; xcurs (column)
000029FB 74F1                   <1>       jz     short vga_wtty_1
000029FD FECA                   <1>       dec    dl ; xcurs--;
000029FF EB59                   <1>         jmp     short vga_wtty_12
                                <1> vga_wtty_3:
00002A01 80FF0D                 <1>       cmp    bh, 0Dh ; carriage return (\r)
00002A04 7504                   <1>       jne    short vga_wtty_4
                                <1>       ; xcurs=0;
00002A06 28D2                   <1>       sub    dl, dl ; 0
00002A08 EB50                   <1>         jmp     short vga_wtty_12
                                <1> vga_wtty_4:
00002A0A 80FF0A                 <1>       cmp    bh, 0Ah ; new line (\n)
00002A0D 7504                   <1>       jne    short vga_wtty_5
                                <1>       ; ycurs++;
00002A0F FEC6                   <1>       inc    dh ; next row
00002A11 EB62                   <1>         jmp     short vga_wtty_11
                                <1> vga_wtty_5:
00002A13 80FF09                 <1>       cmp    bh, 09h ; tab stop
00002A16 7527                   <1>       jne    short vga_wtty_8
00002A18 88D0                   <1>       mov    al, dl
00002A1A 6698                   <1>       cbw
00002A1C B108                   <1>       mov    cl, 8
00002A1E F6F1                   <1>       div    cl
00002A20 28E1                   <1>       sub    cl, ah
                                <1>       ;
00002A22 B720                   <1>       mov    bh, 20h ; space
                                <1> vga_wtty_6: ; tab stop loop
00002A24 6651                   <1>       push   cx
00002A26 6653                   <1>       push   bx
00002A28 E812000000             <1>       call   vga_wtty_8
00002A2D 665B                   <1>       pop    bx  ; bh = character, bl = color
00002A2F 6659                   <1>       pop    cx
00002A31 FEC9                   <1>       dec    cl
00002A33 7409                   <1>       jz     short vga_wtty_7
00002A35 668B15[1E520100]       <1>       mov    dx, [CURSOR_POSN] ; new cursor position (pg 0)
00002A3C EBE6                   <1>       jmp    short vga_wtty_6
                                <1> vga_wtty_7:
00002A3E C3                     <1>       retn
                                <1>       ;
                                <1> vga_wtty_8:
00002A3F 83C64F                 <1>       add    esi, vga_g_memmodel - (vga_g_modes + 1)
                                <1>       ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
00002A42 BF00000A00             <1>       mov    edi, 0A0000h
                                <1>       ;
00002A47 88F8                   <1>       mov    al, bh ; character
                                <1>       ;
00002A49 803E04                 <1>       cmp    byte [esi], PLANAR4
00002A4C 7414                   <1>       je     short vga_wtty_planar
00002A4E 803E03                 <1>       cmp    byte [esi], PLANAR1
00002A51 740F                   <1>       je     short vga_wtty_planar
                                <1> vga_wtty_linear8:
                                <1>       ; write_gfx_char_lin(car,attr,xcurs,ycurs,nbcols);
                                <1>       ; AL = car, BL = attr (color), DL = xcurs, DH = ycurs,
                                <1>       ; [CRT_COLS] = nbcols
00002A53 E8F3FCFFFF             <1>       call   write_gfx_char_lin
00002A58 EB0D                   <1>       jmp    short vga_wtty_9
                                <1>
                                <1> vga_wtty_12:
                                <1>       ; 09/07/2016
                                <1>       ; set cursor position
                                <1>       ; NOTE: Hardware cursor position will not be set
                                <1>       ;   in any VGA modes (>7)
                                <1>       ;   But, cursor position will be saved into
                                <1>       ;   [CURSOR_POSN].
                                <1>       ;   TRDOS 386 (TRDOS v2.0) uses only one page
                                <1>       ;   (page 0) for all graphics modes.
                                <1>
00002A5A 668915[1E520100]       <1>       mov    [CURSOR_POSN], dx ; save cursor pos for pg 0
                                <1>       ; 04/08/2016
```

```
4658                                    <1>       ;mov   bh, [ACTIVE_PAGE] ; = 0
4659                                    <1>       ;call  _set_cpos
4660 00002A61 C3                       <1>       retn
4661                                    <1>
4662                                    <1> vga_wtty_planar:
4663                                    <1>       ; write_gfx_char_pl4(car,attr,xcurs,ycurs,nbcols,cheight);
4664                                    <1>       ; AL = car, BL = attr (color), DL = xcurs, DH = ycurs,
4665                                    <1>       ; [CRT_COLS]= nbcols, [CHAR_HEIGHT] = cheight
4666 00002A62 E87BFDFFFF                <1>       call   write_gfx_char_pl4
4667                                    <1> vga_wtty_9:
4668 00002A67 FEC2                     <1>       inc    dl ; xcurs++;
4669                                    <1> vga_wtty_10:
4670                                    <1>       ; Do we need to wrap ?
4671                                    <1>       ; if(xcurs==nbcols)
4672 00002A69 3A15[C45E0000]           <1>       cmp    dl, [CRT_COLS] ; [VGA_COLS]
4673 00002A6F 7204                     <1>       jb     short vga_wtty_11 ; no
4674 00002A71 28D2                     <1>       sub    dl, dl ; xcurs=0;
4675 00002A73 FEC6                     <1>       inc    dh ;  ycurs++;
4676                                    <1> vga_wtty_11:
4677                                    <1>       ; Do we need to scroll ?
4678                                    <1>       ; if(ycurs==nbrows)
4679 00002A75 3A35[CA5E0000]           <1>       cmp    dh, [VGA_ROWS]
4680 00002A7B 72DD                     <1>       jb     short vga_wtty_12 ; no
4681                                    <1>       ;
4682                                    <1>       ; biosfn_scroll (nblines,attr,rul,cul,rlr,clr,page,dir)
4683                                    <1>       ; al = nblines = 1, bl = attr (color) = 0
4684                                    <1>       ; ch = rul, cl = cul, dh = rlr, dl = clr, page = 0
4685                                    <1>       ; dir = SCROLL_UP
4686                                    <1>
4687 00002A7D B001                     <1>       mov    al, 1
4688 00002A7F 28DB                     <1>       sub    bl, bl ; 0 ; blank/black line (attr=0) will be used
4689 00002A81 6629C9                   <1>       sub    cx, cx ; 0,0
4690                                    <1>
4691                                    <1>       ; 06/08/2016
4692 00002A84 8A35[CA5E0000]           <1>       mov    dh, [VGA_ROWS]
4693 00002A8A FECE                     <1>       dec    dh ; nbrows -1
4694                                    <1>
4695 00002A8C 6652                     <1>       push   dx     ; 04/08/2016
4696 00002A8E 8A15[C45E0000]           <1>       mov    dl, [CRT_COLS]
4697 00002A94 FECA                     <1>       dec    dl ; nbcols -1
4698                                    <1>
4699 00002A96 8A25[C25E0000]           <1>       mov    ah, [CRT_MODE]
4700                                    <1>
4701                                    <1>       ; biosfn_scroll(0x01,0x00,0,0,nbrows-1,nbcols-1,page,SCROLL_UP);
4702 00002A9C E808F5FFFF               <1>       call   vga_graphics_up
4703                                    <1>       ; 04/08/2016
4704 00002AA1 665A                     <1>       pop    dx
4705                                    <1>       ;dec   dh ; ycurs-=1
4706 00002AA3 EBB5                     <1>       jmp    short vga_wtty_12
4707                                    <1>
4708                                    <1> font_setup:
4709                                    <1>       ; 09/07/2016
4710                                    <1>       ; character generator (font loading) functions
4711                                    <1>       ;
4712                                    <1>       ; derived from 'Plex86/Bochs VGABios' source code
4713                                    <1>       ; vgabios-0.7a (2011)
4714                                    <1>       ; by the LGPL VGABios developers Team (2001-2008)
4715                                    <1>       ; 'vgabios.c', 'int10_func'
4716                                    <1>
4717                                    <1>       ; AX = 1100H ; Load User-Defined Font (EGA/VGA)
4718                                    <1>       ;
4719                                    <1>         ; BH    height of each character (bytes per character definition)
4720                                    <1>         ; (BL   font block to load (EGA: 0-3; VGA: 0-7))
4721                                    <1>       ; CX    number of characters to redefine (<=256)
4722                                    <1>         ; DX    ASCII code of the first character defined at ES:BP
4723                                    <1>         ; EBP      address of font-definition information
4724                                    <1>       ;      (in user's memory space)
4725                                    <1>
4726                                    <1>       ; case 0x11:
4727                                    <1>       ; switch(GET_AL())
4728                                    <1>            ; {
4729                                    <1>       ; case 0x00:
4730                                    <1>         ; case 0x10:
4731                                    <1>         ; biosfn_load_text_user_pat(GET_AL(),ES,BP,CX,DX,GET_BL(),GET_BH());
4732                                    <1>         ; break;
4733                                    <1>
4734                                    <1>       ; AX = 1110H ; Load and Activate User-Defined Font (EGA/VGA)
4735 00002AA5 08C0                     <1>       or     al, al ; 0
4736 00002AA7 7404                     <1>       jz     short font_setup_0
4737 00002AA9 3C10                     <1>       cmp    al, 10h
4738 00002AAB 7511                     <1>       jne    short font_setup_1
4739                                    <1> font_setup_0:
4740 00002AAD E8B7000000               <1>       call   transfer_user_fonts
4741 00002AB2 721C                     <1>       jc     short font_setup_error
4742 00002AB4 E8C2000000               <1>       call   load_text_user_pat
4743 00002AB9 E996EAFFFF               <1>         jmp     VIDEO_RETURN
4744                                    <1> font_setup_1:
4745                                    <1>       ; AX = 1101H ; Load ROM 8x14 Character Set (EGA/VGA)
4746                                    <1>       ; case 0x01:
4747                                    <1>         ; case 0x11:
4748                                    <1>         ; biosfn_load_text_8_14_pat(GET_AL(),GET_BL());
4749                                    <1>         ; break;
4750 00002ABE 3C01                     <1>       cmp    al, 1
4751 00002AC0 7404                     <1>       je     short font_setup_2
4752 00002AC2 3C11                     <1>       cmp    al, 11h
4753 00002AC4 7511                     <1>       jne    short font_setup_3
4754                                    <1> font_setup_2:
4755                                    <1>       ; AX = 1111H ; Load and Activate ROM 8x14 Character Set (EGA/VGA)
4756                                    <1>       ; (BL = font block to load (EGA: 0-3; VGA: 0-7))
4757 00002AC6 E8EE010000               <1>       call   load_text_8_14_pat
4758 00002ACB E984EAFFFF               <1>         jmp     VIDEO_RETURN
4759                                    <1> font_setup_error:
```

```
4760 00002AD0 29C0              <1>       sub    eax, eax ; 0 -> fonts could not be loaded
4761 00002AD2 E982EAFFFF        <1>       jmp    _video_return
4762                            <1> font_setup_3:
4763                            <1>       ; AX = 1102H ; Load ROM 8x8 Character Set (EGA/VGA)
4764                            <1>       ; case 0x02:
4765                            <1>         ; case 0x12:
4766                            <1>         ; biosfn_load_text_8_8_pat(GET_AL(),GET_BL());
4767                            <1>         ; break;
4768 00002AD7 3C02              <1>       cmp    al, 2
4769 00002AD9 7404              <1>       je     short font_setup_4
4770 00002ADB 3C12              <1>       cmp    al, 12h
4771 00002ADD 750A              <1>       jne    short font_setup_5
4772                            <1> font_setup_4:
4773                            <1>       ; AX = 1112H ; Load and Activate ROM 8x8 Character Set (EGA/VGA)
4774                            <1>       ; (BL = font block to load (EGA: 0-3; VGA: 0-7))
4775 00002ADF E805020000        <1>       call   load_text_8_8_pat
4776 00002AE4 E96BEAFFFF        <1>         jmp     VIDEO_RETURN
4777                            <1> font_setup_5:
4778                            <1>       ; AX = 1104H ; Load ROM 8x16 Character Set (EGA/VGA)
4779                            <1>       ; case 0x04:
4780                            <1>         ; case 0x14:
4781                            <1>         ; biosfn_load_text_8_16_pat(GET_AL(),GET_BL());
4782                            <1>         ; break;
4783 00002AE9 3C04              <1>       cmp    al, 4
4784 00002AEB 7404              <1>       je     short font_setup_6
4785 00002AED 3C14              <1>       cmp    al, 14h
4786 00002AEF 750A              <1>       jne    short font_setup_7
4787                            <1> font_setup_6:
4788                            <1>       ; AX = 1114H ; Load and Activate ROM 8x16 Character Set (EGA/VGA)
4789                            <1>       ; (BL = font block to load (EGA: 0-3; VGA: 0-7))
4790 00002AF1 E823020000        <1>       call   load_text_8_16_pat
4791 00002AF6 E959EAFFFF        <1>         jmp     VIDEO_RETURN
4792                            <1> font_setup_7:
4793                            <1>       ; Note: AX=1120h (Setup INT 1Fh, EXT_PTR) is not needed
4794                            <1>       ; for TRDOS 386 (TRDIOS v2.0) video functionality;
4795                            <1>       ; because, originally EXT_PTR (font address) was used for
4796                            <1>       ; chars 80h to 0FFh (after the first 128 ASCII char fonts), for
4797                            <1>       ; CGA graphics mode; currenty, 'vgafont8' address has 256 chars!
4798                            <1>       ;
4799                            <1>       ; case 0x20:
4800                            <1>         ; biosfn_load_gfx_8_8_chars(ES,BP);
4801                            <1>         ; break;
4802                            <1>       ; case 0x21:
4803                            <1>         ; biosfn_load_gfx_user_chars(ES,BP,CX,GET_BL(),GET_DL());
4804                            <1>         ; break;
4805                            <1>       ; AX = 1121H ; Setup User-Defined Font for Graphics Mode (VGA)
4806                            <1>       ; BL   screen rows code: 00H = user-specified (in DL)
4807                            <1>         ;                      01H = 14 rows
4808                            <1>         ;                      02H = 25 rows
4809                            <1>         ;                      03H = 43 rows
4810                            <1>         ; CX   bytes per character definition
4811                            <1>         ; DL   (when BL=0) custom number of character rows on screen
4812                            <1>         ; EBP  address of font-definition information (user's mem space)
4813                            <1>
4814 00002AFB 3C21              <1>       cmp    al, 21h
4815 00002AFD 751A              <1>       jne    short font_setup_9
4816                            <1>
4817                            <1>       ; TRDOS 386 modification !
4818                            <1>       ; dh = 0 -> 256 characters
4819                            <1>       ; dh = 80h -> 128 characters
4820                            <1>       ; (If DH <> 0 and DH <> 80h -> invalid)
4821 00002AFF 20F6              <1>       and    dh, dh
4822 00002B01 7405              <1>       jz     short font_setup_8 ; 256 characters
4823 00002B03 80FE80            <1>       cmp    dh, 80h ; 128 characters
4824 00002B06 75C8              <1>       jne    short font_setup_error ; invalid !
4825                            <1> font_setup_8:
4826 00002B08 E85C000000        <1>       call   transfer_user_fonts
4827 00002B0D 72C1              <1>       jc     short font_setup_error
4828                            <1>       ; ebp = user's font data address in system's memory space
4829 00002B0F E836020000        <1>       call   load_gfx_user_chars
4830 00002B14 E93BEAFFFF        <1>         jmp     VIDEO_RETURN
4831                            <1> font_setup_9:
4832                            <1>       ; case 0x22:
4833                            <1>         ; biosfn_load_gfx_8_14_chars(GET_BL());
4834                            <1>         ; break;
4835 00002B19 3C22              <1>       cmp    al, 22h
4836 00002B1B 750A              <1>       jne    short font_setup_10
4837 00002B1D E866020000        <1>       call   load_gfx_8_14_chars
4838 00002B22 E92DEAFFFF        <1>         jmp     VIDEO_RETURN
4839                            <1> font_setup_10:
4840                            <1>       ; case 0x23:
4841                            <1>         ; biosfn_load_gfx_8_8_dd_chars(GET_BL());
4842                            <1>         ; break;
4843 00002B27 3C23              <1>       cmp    al, 23h
4844 00002B29 750A              <1>       jne    short font_setup_11
4845 00002B2B E899020000        <1>       call   load_gfx_8_8_chars
4846 00002B30 E91FEAFFFF        <1>         jmp     VIDEO_RETURN
4847                            <1> font_setup_11:
4848                            <1>       ; case 0x24:
4849                            <1>         ; biosfn_load_gfx_8_16_chars(GET_BL());
4850                            <1>         ; break;
4851 00002B35 3C24              <1>       cmp    al, 24h
4852 00002B37 750A              <1>       jne    short font_setup_12
4853 00002B39 E8CC020000        <1>       call   load_gfx_8_16_chars
4854 00002B3E E911EAFFFF        <1>         jmp     VIDEO_RETURN
4855                            <1> font_setup_12:
4856                            <1>       ; case 0x30:
4857                            <1>         ; biosfn_get_font_info(GET_BH(),&ES,&BP,&CX,&DX);
4858                            <1>         ; break;
4859 00002B43 3C30              <1>       cmp    al, 30h
4860 00002B45 750A              <1>       jne    short font_setup_13
4861 00002B47 E8FF020000        <1>       call   get_font_info
```

```
4862                                <1>       ; eax = return value (info: 4 bytes for 4 parms)
4863                                <1>       ; eax = 0 -> invalid function (input)
4864 00002B4C E908EAFFFF           <1>         jmp     _video_return
4865                                <1> font_setup_13:
4866 00002B51 3C03                 <1>       cmp    al, 03h ; AX = 1103h
4867 00002B53 750D                 <1>       jne    short font_setup_14
4868                                <1>       ; biosfn_set_text_block_specifier:
4869                                <1>       ; BL = font block selector code
4870                                <1>       ; NOTE: TRDOS 386 only uses and sets font block 0
4871                                <1>       ; (It is as BL = 0 for TRDOS 386)
4872 00002B55 66BAC403             <1>       mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
4873                                <1>       ;mov   ah, bl
4874 00002B59 28E4                 <1>       sub    ah, ah ; 0
4875                                <1>       ;mov   al, 03h
4876 00002B5B 66EF                 <1>       out    dx, ax
4877 00002B5D E9F2E9FFFF           <1>       jmp     VIDEO_RETURN
4878                                <1>
4879                                <1> font_setup_14:
4880 00002B62 29C0                 <1>       sub    eax, eax ; 0 = invalid function
4881 00002B64 E9F0E9FFFF           <1>         jmp     _video_return
4882                                <1>
4883                                <1> transfer_user_fonts:
4884                                <1>       ; 09/07/2016
4885                                <1>       ;and   ecx, 0FFFFh
4886                                <1>       ; ECX = byte count
4887                                <1>       ;push  ecx
4888 00002B69 89EE                 <1>       mov    esi, ebp ; user buffer
4889 00002B6B BF00000700           <1>       mov    edi, Cluster_Buffer  ; system buffer
4890 00002B70 E85BBD0000           <1>       call   transfer_from_user_buffer
4891                                <1>       ;pop   ecx
4892                                <1>       ; ecx = transfer (byte) count = character count
4893 00002B75 BD00000700           <1>       mov    ebp, Cluster_Buffer
4894                                <1>       ; jc   VIDEO_RETURN -> failed
4895 00002B7A C3                   <1>       retn
4896                                <1>
4897                                <1> load_text_user_pat:
4898                                <1>       ; 26/07/2016
4899                                <1>       ; 09/07/2016
4900                                <1>       ; load user defined (EGA/VGA) text fonts
4901                                <1>       ;
4902                                <1>       ; derived from 'Plex86/Bochs VGABios' source code
4903                                <1>       ; vgabios-0.7a (2011)
4904                                <1>       ; by the LGPL VGABios developers Team (2001-2008)
4905                                <1>       ; 'vgabios.c', 'biosfn_load_text_user_pat'
4906                                <1>
4907                                <1>       ; biosfn_load_text_user_pat (AL,ES,BP,CX,DX,BL,BH)
4908                                <1>
4909                                <1>       ; get_font_access();
4910                                <1>       ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
4911                                <1>       ; for(i=0;i<CX;i++)
4912                                <1>       ; {
4913                                <1>       ;  src = BP + i * BH;
4914                                <1>       ;  dest = blockaddr + (DX + i) * 32;
4915                                <1>       ;  memcpyb(0xA000, dest, ES, src, BH);
4916                                <1>       ; }
4917                                <1>       ; release_font_access();
4918                                <1>       ; if(AL>=0x10)
4919                                <1>       ; {
4920                                <1>       ; set_scan_lines(BH);
4921                                <1>       ; }
4922                                <1>
4923 00002B7B 50                   <1>       push   eax
4924 00002B7C E83C000000           <1>       call   get_font_access
4925 00002B81 28DB                 <1>       sub    bl, bl ; i = 0
4926                                <1> ltup_1:
4927 00002B83 88D8                 <1>       mov    al, bl
4928 00002B85 F6E7                 <1>       mul    bh
4929 00002B87 0FB7F0               <1>       movzx  esi, ax
4930 00002B8A 01EE                 <1>       add    esi, ebp
4931 00002B8C 88D8                 <1>       mov    al, bl
4932 00002B8E 28E4                 <1>       sub    ah, ah
4933 00002B90 6601D0               <1>       add    ax, dx ; (DX + i)
4934 00002B93 66C1E005             <1>       shl    ax, 5  ; * 32
4935 00002B97 0FB7F8               <1>       movzx  edi, ax
4936 00002B9A 81C700000A00         <1>       add    edi, 0A0000h
4937 00002BA0 51                   <1>       push   ecx
4938 00002BA1 0FB6CF               <1>       movzx  ecx, bh
4939 00002BA4 F3A4                 <1>       rep    movsb
4940 00002BA6 59                   <1>       pop    ecx
4941 00002BA7 FEC3                 <1>       inc    bl
4942 00002BA9 38CB                 <1>       cmp    bl, cl
4943 00002BAB 75D6                 <1>       jne    short ltup_1
4944                                <1>       ;
4945 00002BAD E840000000           <1>       call   release_font_access
4946                                <1>       ;
4947 00002BB2 58                   <1>       pop    eax
4948                                <1>       ; if(AL>=0x10)
4949 00002BB3 3C10                 <1>       cmp    al, 10h
4950 00002BB5 7205                 <1>       jb     short ltup_2
4951                                <1>       ; set_scan_lines(BH);
4952 00002BB7 E875000000           <1>       call   set_scan_lines
4953                                <1> ltup_2:
4954 00002BBC C3                   <1>       retn
4955                                <1>
4956                                <1> get_font_access:
4957                                <1>       ; 09/07/2016
4958                                <1>       ;
4959                                <1>       ; derived from 'Plex86/Bochs VGABios' source code
4960                                <1>       ; vgabios-0.7a (2011)
4961                                <1>       ; by the LGPL VGABios developers Team (2001-2008)
4962                                <1>       ; 'vgabios.c', 'get_font_access'
4963                                <1>
```

```
4964                                    <1>       ; get_font_access()
4965 00002BBD 52                       <1>       push   edx
4966 00002BBE 66BAC403                 <1>       mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
4967 00002BC2 66B80001                 <1>       mov    ax, 0100h
4968 00002BC6 66EF                     <1>       out    dx, ax
4969 00002BC8 66B80204                 <1>       mov    ax, 0402h
4970 00002BCC 66EF                     <1>       out    dx, ax
4971 00002BCE 66B80407                 <1>       mov    ax, 0704h
4972 00002BD2 66EF                     <1>       out    dx, ax
4973 00002BD4 66B80003                 <1>       mov    ax, 0300h
4974 00002BD8 66EF                     <1>       out    dx, ax
4975 00002BDA 66BACE03                 <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
4976 00002BDE 66B80402                 <1>       mov    ax, 0204h
4977 00002BE2 66EF                     <1>       out    dx, ax
4978 00002BE4 66B80500                 <1>       mov    ax, 0005h
4979 00002BE8 66EF                     <1>       out    dx, ax
4980 00002BEA 66B80604                 <1>       mov    ax, 0406h
4981 00002BEE 66EF                     <1>       out    dx, ax
4982 00002BF0 5A                       <1>       pop    edx
4983 00002BF1 C3                       <1>       retn
4984                                    <1>
4985                                    <1> release_font_access:
4986                                    <1>       ; 29/07/2016
4987                                    <1>       ; 09/07/2016
4988                                    <1>       ;
4989                                    <1>       ; derived from 'Plex86/Bochs VGABios' source code
4990                                    <1>       ; vgabios-0.7a (2011)
4991                                    <1>       ; by the LGPL VGABios developers Team (2001-2008)
4992                                    <1>       ; 'vgabios.c', 'release_font_access'
4993                                    <1>
4994 00002BF2 66BAC403                 <1>       mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
4995 00002BF6 66B80001                 <1>       mov    ax, 0100h
4996 00002BFA 66EF                     <1>       out    dx, ax
4997 00002BFC 66B80203                 <1>       mov    ax, 0302h
4998 00002C00 66EF                     <1>       out    dx, ax
4999 00002C02 66B80403                 <1>       mov    ax, 0304h
5000 00002C06 66EF                     <1>       out    dx, ax
5001 00002C08 66B80003                 <1>       mov    ax, 0300h
5002 00002C0C 66EF                     <1>       out    dx, ax
5003 00002C0E 66BACC03                 <1>       mov    dx, 3CCh ; VGAREG_READ_MISC_OUTPUT
5004 00002C12 EC                       <1>       in     al, dx
5005 00002C13 2401                     <1>       and    al, 01h
5006 00002C15 C0E002                   <1>       shl    al, 2
5007 00002C18 0C0A                     <1>       or     al, 0Ah
5008 00002C1A 88C4                     <1>       mov    ah, al
5009 00002C1C B006                     <1>       mov    al, 06h
5010 00002C1E 66BACE03                 <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
5011 00002C22 66EF                     <1>       out    dx, ax
5012 00002C24 66B80400                 <1>       mov    ax, 0004h
5013 00002C28 66EF                     <1>       out    dx, ax
5014 00002C2A 66B80510                 <1>       mov    ax, 1005h
5015 00002C2E 66EF                     <1>       out    dx, ax
5016 00002C30 C3                       <1>       retn
5017                                    <1>
5018                                    <1> set_scan_lines:
5019                                    <1>       ; 09/07/2016
5020                                    <1>       ;
5021                                    <1>       ; derived from 'Plex86/Bochs VGABios' source code
5022                                    <1>       ; vgabios-0.7a (2011)
5023                                    <1>       ; by the LGPL VGABios developers Team (2001-2008)
5024                                    <1>       ; 'vgabios.c', 'set_scan_lines'
5025                                    <1>
5026                                    <1>       ; set_scan_lines(lines)
5027                                    <1>       ; BH = lines
5028                                    <1>
5029                                    <1>       ; outb(crtc_addr, 0x09);
5030 00002C31 66BAD403                 <1>       mov    dx, 3D4h ; CRTC_ADDRESS = 3D4h (always)
5031 00002C35 B009                     <1>       mov    al, 09h
5032 00002C37 EE                       <1>       out    dx, al
5033                                    <1>       ; crtc_r9 = inb(crtc_addr+1);
5034 00002C38 6642                     <1>       inc    dx ; 3D5h
5035 00002C3A EC                       <1>       in     al, dx
5036                                    <1>       ; crtc_r9 = (crtc_r9 & 0xe0) | (lines - 1);
5037 00002C3B 24E0                     <1>       and    al, 0E0h
5038 00002C3D FECF                     <1>       dec    bh ; lines - 1
5039 00002C3F 08F8                     <1>       or     al, bh
5040                                    <1>       ; outb(crtc_addr+1, crtc_r9);
5041 00002C41 EE                       <1>       out    dx, al
5042                                    <1>       ;inc   bh
5043                                    <1>       ; if(lines==8)
5044                                    <1>       ;cmp   bh, 8
5045 00002C42 80FF07                   <1>       cmp    bh, 7
5046 00002C45 7506                     <1>       jne    short ssl_1
5047                                    <1>       ; biosfn_set_cursor_shape(0x06,0x07);
5048 00002C47 66B90706                 <1>       mov    cx, 0607h
5049 00002C4B EB06                     <1>       jmp    short ssl_2
5050                                    <1> ssl_1:
5051                                    <1>       ; biosfn_set_cursor_shape(lines-4,lines-3);
5052 00002C4D 88F9                     <1>       mov    cl, bh ; lines - 1
5053 00002C4F 88CD                     <1>       mov    ch, cl ; lines - 1 (16 -> 15)
5054 00002C51 FECD                     <1>       dec    ch  ; lines - 2 (16 -> 14)
5055                                    <1> ssl_2:
5056                                    <1>       ; CH = start line, CL = stop line
5057 00002C53 B40A                     <1>       mov    ah, 10 ; 6845 register for cursor set
5058 00002C55 66890D[DB5E0000]         <1>       mov    [CURSOR_MODE], cx ; save in data area
5059 00002C5C E812F1FFFF               <1>       call   m16    ; output cx register
5060                                    <1>       ; write_word(BIOSMEM_SEG,BIOSMEM_CHAR_HEIGHT, lines);
5061 00002C61 FEC7                     <1>       inc    bh ; lines
5062 00002C63 883D[C65E0000]           <1>       mov    [CHAR_HEIGHT], bh
5063                                    <1>       ;  outb(crtc_addr, 0x12);
5064 00002C69 66BAD403                 <1>       mov    dx, 3D4h ; CRTC_ADDRESS
5065 00002C6D B012                     <1>       mov    al, 12h
```

```
5066 00002C6F EE                <1>       out    dx, al
5067                            <1>       ; vde = inb(crtc_addr+1);
5068 00002C70 6642              <1>       inc    dx
5069 00002C72 EC                <1>       in     al, dx
5070 00002C73 88C4              <1>       mov    ah, al
5071                            <1>       ; outb(crtc_addr, 0x07);
5072 00002C75 664A              <1>       dec    dx
5073 00002C77 B007              <1>       mov    al, 07h
5074 00002C79 EE                <1>       out    dx, al
5075                            <1>       ; ovl = inb(crtc_addr+1);
5076 00002C7A 6642              <1>       inc    dx
5077 00002C7C EC                <1>       in     al, dx
5078                            <1>       ; vde += (((ovl & 0x02) << 7) + ((ovl & 0x40) << 3) + 1);
5079 00002C7D 88E2              <1>       mov    dl, ah ; vde
5080 00002C7F 88C6              <1>       mov    dh, al ; ovl
5081 00002C81 6683E002          <1>       and    ax, 02h
5082 00002C85 66C1E007          <1>       shl    ax, 7
5083 00002C89 6689C1            <1>       mov    cx, ax ; (ovl & 0x02) << 7)
5084 00002C8C 88F0              <1>       mov    al, dh ; ovl
5085 00002C8E 6683E040          <1>       and    ax, 40h
5086 00002C92 66C1E003          <1>       shl    ax, 3  ; (ovl & 0x40) << 3)
5087 00002C96 6640              <1>       inc    ax ; + 1
5088 00002C98 6601C8            <1>       add    ax, cx
5089 00002C9B 30F6              <1>       xor    dh, dh
5090 00002C9D 6601D0            <1>       add    ax, dx ; + vde
5091                            <1>       ; rows = vde / lines;
5092 00002CA0 F6F7              <1>       div    bh
5093                            <1>       ;dec   al ; rows -1
5094                            <1>       ; write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, rows-1);
5095 00002CA2 A2[CA5E0000]      <1>       mov    [VGA_ROWS], al ; rows (not 'rows-1' !)
5096                            <1>       ; write_word(BIOSMEM_SEG,BIOSMEM_PAGE_SIZE, rows * cols * 2);
5097 00002CA7 8A25[C45E0000]    <1>       mov    ah, [CRT_COLS]
5098 00002CAD F6E4              <1>       mul    ah
5099 00002CAF 66D1E0            <1>       shl    ax, 1
5100 00002CB2 66A3[9C5F0100]    <1>       mov    [CRT_LEN], ax
5101 00002CB8 C3                <1>       retn
5102                            <1>
5103                            <1> load_text_8_14_pat:
5104                            <1>       ; 26/07/2016
5105                            <1>       ; 25/07/2016
5106                            <1>       ; 23/07/2016
5107                            <1>       ; 09/07/2016
5108                            <1>       ; load user defined (EGA/VGA) text fonts
5109                            <1>       ;
5110                            <1>       ; derived from 'Plex86/Bochs VGABios' source code
5111                            <1>       ; vgabios-0.7a (2011)
5112                            <1>       ; by the LGPL VGABios developers Team (2001-2008)
5113                            <1>       ; 'vgabios.c', 'biosfn_load_text_8_14_pat'
5114                            <1>
5115                            <1>       ; biosfn_load_text_8_14_pat (AL,BL)
5116                            <1>
5117                            <1>       ; get_font_access();
5118                            <1>       ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
5119                            <1>       ; for(i=0;i<0x100;i++)
5120                            <1>       ; {
5121                            <1>       ;  src = i * 14;
5122                            <1>       ;  dest = blockaddr + i * 32;
5123                            <1>       ;  memcpyb(0xA000, dest, 0xC000, vgafont14+src, 14);
5124                            <1>       ; }
5125                            <1>       ; release_font_access();
5126                            <1>       ; if(AL>=0x10)
5127                            <1>       ; {
5128                            <1>       ; set_scan_lines(14);
5129                            <1>       ; }
5130                            <1>
5131 00002CB9 50                <1>       push   eax
5132 00002CBA E8FEFEFFFF        <1>       call   get_font_access
5133                            <1>
5134                            <1>       ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
5135                            <1>       ;mov   dl, bl
5136                            <1>       ;and   dl, 3
5137                            <1>       ;shl   dx, 14
5138                            <1>       ;xchg  dx, bx
5139                            <1>       ;and   dl, 4
5140                            <1>       ;shl   dx, 11
5141                            <1>       ;add   dx, bx
5142                            <1>
5143                            <1>       ;xor   dx, dx  ; blockaddr = 0
5144                            <1>       ; Always block 0 for TRDOS 386 ! (blockaddr=0(
5145                            <1>
5146 00002CBF 28DB              <1>       sub    bl, bl ; i = 0
5147 00002CC1 B70E              <1>       mov    bh, 14
5148 00002CC3 BE[7C2E0100]      <1>       mov    esi, vgafont14
5149 00002CC8 BF00000A00        <1>       mov    edi, 0A0000h
5150                            <1> lt8_14_1:
5151                            <1>       ;mov   al, bl
5152                            <1>       ;mul   bh
5153                            <1>       ;movzx esi, ax
5154                            <1>       ;add   esi, vgafont14
5155                            <1>       ;mov   al, bl
5156                            <1>       ;sub   ah, ah
5157                            <1>       ;shl   ax, 5 ; * 32
5158                            <1>       ;;add  ax, dx ; blockaddr + i * 32;
5159                            <1>       ;movzx edi, ax ; dest
5160                            <1>       ;add   edi, 0A0000h
5161 00002CCD 0FB6CF            <1>       movzx  ecx, bh
5162 00002CD0 F3A4              <1>       rep    movsb
5163 00002CD2 83C712            <1>       add    edi, 18 ; 32 - 14
5164 00002CD5 FEC3              <1>       inc    bl
5165 00002CD7 75F4              <1>       jnz    short lt8_14_1
5166                            <1>       ;
5167 00002CD9 E814FFFFFF        <1>       call   release_font_access
```

```
                                <1>       ;
00002CDE 58                     <1>       pop    eax
                                <1>       ; if(AL>=0x10)
00002CDF 3C10                   <1>       cmp    al, 10h
00002CE1 7205                   <1>       jb     short lt8_14_4
                                <1>       ; BH = 14
                                <1>       ; set_scan_lines(14);
00002CE3 E849FFFFFF             <1>       call   set_scan_lines
                                <1> lt8_14_4:
00002CE8 C3                     <1>       retn
                                <1>
                                <1> load_text_8_8_pat:
                                <1>       ; 26/07/2016
                                <1>       ; 25/07/2016
                                <1>       ; 23/07/2016
                                <1>       ; 09/07/2016
                                <1>       ; load user defined (EGA/VGA) text fonts
                                <1>       ;
                                <1>       ; derived from 'Plex86/Bochs VGABios' source code
                                <1>       ; vgabios-0.7a (2011)
                                <1>       ; by the LGPL VGABios developers Team (2001-2008)
                                <1>       ; 'vgabios.c', 'biosfn_load_text_8_8_pat'
                                <1>
                                <1>       ; biosfn_load_text_8_8_pat (AL,BL)
                                <1>
                                <1>       ; get_font_access();
                                <1>       ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
                                <1>       ; for(i=0;i<0x100;i++)
                                <1>       ; {
                                <1>       ;  src = i * 8;
                                <1>       ;  dest = blockaddr + i * 32;
                                <1>       ;  memcpyb(0xA000, dest, 0xC000, vgafont8+src, 8);
                                <1>       ; }
                                <1>       ; release_font_access();
                                <1>       ; if(AL>=0x10)
                                <1>       ; {
                                <1>       ; set_scan_lines(8);
                                <1>       ; }
                                <1>
00002CE9 50                     <1>       push   eax
00002CEA E8CEFEFFFF             <1>       call   get_font_access
                                <1>
                                <1>       ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
                                <1>       ;mov   dl, bl
                                <1>       ;and   dl, 3
                                <1>       ;shl   dx, 14
                                <1>       ;xchg  dx, bx
                                <1>       ;and   dl, 4
                                <1>       ;shl   dx, 11
                                <1>       ;add   dx, bx
                                <1>
                                <1>       ;xor   dx, dx  ; blockaddr = 0
                                <1>       ; Always block 0 for TRDOS 386 ! (blockaddr=0(
                                <1>
00002CEF 28DB                   <1>       sub    bl, bl ; i = 0
00002CF1 B708                   <1>       mov    bh, 8
00002CF3 BE[7C260100]           <1>       mov    esi, vgafont8
00002CF8 BF00000A00             <1>       mov    edi, 0A0000h
                                <1> lt8_8_1:
                                <1>       ;mov   al, bl
                                <1>       ;mul   bh
                                <1>       ;movzx esi, ax
                                <1>       ;add   esi, vgafont8
                                <1>       ;mov   al, bl
                                <1>       ;sub   ah, ah
                                <1>       ;shl   ax, 5 ; * 32
                                <1>       ;;add  ax, dx ; blockaddr + i * 32;
                                <1>       ;movzx edi, ax ; dest
                                <1>       ;add   edi, 0A0000h
00002CFD 0FB6CF                 <1>       movzx  ecx, bh
00002D00 F3A4                   <1>       rep    movsb
00002D02 83C718                 <1>       add    edi, 24 ; 32 - 8
00002D05 FEC3                   <1>       inc    bl
00002D07 75F4                   <1>       jnz    short lt8_8_1
                                <1>       ;
00002D09 E8E4FEFFFF             <1>       call   release_font_access
                                <1>       ;
00002D0E 58                     <1>       pop    eax
                                <1>       ; if(AL>=0x10)
00002D0F 3C10                   <1>       cmp    al, 10h
00002D11 7205                   <1>       jb     short lt8_8_2
                                <1>       ; BH = 8
                                <1>       ; set_scan_lines(8);
00002D13 E819FFFFFF             <1>       call   set_scan_lines
                                <1> lt8_8_2:
00002D18 C3                     <1>       retn
                                <1>
                                <1> load_text_8_16_pat:
                                <1>       ; 26/07/2016
                                <1>       ; 25/07/2016
                                <1>       ; 23/07/2016
                                <1>       ; 09/07/2016
                                <1>       ; load user defined (EGA/VGA) text fonts
                                <1>       ;
                                <1>       ; derived from 'Plex86/Bochs VGABios' source code
                                <1>       ; vgabios-0.7a (2011)
                                <1>       ; by the LGPL VGABios developers Team (2001-2008)
                                <1>       ; 'vgabios.c', 'biosfn_load_text_8_16_pat'
                                <1>
                                <1>       ; biosfn_load_text_8_16_pat (AL,BL)
                                <1>
                                <1>       ; get_font_access();
```

```
                                <1>       ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
                                <1>       ; for(i=0;i<0x100;i++)
                                <1>       ; {
                                <1>       ;  src = i * 16;
                                <1>       ;  dest = blockaddr + i * 32;
                                <1>       ;  memcpyb(0xA000, dest, 0xC000, vgafont16+src, 16);
                                <1>       ; }
                                <1>       ; release_font_access();
                                <1>       ; if(AL>=0x10)
                                <1>       ; {
                                <1>       ; set_scan_lines(16);
                                <1>       ; }
                                <1>
00002D19 50                     <1>       push   eax
00002D1A E89EFEFFFF             <1>       call   get_font_access
                                <1>
                                <1>       ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
                                <1>       ;mov   dl, bl
                                <1>       ;and   dl, 3
                                <1>       ;shl   dx, 14
                                <1>       ;xchg  dx, bx
                                <1>       ;and   dl, 4
                                <1>       ;shl   dx, 11
                                <1>       ;add   dx, bx
                                <1>
                                <1>       ;xor   dx, dx  ; blockaddr = 0
                                <1>       ; Always block 0 for TRDOS 386 ! (blockaddr=0(
                                <1>
00002D1F 28DB                   <1>       sub    bl, bl ; i = 0
00002D21 B710                   <1>       mov    bh, 16
00002D23 BE[7C3C0100]           <1>       mov    esi, vgafont16
00002D28 BF00000A00             <1>       mov    edi, 0A0000h
00002D2D 0FB6C7                 <1>       movzx  eax, bh
                                <1> lt8_16_1:
                                <1>       ;mov   al, bl
                                <1>       ;mul   bh
                                <1>       ;movzx esi, ax
                                <1>       ;add   esi, vgafont16
                                <1>       ;mov   al, bl ; i
                                <1>       ;sub   ah, ah
                                <1>       ;shl   ax, 5 ; * 32
                                <1>       ;;add  ax, dx ; blockaddr + i * 32;
                                <1>       ;movzx edi, ax ; dest
                                <1>       ;add   edi, 0A0000h
                                <1>       ;movzx ecx, bh
00002D30 89C1                   <1>       mov    ecx, eax ; 16
00002D32 F3A4                   <1>       rep    movsb
00002D34 01C7                   <1>       add    edi, eax ; add edi, 16
00002D36 FEC3                   <1>       inc    bl
00002D38 75F6                   <1>       jnz    short lt8_16_1
                                <1>       ;
00002D3A E8B3FEFFFF             <1>       call   release_font_access
                                <1>       ;
00002D3F 58                     <1>       pop    eax
                                <1>       ; if(AL>=0x10)
00002D40 3C10                   <1>       cmp    al, 10h
00002D42 7205                   <1>       jb     short lt8_16_2
                                <1>       ; BH = 16
                                <1>       ; set_scan_lines(16);
00002D44 E8E8FEFFFF             <1>       call   set_scan_lines
                                <1> lt8_16_2:
00002D49 C3                     <1>       retn
                                <1>
                                <1> load_gfx_user_chars:
                                <1>       ; 08/08/2016
                                <1>       ; 10/07/2016
                                <1>       ; Setup User-Defined Font for Graphics Mode (VGA)
                                <1>       ;
                                <1>       ; derived from 'Plex86/Bochs VGABios' source code
                                <1>       ; vgabios-0.7a (2011)
                                <1>       ; by the LGPL VGABios developers Team (2001-2008)
                                <1>       ; 'vgabios.c', 'biosfn_load_gfx_user_chars'
                                <1>
                                <1>       ; biosfn_load_gfx_user_chars (ES,BP,CX,BL,DL)
                                <1>       ; /* set 0x43 INT pointer */
                                <1>       ; write_word(0x0, 0x43*4, BP);
                                <1>       ; write_word(0x0, 0x43*4+2, ES);
00002D4A 31C0                   <1>       xor    eax, eax
00002D4C 48                     <1>       dec    eax ; 0FFFFFFFFh (user defined fonts)
00002D4D A3[AE5F0100]           <1>       mov    [VGA_INT43H], eax
                                <1>
                                <1>       ; BL   screen rows code: 00H = user-specified (in DL)
                                <1>         ;                      01H = 14 rows
                                <1>         ;                      02H = 25 rows
                                <1>         ;                      03H = 43 rows
                                <1>         ; CX   bytes per character definition
                                <1>         ; DL   (when BL=0) custom number of character rows on screen
                                <1>       ; dh = 0 -> 256 characters
                                <1>       ; dh = 80h -> 128 characters
                                <1>       ; (If DH <> 0 and DH <> 80h -> invalid)
                                <1>         ; EBP  address of font-definition information (user's mem space)
                                <1>
                                <1>       ; switch (BL) {
                                <1>       ; case 0:
                                <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, DL-1);
                                <1>       ;    break;
00002D52 20DB                   <1>       and    bl, bl
00002D54 7508                   <1>       jnz    short l_gfx_uc_1
00002D56 8815[CA5E0000]         <1>       mov    [VGA_ROWS], dl  ; not DL-1 !
00002D5C EB23                   <1>       jmp    short l_gfx_uc_4
                                <1> l_gfx_uc_1:
                                <1>       ; case 1:
```

```
                                <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 13);
                                <1>       ;    break;
00002D5E FECB                   <1>       dec    bl
00002D60 7509                   <1>       jnz    short l_gfx_uc_2
                                <1>       ; bl = 1
00002D62 C605[CA5E0000]0E       <1>       mov    byte [VGA_ROWS], 14  ; not 13 !
00002D69 EB16                   <1>       jmp    short l_gfx_uc_4
                                <1> l_gfx_uc_2:
00002D6B FECB                   <1>       dec    bl
00002D6D 740B                   <1>       jz     short l_gfx_uc_3 ; bl = 2
00002D6F FECB                   <1>       dec    bl
00002D71 750E                   <1>       jnz    short l_gfx_uc_4 ; bl > 3
                                <1>       ; bl = 3
                                <1>       ; case 3:
                                <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 42);
                                <1>       ;    break;
00002D73 C605[CA5E0000]2B       <1>       mov    byte [VGA_ROWS], 43  ; not 42 !
                                <1> l_gfx_uc_3:
                                <1>       ; case 2:
                                <1>       ; default:
                                <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 24);
                                <1>       ;    break;
                                <1>       ; bl = 2 or bl > 3
00002D7A C605[CA5E0000]19       <1>       mov    byte [VGA_ROWS], 25  ; not 24 !
                                <1>       ; }
                                <1> l_gfx_uc_4:
                                <1>       ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, CX);
00002D81 880D[C65E0000]         <1>       mov    [CHAR_HEIGHT], cl
                                <1>       ; }
00002D87 C3                     <1>       retn
                                <1>
                                <1> load_gfx_8_14_chars:
                                <1>       ; 08/08/2016
                                <1>       ; 10/07/2016
                                <1>       ; Setup ROM 8x14 Font for Graphics Mode (VGA)
                                <1>       ;
                                <1>       ; derived from 'Plex86/Bochs VGABios' source code
                                <1>       ; vgabios-0.7a (2011)
                                <1>       ; by the LGPL VGABios developers Team (2001-2008)
                                <1>       ; 'vgabios.c', 'biosfn_load_gfx_8_14_chars'
                                <1>
                                <1>       ; biosfn_load_gfx_8_14_chars (BL)
                                <1>       ; /* set 0x43 INT pointer */
                                <1>       ; write_word(0x0, 0x43*4, &vgafont14);
                                <1>       ; write_word(0x0, 0x43*4+2, 0xC000);
00002D88 C705[AE5F0100]-        <1>       mov    dword [VGA_INT43H], vgafont14
00002D8E [7C2E0100]             <1>
                                <1>
                                <1>       ; BL    screen rows code: 00H = user-specified (in DL)
                                <1>         ;                       01H = 14 rows
                                <1>         ;                       02H = 25 rows
                                <1>         ;                       03H = 43 rows
                                <1>         ; DL    (when BL=0) custom number of char rows on screen
                                <1>
                                <1>       ; switch (BL) {
                                <1>       ; case 0:
                                <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, DL-1);
                                <1>       ;    break;
00002D92 20DB                   <1>       and    bl, bl
00002D94 7508                   <1>       jnz    short l_gfx_8_14c_1
00002D96 8815[CA5E0000]         <1>       mov    [VGA_ROWS], dl  ; not DL-1 !
00002D9C EB23                   <1>       jmp    short l_gfx_8_14c_4
                                <1> l_gfx_8_14c_1:
                                <1>       ; case 1:
                                <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 13);
                                <1>       ;    break;
00002D9E FECB                   <1>       dec    bl
00002DA0 7509                   <1>       jnz    short l_gfx_8_14c_2
                                <1>       ; bl = 1
00002DA2 C605[CA5E0000]0E       <1>       mov    byte [VGA_ROWS], 14  ; not 13 !
00002DA9 EB16                   <1>       jmp    short l_gfx_8_14c_4
                                <1> l_gfx_8_14c_2:
00002DAB FECB                   <1>       dec    bl
00002DAD 740B                   <1>       jz     short l_gfx_8_14c_3 ; bl = 2
00002DAF FECB                   <1>       dec    bl
00002DB1 750E                   <1>       jnz    short l_gfx_8_14c_4 ; bl > 3
                                <1>       ; bl = 3
                                <1>       ; case 3:
                                <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 42);
                                <1>       ;    break;
00002DB3 C605[CA5E0000]2B       <1>       mov    byte [VGA_ROWS], 43  ; not 42 !
                                <1> l_gfx_8_14c_3:
                                <1>       ; case 2:
                                <1>       ; default:
                                <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 24);
                                <1>       ;    break;
                                <1>       ; bl = 2 or bl > 3
00002DBA C605[CA5E0000]19       <1>       mov    byte [VGA_ROWS], 25  ; not 24 !
                                <1>       ; }
                                <1> l_gfx_8_14c_4:
                                <1>       ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, 14);
00002DC1 C605[C65E0000]0E       <1>         mov     byte [CHAR_HEIGHT], 14
                                <1>       ; }
00002DC8 C3                     <1>       retn
                                <1>
                                <1> load_gfx_8_8_chars:
                                <1>       ; 08/08/2016
                                <1>       ; 10/07/2016
                                <1>       ; Setup ROM 8x14 Font for Graphics Mode (VGA)
                                <1>       ;
                                <1>       ; derived from 'Plex86/Bochs VGABios' source code
                                <1>       ; vgabios-0.7a (2011)
```

```
                                <1>       ; by the LGPL VGABios developers Team (2001-2008)
                                <1>       ; 'vgabios.c', 'biosfn_load_gfx_8_8_dd_chars'
                                <1>
                                <1>       ; biosfn_load_gfx_8_8_dd_chars (BL)
                                <1>       ; /* set 0x43 INT pointer */
                                <1>       ; write_word(0x0, 0x43*4, &vgafont8);
                                <1>       ; write_word(0x0, 0x43*4+2, 0xC000);
00002DC9 C705[AE5F0100]-        <1>       mov   dword [VGA_INT43H], vgafont8
00002DCF [7C260100]             <1>
                                <1>
                                <1>       ; BL    screen rows code: 00H = user-specified (in DL)
                                <1>         ;                       01H = 14 rows
                                <1>         ;                       02H = 25 rows
                                <1>         ;                       03H = 43 rows
                                <1>         ; DL    (when BL=0) custom number of char rows on screen
                                <1>
                                <1>       ; switch (BL) {
                                <1>       ; case 0:
                                <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, DL-1);
                                <1>       ;    break;
00002DD3 20DB                   <1>       and   bl, bl
00002DD5 7508                   <1>       jnz   short l_gfx_8_8c_1
00002DD7 8815[CA5E0000]         <1>       mov   [VGA_ROWS], dl  ; not DL-1 !
00002DDD EB23                   <1>       jmp   short l_gfx_8_8c_4
                                <1> l_gfx_8_8c_1:
                                <1>       ; case 1:
                                <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 13);
                                <1>       ;    break;
00002DDF FECB                   <1>       dec   bl
00002DE1 7509                   <1>       jnz   short l_gfx_8_8c_2
                                <1>       ; bl = 1
00002DE3 C605[CA5E0000]0E       <1>       mov   byte [VGA_ROWS], 14  ; not 13 !
00002DEA EB16                   <1>       jmp   short l_gfx_8_8c_4
                                <1> l_gfx_8_8c_2:
00002DEC FECB                   <1>       dec   bl
00002DEE 740B                   <1>       jz    short l_gfx_8_8c_3 ; bl = 2
00002DF0 FECB                   <1>       dec   bl
00002DF2 750E                   <1>       jnz   short l_gfx_8_8c_4 ; bl > 3
                                <1>       ; bl = 3
                                <1>       ; case 3:
                                <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 42);
                                <1>       ;    break;
00002DF4 C605[CA5E0000]2B       <1>       mov   byte [VGA_ROWS], 43  ; not 42 !
                                <1> l_gfx_8_8c_3:
                                <1>       ; case 2:
                                <1>       ; default:
                                <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 24);
                                <1>       ;    break;
                                <1>       ; bl = 2 or bl > 3
00002DFB C605[CA5E0000]19       <1>       mov   byte [VGA_ROWS], 25  ; not 24 !
                                <1>       ; }
                                <1> l_gfx_8_8c_4:
                                <1>       ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, 8);
00002E02 C605[C65E0000]08       <1>         mov     byte [CHAR_HEIGHT], 8
                                <1>       ; }
00002E09 C3                     <1>       retn
                                <1>
                                <1> load_gfx_8_16_chars:
                                <1>       ; 08/08/2016
                                <1>       ; 10/07/2016
                                <1>       ; Setup ROM 8x14 Font for Graphics Mode (VGA)
                                <1>       ;
                                <1>       ; derived from 'Plex86/Bochs VGABios' source code
                                <1>       ; vgabios-0.7a (2011)
                                <1>       ; by the LGPL VGABios developers Team (2001-2008)
                                <1>       ; 'vgabios.c', 'biosfn_load_gfx_8_16_chars'
                                <1>
                                <1>       ; biosfn_load_gfx_8_16_chars (BL)
                                <1>       ; /* set 0x43 INT pointer */
                                <1>       ; write_word(0x0, 0x43*4, &vgafont16);
                                <1>       ; write_word(0x0, 0x43*4+2, 0xC000);
00002E0A C705[AE5F0100]-        <1>       mov   dword [VGA_INT43H], vgafont16
00002E10 [7C3C0100]             <1>
                                <1>
                                <1>       ; BL    screen rows code: 00H = user-specified (in DL)
                                <1>         ;                       01H = 14 rows
                                <1>         ;                       02H = 25 rows
                                <1>         ;                       03H = 43 rows
                                <1>         ; DL    (when BL=0) custom number of char rows on screen
                                <1>
                                <1>       ; switch (BL) {
                                <1>       ; case 0:
                                <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, DL-1);
                                <1>       ;    break;
00002E14 20DB                   <1>       and   bl, bl
00002E16 7508                   <1>       jnz   short l_gfx_8_16c_1
00002E18 8815[CA5E0000]         <1>       mov   [VGA_ROWS], dl  ; not DL-1 !
00002E1E EB23                   <1>       jmp   short l_gfx_8_16c_4
                                <1> l_gfx_8_16c_1:
                                <1>       ; case 1:
                                <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 13);
                                <1>       ;    break;
00002E20 FECB                   <1>       dec   bl
00002E22 7509                   <1>       jnz   short l_gfx_8_16c_2
                                <1>       ; bl = 1
00002E24 C605[CA5E0000]0E       <1>       mov   byte [VGA_ROWS], 14  ; not 13 !
00002E2B EB16                   <1>       jmp   short l_gfx_8_16c_4
                                <1> l_gfx_8_16c_2:
00002E2D FECB                   <1>       dec   bl
00002E2F 740B                   <1>       jz    short l_gfx_8_16c_3 ; bl = 2
00002E31 FECB                   <1>       dec   bl
00002E33 750E                   <1>       jnz   short l_gfx_8_16c_4 ; bl > 3
```

```
                                <1>       ; bl = 3
                                <1>       ; case 3:
                                <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 42);
                                <1>       ;    break;
00002E35 C605[CA5E0000]2B       <1>       mov   byte [VGA_ROWS], 43  ; not 42 !
                                <1> l_gfx_8_16c_3:
                                <1>       ; case 2:
                                <1>       ; default:
                                <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 24);
                                <1>       ;    break;
                                <1>       ; bl = 2 or bl > 3
00002E3C C605[CA5E0000]19       <1>       mov   byte [VGA_ROWS], 25  ; not 24 !
                                <1>       ; }
                                <1> l_gfx_8_16c_4:
                                <1>       ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, 16);
00002E43 C605[C65E0000]10       <1>         mov     byte [CHAR_HEIGHT], 16
                                <1>       ; }
00002E4A C3                     <1>       retn
                                <1>
                                <1> get_font_info:
                                <1>       ; 19/09/2016
                                <1>       ; 08/08/2016
                                <1>       ; 10/07/2016
                                <1>       ; Get Current Character Generator Info (VGA)
                                <1>       ;
                                <1>       ; derived from 'Plex86/Bochs VGABios' source code
                                <1>       ; vgabios-0.7a (2011)
                                <1>       ; by the LGPL VGABios developers Team (2001-2008)
                                <1>       ; 'vgabios.c', 'biosfn_get_font_info'
                                <1>
                                <1>       ; Modified for TRDOS 386 !
                                <1>       ;
                                <1>       ; INPUT ->
                                <1>       ;    AX = 1130h
                                <1>       ;    BL = 0 -> Get info for current VGA font
                                <1>       ;             (BH = unused)
                                <1>       ;    19/09/2016
                                <1>       ;    BL > 0 -> Get requested character font data
                                <1>       ;       BL = 1 -> vgafont8
                                <1>       ;         BL = 2 -> vgafont14
                                <1>       ;       BL = 3 -> vgafont16
                                <1>       ;       BL > 3 -> Invalid function (for now!)
                                <1>       ;       BH = ASCII code of the first character
                                <1>       ;       ECX = Number of characters from the 1st char
                                <1>       ;       ECX >= 256 -> All (256-BH) characters
                                <1>       ;       ECX = 0 -> All characters (BH = unused)
                                <1>       ;        EDX = User's Buffer Address
                                <1>       ; OUTPUT ->
                                <1>       ;    AL = height (scanlines), bytes per character
                                <1>       ;    AH = screen rows
                                <1>       ;    Byte 16-23 of EAX = number of columns
                                <1>       ;    Byte 24-31 of EAX =
                                <1>       ;        0 -> default font (not configured yet)
                                <1>       ;        0FFh -> user defined font
                                <1>       ;        14 = vgafont14
                                <1>       ;         8 = vgafont8
                                <1>       ;        16 = vgafont16
                                <1>       ;   If BL input > 0 ->
                                <1>       ;       EAX = Actual transfer count
                                <1>       ;
00002E4B 20DB                   <1>       and   bl, bl
00002E4D 7408                   <1>       jz    short gfi_0
                                <1>       ; invalid function (input)
00002E4F 80FB03                 <1>       cmp   bl, 3
00002E52 7642                   <1>       jna   short gfi_4
00002E54 31C0                   <1>       xor   eax, eax ; 0
00002E56 C3                     <1>       retn
                                <1> gfi_0:
00002E57 A0[C65E0000]           <1>       mov   al, [CHAR_HEIGHT]
00002E5C 8A25[CA5E0000]         <1>       mov   ah, [VGA_ROWS]
00002E62 C1E010                 <1>       shl   eax, 16
00002E65 A0[C45E0000]           <1>       mov   al, [CRT_COLS]
00002E6A 8B0D[AE5F0100]         <1>       mov   ecx, [VGA_INT43H]
00002E70 21C9                   <1>       and   ecx, ecx
00002E72 741E                   <1>       jz    short gfi_2 ; 0 = default font
00002E74 41                     <1>       inc   ecx ; 0FFFFFFFFh -> 0 (user defined font)
00002E75 7504                   <1>       jnz   short gfi_1
00002E77 FECC                   <1>       dec   ah ; 0FFh
00002E79 EB17                   <1>       jmp   short gfi_2
                                <1> gfi_1:
00002E7B 49                     <1>       dec   ecx ; 08/08/2016
00002E7C B40E                   <1>       mov   ah, 14
00002E7E 81F9[7C2E0100]         <1>       cmp   ecx, vgafont14
00002E84 740C                   <1>       je    short gfi_2
00002E86 B408                   <1>       mov   ah, 8
00002E88 81F9[7C260100]         <1>       cmp   ecx, vgafont8
00002E8E 7402                   <1>       je    short gfi_2
                                <1>       ; vgafont16
00002E90 D0E4                   <1>       shl   ah, 1 ; ah = 16
                                <1> gfi_2:
00002E92 C1C010                 <1>       rol   eax, 16
                                <1> gfi_3:
00002E95 C3                     <1>       retn
                                <1> gfi_4:
00002E96 89D7                   <1>       mov   edi, edx ; **
00002E98 80FB02                 <1>       cmp   bl, 2
00002E9B 720B                   <1>       jb    short gfi_5
00002E9D 772F                   <1>       ja    short gfi_7
                                <1>       ;BL = 2 -> vgafont14
00002E9F BE[7C2E0100]           <1>       mov   esi, vgafont14 ; *
00002EA4 B30E                   <1>       mov   bl, 14
00002EA6 EB07                   <1>       jmp   short gfi_6
```

```
                        <1> gfi_5:
                        <1>       ;BL = 1 -> vgafont8
00002EA8 BE[7C260100]   <1>       mov   esi, vgafont8 ; *
00002EAD B308           <1>       mov   bl, 8
                        <1> gfi_6:
00002EAF 09C9           <1>       or    ecx, ecx
00002EB1 7424           <1>       jz    short gfi_8 ; all chars from the 00h
00002EB3 88F8           <1>       mov   al, bh ; character index
00002EB5 F6E3           <1>       mul   bl ; char index * char height/size
00002EB7 0FB7D0         <1>       movzx edx, ax
00002EBA 01D6           <1>       add   esi, edx ; *
00002EBC 66BAFF00       <1>       mov   dx, 255
00002EC0 28FA           <1>       sub   dl, bh
00002EC2 6642           <1>       inc   dx
00002EC4 39D1           <1>       cmp   ecx, edx
00002EC6 770F           <1>       ja    short gfi_8
00002EC8 7412           <1>       je    short gfi_9
00002ECA 89D1           <1>       mov   ecx, edx
00002ECC EB0E           <1>       jmp   short gfi_9
                        <1> gfi_7:
                        <1>       ;BL = 3 -> vgafont16
00002ECE BE[7C3C0100]   <1>       mov   esi, vgafont16 ; *
00002ED3 B310           <1>       mov   bl, 16
00002ED5 EBD8           <1>       jmp   short gfi_6
                        <1> gfi_8:
00002ED7 B900010000     <1>       mov   ecx, 256
                        <1> gfi_9:
00002EDC 6689C8         <1>       mov   ax, cx ; character count
00002EDF 30FF           <1>       xor   bh, bh
00002EE1 66F7E3         <1>       mul   bx ; char count * char height/size
00002EE4 6689C1         <1>       mov   cx, ax
                        <1>
                        <1>       ; ESI = source address in system space
                        <1>       ; EDI = user's buffer address
                        <1>       ; ECX = transfer (byte) count
00002EE7 E89AB90000     <1>       call  transfer_to_user_buffer
00002EEC 89C8           <1>       mov   eax, ecx ; actual transfer count
00002EEE C3             <1>       retn
                        <1>
                        <1> vga_pal_funcs:
                        <1>       ; 10/08/2016
                        <1>       ; VGA Palette functions
                        <1>       ;
                        <1>       ; derived from 'Plex86/Bochs VGABios' source code
                        <1>       ; vgabios-0.7a (2011)
                        <1>       ; by the LGPL VGABios developers Team (2001-2008)
                        <1>       ; 'vgabios.c', 'vgarom.asm'
                        <1>
00002EEF 3C00           <1>       cmp   al, 0
00002EF1 0F848F000000   <1>         je      set_single_palette_reg
                        <1> vga_palf_1001:
00002EF7 3C01           <1>       cmp   al, 1
00002EF9 0F84B4000000   <1>         je      set_overscan_border_color
                        <1> vga_palf_1002:
00002EFF 3C02           <1>       cmp   al, 2
00002F01 0F84B0000000   <1>         je      set_all_palette_reg
                        <1> vga_palf_1003:
00002F07 3C03           <1>       cmp   al, 3
00002F09 0F84E8000000   <1>         je      toggle_intensity
                        <1> vga_palf_1007:
00002F0F 3C07           <1>       cmp   al, 7
00002F11 0F840D010000   <1>         je      get_single_palette_reg
00002F17 7266           <1>       jb    short vga_palf_unknown
                        <1> vga_palf_1008:
00002F19 3C08           <1>       cmp   al, 8
00002F1B 0F8437010000   <1>         je      read_overscan_border_color
                        <1> vga_palf_1009:
00002F21 3C09           <1>       cmp   al, 9
00002F23 0F8433010000   <1>       je    get_all_palette_reg
                        <1> vga_palf_1010:
00002F29 3C10           <1>       cmp   al, 10h
00002F2B 0F8487010000   <1>       je    set_single_dac_reg
00002F31 724C           <1>       jb    short vga_palf_unknown
                        <1> vga_palf_1012:
00002F33 3C12           <1>       cmp   al, 12h
00002F35 0F8498010000   <1>         je      set_all_dac_reg
00002F3B 7242           <1>       jb    short vga_palf_unknown
                        <1> vga_palf_1013:
00002F3D 3C13           <1>       cmp   al, 13h
00002F3F 0F84CC010000   <1>         je      select_video_dac_color_page
                        <1> vga_palf_1015:
00002F45 3C15           <1>       cmp   al, 15h
00002F47 0F8412020000   <1>       je    read_single_dac_reg
00002F4D 7230           <1>       jb    short vga_palf_unknown
                        <1> vga_palf_1017:
00002F4F 3C17           <1>       cmp   al, 17h
00002F51 0F8428020000   <1>         je      read_all_dac_reg
00002F57 7226           <1>       jb    short vga_palf_unknown
                        <1> vga_palf_1018:
00002F59 3C18           <1>       cmp   al, 18h
00002F5B 0F845E020000   <1>         je      set_pel_mask
                        <1> vga_palf_1019:
00002F61 3C19           <1>       cmp   al, 19h
00002F63 0F8462020000   <1>         je      read_pel_mask
                        <1> vga_palf_101A:
00002F69 3C1A           <1>       cmp   al, 1Ah
00002F6B 0F8468020000   <1>         je      read_video_dac_state
                        <1> vga_palf_101B:
00002F71 3C1B           <1>       cmp   al, 1Bh
                        <1>       ;jne   short vga_palf_unknown
00002F73 770A           <1>       ja    short vga_palf_unknown
                        <1>
```

```
5777 00002F75 E80CF7FFFF          <1>       call   gray_scale_summing
5778 00002F7A E9D5E5FFFF          <1>         jmp     VIDEO_RETURN
5779                              <1>
5780                              <1> vga_palf_unknown:
5781 00002F7F 29C0                <1>       sub    eax, eax ; 0 = invalid function
5782 00002F81 E9D3E5FFFF          <1>         jmp     _video_return
5783                              <1>
5784                              <1> set_single_palette_reg:
5785                              <1>       ; 10/08/2016
5786                              <1>       ; Set One Palette Register
5787                              <1>       ; BL = register number to set
5788                              <1>       ;     (a 4-bit attribute nibble: 00h-0Fh)
5789                              <1>       ; BH = 6-bit RGB color to display
5790                              <1>       ;      for that attribute
5791                              <1>
5792 00002F86 80FB14              <1>       cmp    bl, 14h
5793                              <1>       ;ja    short no_actl_reg1
5794 00002F89 0F87C5E5FFFF        <1>       ja     VIDEO_RETURN
5795 00002F8F 6650                <1>       push   ax
5796 00002F91 6652                <1>       push   dx
5797 00002F93 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
5798 00002F97 EC                  <1>       in     al, dx
5799 00002F98 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
5800 00002F9C 88D8                <1>       mov    al, bl
5801 00002F9E EE                  <1>       out    dx, al
5802 00002F9F 88F8                <1>       mov    al, bh
5803 00002FA1 EE                  <1>       out    dx, al
5804 00002FA2 B020                <1>       mov    al, 20h
5805 00002FA4 EE                  <1>       out    dx, al
5806                              <1>       ; ifdef VBOX
5807 00002FA5 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
5808 00002FA9 EC                  <1>       in     al, dx
5809                              <1>       ; endif ; VBOX
5810 00002FAA 665A                <1>       pop    dx
5811 00002FAC 6658                <1>       pop    ax
5812                              <1> ;no_actl_reg1:
5813 00002FAE E9A1E5FFFF          <1>       jmp    VIDEO_RETURN
5814                              <1>
5815                              <1> set_overscan_border_color:
5816                              <1>       ; 10/08/2016
5817                              <1>       ; Set Overscan/Border Color Register
5818                              <1>       ; BH = 6-bit RGB color to display
5819                              <1>       ;      for that attribute
5820                              <1>
5821 00002FB3 B311                <1>       mov    bl, 11h
5822 00002FB5 EBCF                <1>       jmp    short set_single_palette_reg
5823                              <1>
5824                              <1> set_all_palette_reg:
5825                              <1>       ; 10/08/2016
5826                              <1>       ; Set All Palette Registers and Overscan
5827                              <1>       ; EDX = Address of 17 bytes;
5828                              <1>       ; an rgbRGB value for each of 16 palette
5829                              <1>       ; registers plus one for the border.
5830                              <1>
5831 00002FB7 89D6                <1>       mov    esi, edx ; user buffer
5832 00002FB9 B911000000          <1>       mov    ecx, 17
5833 00002FBE 89E7                <1>       mov    edi, esp
5834 00002FC0 83EC14              <1>       sub    esp, 20
5835 00002FC3 E808B90000          <1>       call   transfer_from_user_buffer
5836                              <1>       ;jc    VIDEO_RETURN
5837                              <1>
5838 00002FC8 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
5839 00002FCC EC                  <1>       in     al, dx
5840 00002FCD B100                <1>       mov    cl, 0
5841 00002FCF 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
5842                              <1> set_palette_loop:
5843 00002FD3 88C8                <1>       mov    al, cl
5844 00002FD5 EE                  <1>       out    dx, al
5845 00002FD6 8A07                <1>       mov    al, [edi]
5846 00002FD8 EE                  <1>       out    dx, al
5847 00002FD9 47                  <1>       inc    edi
5848 00002FDA FEC1                <1>       inc    cl
5849 00002FDC 80F910              <1>       cmp    cl, 10h
5850 00002FDF 75F2                <1>       jne    short set_palette_loop
5851 00002FE1 B011                <1>       mov    al, 11h
5852 00002FE3 EE                  <1>       out    dx, al
5853 00002FE4 8A07                <1>       mov    al, [edi]
5854 00002FE6 EE                  <1>       out    dx, al
5855 00002FE7 B020                <1>       mov    al, 20h
5856 00002FE9 EE                  <1>       out    dx, al
5857                              <1>       ; ifdef VBOX
5858 00002FEA 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
5859 00002FEE EC                  <1>       in     al, dx
5860                              <1>       ; endif ; VBOX
5861 00002FEF 83C414              <1>       add    esp, 20
5862 00002FF2 E95DE5FFFF          <1>       jmp    VIDEO_RETURN
5863                              <1>
5864                              <1> toggle_intensity:
5865                              <1>       ; 10/08/2016
5866                              <1>       ; Select Foreground Blink or Bold Background
5867                              <1>       ; BL = 00h = enable bold backgrounds
5868                              <1>       ;          (16 background colors)
5869                              <1>         ;      01h = enable blinking foreground
5870                              <1>       ;          (8 background colors)
5871                              <1>
5872 00002FF7 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
5873 00002FFB EC                  <1>       in     al, dx
5874 00002FFC 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
5875 00003000 B010                <1>       mov    al, 10h
5876 00003002 EE                  <1>       out    dx, al
5877 00003003 66BAC103            <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
5878 00003007 EC                  <1>       in     al, dx
```

```
5879 00003008 24F7              <1>       and    al, 0F7h
5880 0000300A 80E301            <1>       and    bl, 01h
5881 0000300D C0E303            <1>       shl    bl, 3
5882 00003010 08D8              <1>       or     al, bl
5883 00003012 66BAC003          <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
5884 00003016 EE                <1>       out    dx, al
5885 00003017 B020              <1>       mov    al, 20h
5886 00003019 EE                <1>       out    dx, al
5887                            <1>       ; ifdef VBOX
5888 0000301A 66BADA03          <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
5889 0000301E EC                <1>       in     al, dx
5890                            <1>       ; endif ; VBOX
5891 0000301F E930E5FFFF        <1>       jmp    VIDEO_RETURN
5892                            <1>
5893                            <1> get_single_palette_reg:
5894                            <1>       ; 10/08/2016
5895                            <1>       ; Read One Palette Register
5896                            <1>         ; INPUT:
5897                            <1>       ; BL = Palette register to read (00h-0Fh)
5898                            <1>       ; OUTPUT:
5899                            <1>       ; BH = Current rgbRGB value of specified register
5900                            <1>       ;      for that attribute
5901                            <1>
5902 00003024 80FB14            <1>       cmp    bl, 14h
5903                            <1>       ;ja    short no_actl_reg2
5904 00003027 0F8727E5FFFF      <1>       ja     VIDEO_RETURN
5905                            <1>
5906 0000302D 66BADA03          <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
5907 00003031 EC                <1>       in     al, dx
5908 00003032 66BAC003          <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
5909 00003036 88D8              <1>       mov    al, bl
5910 00003038 EE                <1>       out    dx, al
5911 00003039 66BAC103          <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
5912 0000303D EC                <1>       in     al, dx
5913 0000303E 8844240D          <1>       mov    [esp+13], al ; bh
5914 00003042 66BADA03          <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
5915 00003046 EC                <1>       in     al, dx
5916 00003047 66BAC003          <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
5917 0000304B B020              <1>       mov    al, 20h
5918 0000304D EE                <1>       out    dx, al
5919                            <1>       ; ifdef VBOX
5920 0000304E 66BADA03          <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
5921 00003052 EC                <1>       in     al, dx
5922                            <1>       ; endif ; VBOX
5923 00003053 E9FCE4FFFF        <1>       jmp    VIDEO_RETURN
5924                            <1>
5925                            <1> read_overscan_border_color:
5926                            <1>       ; 10/08/2016
5927                            <1>       ; Read Overscan Register
5928                            <1>       ; OUTPUT:
5929                            <1>       ; BH = current rgbRGB value
5930                            <1>       ;      of the overscan/border register
5931                            <1>
5932 00003058 B311              <1>       mov    bl, 11h
5933 0000305A EBC8              <1>       jmp    short get_single_palette_reg
5934                            <1>
5935                            <1> get_all_palette_reg:
5936                            <1>       ; 10/08/2016
5937                            <1>       ; Read All Palette Registers
5938                            <1>       ; EDX = Address of 17-byte buffer
5939                            <1>       ;      to receive data
5940                            <1>
5941 0000305C 89D7              <1>       mov    edi, edx
5942 0000305E 89E3              <1>       mov    ebx, esp
5943 00003060 89DE              <1>       mov    esi, ebx
5944 00003062 83EC14            <1>       sub    esp, 20
5945                            <1>
5946 00003065 B100              <1>       mov    cl, 0
5947                            <1> get_palette_loop:
5948 00003067 66BADA03          <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
5949 0000306B EC                <1>       in     al, dx
5950 0000306C 66BAC003          <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
5951 00003070 88C8              <1>       mov    al, cl
5952 00003072 EE                <1>       out    dx, al
5953 00003073 66BAC103          <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
5954 00003077 EC                <1>       in     al, dx
5955 00003078 8803              <1>       mov    [ebx], al
5956 0000307A 43                <1>       inc    ebx
5957 0000307B FEC1              <1>       inc    cl
5958 0000307D 80F910            <1>       cmp    cl, 10h
5959 00003080 75E5              <1>       jne    short get_palette_loop
5960 00003082 66BADA03          <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
5961 00003086 EC                <1>       in     al, dx
5962 00003087 66BAC003          <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
5963 0000308B B011              <1>       mov    al, 11h
5964 0000308D EE                <1>       out    dx, al
5965 0000308E 66BAC103          <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
5966 00003092 EC                <1>       in     al, dx
5967 00003093 8803              <1>       mov    [ebx], al
5968 00003095 66BADA03          <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
5969 00003099 EC                <1>       in     al, dx
5970 0000309A 66BAC003          <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
5971 0000309E B020              <1>       mov    al, 20h
5972 000030A0 EE                <1>       out    dx, al
5973                            <1>       ; ifdef VBOX
5974 000030A1 66BADA03          <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
5975 000030A5 EC                <1>       in     al, dx
5976                            <1>       ; endif ; VBOX
5977                            <1>
5978 000030A6 B911000000        <1>       mov    ecx, 17 ; transfer (byte) count
5979                            <1>       ; ESI = source address in system space
5980                            <1>       ; EDI = user's buffer address
```

```
5981 000030AB E8D6B70000          <1>       call   transfer_to_user_buffer
5982                              <1>
5983 000030B0 83C414              <1>       add    esp, 20
5984 000030B3 E99CE4FFFF          <1>       jmp    VIDEO_RETURN
5985                              <1>
5986                              <1> set_single_dac_reg:
5987                              <1>       ; 10/08/2016
5988                              <1>       ; Set One DAC Color Register
5989                              <1>       ; BX = color register to set (0-255)
5990                              <1>         ; CH = green value (00h-3Fh)
5991                              <1>         ; CL = blue value  (00h-3Fh)
5992                              <1>         ; DH = red value   (00h-3Fh)
5993                              <1>
5994 000030B8 6652                <1>       push   dx
5995 000030BA 66BAC803            <1>       mov    dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
5996 000030BE 88D8                <1>       mov    al, bl
5997 000030C0 EE                  <1>       out    dx, al
5998                              <1>       ;mov   dx, 3C9h ; VGAREG_DAC_DATA
5999 000030C1 6642                <1>       inc    dx
6000 000030C3 6658                <1>       pop    ax
6001 000030C5 88E0                <1>       mov    al, ah
6002 000030C7 EE                  <1>       out    dx, al
6003 000030C8 88E8                <1>       mov    al, ch
6004 000030CA EE                  <1>       out    dx, al
6005 000030CB 88C8                <1>       mov    al, cl
6006 000030CD EE                  <1>       out    dx, al
6007 000030CE E981E4FFFF          <1>       jmp    VIDEO_RETURN
6008                              <1>
6009                              <1> set_all_dac_reg:
6010                              <1>       ; 12/08/2016
6011                              <1>       ; 11/08/2016
6012                              <1>       ; 10/08/2016
6013                              <1>       ; Set a Block of DAC Color Register
6014                              <1>       ; BX = first DAC register to set (0-00FFh)
6015                              <1>       ; ECX = number of registers to set (0-00FFh)
6016                              <1>       ; EDX = addr of a table of R,G,B values
6017                              <1>       ;     (it will be CX*3 bytes long)
6018                              <1>
6019 000030D3 89D6                <1>       mov    esi, edx ; user buffer
6020 000030D5 89CA                <1>       mov    edx, ecx
6021 000030D7 66D1E1              <1>       shl    cx, 1 ; *2
6022 000030DA 01D1                <1>       add    ecx, edx ; ecx = 3*ecx
6023 000030DC 89E5                <1>       mov    ebp, esp
6024 000030DE 89EF                <1>       mov    edi, ebp
6025 000030E0 29CF                <1>       sub    edi, ecx
6026 000030E2 6683E7FC            <1>       and    di, 0FFFCh ; (dword alignment)
6027 000030E6 89FC                <1>       mov    esp, edi
6028 000030E8 E8E3B70000          <1>       call   transfer_from_user_buffer
6029                              <1>       ;jc    VIDEO_RETURN
6030                              <1>
6031 000030ED 89D1                <1>       mov    ecx, edx
6032 000030EF 66BAC803            <1>       mov    dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
6033 000030F3 88D8                <1>       mov    al, bl
6034 000030F5 EE                  <1>       out    dx, al
6035 000030F6 66BAC903            <1>       mov    dx, 3C9h ; VGAREG_DAC_DATA
6036                              <1> set_dac_loop:
6037 000030FA 8A07                <1>       mov    al, [edi]
6038 000030FC EE                  <1>       out    dx, al
6039 000030FD 47                  <1>       inc    edi
6040 000030FE 8A07                <1>       mov    al, [edi]
6041 00003100 EE                  <1>       out    dx, al
6042 00003101 47                  <1>       inc    edi
6043 00003102 8A07                <1>       mov    al, [edi]
6044 00003104 EE                  <1>       out    dx, al
6045 00003105 47                  <1>       inc    edi
6046 00003106 6649                <1>       dec    cx
6047 00003108 75F0                <1>       jnz    short set_dac_loop
6048 0000310A 89EC                <1>       mov    esp, ebp
6049 0000310C E943E4FFFF          <1>       jmp    VIDEO_RETURN
6050                              <1>
6051                              <1> select_video_dac_color_page:
6052                              <1>       ; 10/08/2016
6053                              <1>       ; DAC Color Paging Functions
6054                              <1>       ; BL = 00H = select color paging mode
6055                              <1>         ;       BH = paging mode
6056                              <1>         ;            00h = 4 blocks of 64 registers
6057                              <1>         ;            01h = 16 blocks of 16 registers
6058                              <1>       ; BL = 01H = activate color page
6059                              <1>         ;       BH = DAC color page number
6060                              <1>         ;            00h-03h (4-page/64-reg mode)
6061                              <1>         ;            00h-0Fh (16-page/16-reg mode)
6062                              <1>
6063 00003111 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
6064 00003115 EC                  <1>       in     al, dx
6065 00003116 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
6066 0000311A B010                <1>       mov    al, 10h
6067 0000311C EE                  <1>       out    dx, al
6068 0000311D 66BAC103            <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
6069 00003121 EC                  <1>       in     al, dx
6070 00003122 80E301              <1>       and    bl, 01h
6071 00003125 750E                <1>       jnz    short set_dac_page
6072 00003127 247F                <1>       and    al, 07Fh
6073 00003129 C0E707              <1>       shl    bh, 7
6074 0000312C 08F8                <1>       or     al, bh
6075 0000312E 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
6076 00003132 EE                  <1>       out    dx, al
6077 00003133 EB1D                <1>       jmp    short set_actl_normal
6078                              <1> set_dac_page:
6079 00003135 6650                <1>       push   ax
6080 00003137 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
6081 0000313B EC                  <1>       in     al, dx
6082 0000313C 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
```

```
6083 00003140 B014              <1>       mov    al, 14h
6084 00003142 EE                <1>       out    dx, al
6085 00003143 6658              <1>       pop    ax
6086 00003145 2480              <1>       and    al, 80h
6087 00003147 7503              <1>       jnz    short set_dac_16_page
6088 00003149 C0E702            <1>       shl    bh, 2
6089                            <1> set_dac_16_page:
6090 0000314C 80E70F            <1>       and    bh, 0Fh
6091 0000314F 88F8              <1>       mov    al, bh
6092 00003151 EE                <1>       out    dx, al
6093                            <1> set_actl_normal:
6094 00003152 B020              <1>       mov    al, 20h
6095 00003154 EE                <1>       out    dx, al
6096                            <1>       ; ifdef VBOX
6097 00003155 66BADA03          <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
6098 00003159 EC                <1>       in     al, dx
6099                            <1>       ; endif ; VBOX
6100 0000315A E9F5E3FFFF        <1>       jmp    VIDEO_RETURN
6101                            <1>
6102                            <1> read_single_dac_reg:
6103                            <1>       ; 10/08/2016
6104                            <1>       ; Read One DAC Color Register
6105                            <1>       ; INPUT:
6106                            <1>       ; BX = color register to read (0-255)
6107                            <1>       ; OUTPUT:
6108                            <1>       ; CH = green value (00h-3Fh)
6109                            <1>         ; CL = blue value  (00h-3Fh)
6110                            <1>         ; DH = red value   (00h-3Fh)
6111                            <1>
6112 0000315F 66BAC703          <1>       mov    dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
6113 00003163 88D8              <1>       mov    al, bl
6114 00003165 EE                <1>       out    dx, al
6115 00003166 66BAC903          <1>       mov    dx, 3C9h ; VGAREG_DAC_DATA
6116 0000316A EC                <1>       in     al, dx
6117 0000316B 88442415          <1>       mov    [esp+21], al ; dh
6118 0000316F EC                <1>       in     al, dx
6119 00003170 88C5              <1>       mov    ch, al
6120 00003172 EC                <1>       in     al, dx
6121 00003173 88C1              <1>       mov    cl, al
6122 00003175 66894C2410        <1>       mov    [esp+16], cx ; cx
6123 0000317A E9D5E3FFFF        <1>       jmp    VIDEO_RETURN
6124                            <1>
6125                            <1> read_all_dac_reg:
6126                            <1>       ; 12/08/2016
6127                            <1>       ; 11/08/2016
6128                            <1>       ; 10/08/2016
6129                            <1>       ; Read a Block of DAC Color Registers
6130                            <1>         ; BX = first DAC register to read (0-00FFh)
6131                            <1>         ; ECX = number of registers to read (0-00FFh)
6132                            <1>         ; EDX = addr of a buffer to hold R,G,B values
6133                            <1>       ;      (CX*3 bytes long)
6134                            <1>
6135 0000317F 89D7              <1>       mov    edi, edx ; user buffer
6136 00003181 89CA              <1>       mov    edx, ecx
6137 00003183 66D1E2            <1>       shl    dx, 1 ; *2
6138 00003186 01CA              <1>       add    edx, ecx ; edx = 3*ecx
6139 00003188 89E5              <1>       mov    ebp, esp
6140 0000318A 89EE              <1>       mov    esi, ebp
6141 0000318C 29D6              <1>       sub    esi, edx
6142 0000318E 6683E6FC          <1>       and    si, 0FFFCh ; (dword alignment)
6143 00003192 89F4              <1>       mov    esp, esi
6144 00003194 52                <1>       push   edx ; 3*ecx
6145 00003195 66BAC703          <1>       mov    dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
6146 00003199 88D8              <1>       mov    al, bl
6147 0000319B EE                <1>       out    dx, al
6148 0000319C 66BAC903          <1>       mov    dx, 3C9h ; VGAREG_DAC_DATA
6149 000031A0 89F3              <1>       mov    ebx, esi
6150                            <1> read_dac_loop:
6151 000031A2 EC                <1>       in     al, dx
6152 000031A3 8803              <1>       mov    [ebx], al
6153 000031A5 43                <1>       inc    ebx
6154 000031A6 EC                <1>       in     al, dx
6155 000031A7 8803              <1>       mov    [ebx], al
6156 000031A9 43                <1>       inc    ebx
6157 000031AA EC                <1>       in     al, dx
6158 000031AB 8803              <1>       mov    [ebx], al
6159 000031AD 43                <1>       inc    ebx
6160 000031AE 6649              <1>       dec    cx
6161 000031B0 75F0              <1>       jnz    short read_dac_loop
6162 000031B2 59                <1>       pop    ecx ; 3*ecx
6163                            <1>       ; ECX = transfer (byte) count
6164                            <1>       ; ESI = source address in system space
6165                            <1>       ; EDI = user's buffer address
6166 000031B3 E8CEB60000        <1>       call   transfer_to_user_buffer
6167 000031B8 89EC              <1>       mov    esp, ebp
6168 000031BA E995E3FFFF        <1>       jmp    VIDEO_RETURN
6169                            <1>
6170                            <1> set_pel_mask:
6171                            <1>       ; 10/08/2016
6172                            <1>       ; BL = mask value
6173 000031BF 66BAC603          <1>       mov    dx, 3C6h ; VGAREG_PEL_MASK
6174 000031C3 88D8              <1>       mov    al, bl
6175 000031C5 EE                <1>       out    dx, al
6176 000031C6 E989E3FFFF        <1>       jmp    VIDEO_RETURN
6177                            <1>
6178                            <1> read_pel_mask:
6179                            <1>       ; 10/08/2016
6180                            <1>       ; Output: BL = mask value
6181 000031CB 66BAC603          <1>       mov    dx, 3C6h ; VGAREG_PEL_MASK
6182 000031CF EC                <1>       in     al, dx
6183 000031D0 8844240C          <1>       mov    [esp+12], al ; bl
6184 000031D4 E97BE3FFFF        <1>       jmp    VIDEO_RETURN
```

```
                                <1>
                                <1> read_video_dac_state:
                                <1>       ; 10/08/2016
                                <1>       ; Query DAC Color Paging State
                                <1>       ; Output:
                                <1>       ; BH = current active DAC color page
                                <1>         ; BL = current active DAC paging mode
                                <1>
000031D9 66BADA03               <1>       mov   dx, 3DAh ; VGAREG_ACTL_RESET
000031DD EC                     <1>       in    al, dx
000031DE 66BAC003               <1>       mov   dx, 3C0h ; VGAREG_ACTL_ADDRESS
000031E2 B010                   <1>       mov   al, 10h
000031E4 EE                     <1>       out   dx, al
000031E5 66BAC103               <1>       mov   dx, 3C1h ; VGAREG_ACTL_READ_DATA
000031E9 EC                     <1>       in    al, dx
000031EA 88C3                   <1>       mov   bl, al
000031EC C0EB07                 <1>       shr   bl, 7
000031EF 66BADA03               <1>       mov   dx, 3DAh ; VGAREG_ACTL_RESET
000031F3 EC                     <1>       in    al, dx
000031F4 66BAC003               <1>       mov   dx, 3C0h ; VGAREG_ACTL_ADDRESS
000031F8 B014                   <1>       mov   al, 14h
000031FA EE                     <1>       out   dx, al
000031FB 66BAC103               <1>       mov   dx, 3C1h ; VGAREG_ACTL_READ_DATA
000031FF EC                     <1>       in    al, dx
00003200 88C7                   <1>       mov   bh, al
00003202 80E70F                 <1>       and   bh, 0Fh
00003205 F6C301                 <1>       test  bl, 01
00003208 7503                   <1>       jnz   short get_dac_16_page
0000320A C0EF02                 <1>       shr   bh, 2
                                <1> get_dac_16_page:
0000320D 66BADA03               <1>       mov   dx, 3DAh ; VGAREG_ACTL_RESET
00003211 EC                     <1>       in    al, dx
00003212 66BAC003               <1>       mov   dx, 3C0h ; VGAREG_ACTL_ADDRESS
00003216 B020                   <1>       mov   al, 20h
00003218 EE                     <1>       out   dx, al
                                <1>       ; ifdef VBOX
00003219 66BADA03               <1>       mov   dx, 3DAh ; VGAREG_ACTL_RESET
0000321D EC                     <1>       in    al, dx
                                <1>       ; endif ; VBOX
0000321E 66895C240C             <1>       mov   [esp+12], bx ; bx
00003223 E92CE3FFFF             <1>       jmp   VIDEO_RETURN
                                <1>
                                <1> ; % include 'vidata.s' ; VIDEO DATA
                                <1>
                                <1> ; /// End Of VIDEO FUNCTIONS ///
                                
                                    setup_rtc_int:
                                    ; source: http://wiki.osdev.org/RTC
00003228 FA                             cli          ; disable interrupts
                                        ; default int frequency is 1024 Hz (Lower 4 bits of register A is 0110b or 6)
                                        ; in order to change this ...
                                        ; frequency  = 32768 >> (rate-1) --> 32768 >> 5 = 1024
                                        ; (rate must be above 2 and not over 15)
                                        ; new rate = 15 --> 32768 >> (15-1) = 2 Hz
00003229 B08A                           mov   al, 8Ah
0000322B E670                           out   70h, al ; set index to register A, disable NMI
0000322D 90                             nop
0000322E E471                           in    al, 71h ; get initial value of register A
00003230 88C4                           mov   ah, al
00003232 80E4F0                         and   ah, 0F0h
00003235 B08A                           mov   al, 8Ah
00003237 E670                           out   70h, al ; reset index to register A
00003239 88E0                           mov   al, ah
0000323B 0C0F                           or    al, 0Fh      ; new rate (0Fh -> 15)
0000323D E671                           out   71h, al ; write only our rate to A. Note, rate is the bottom 4 bits.
                                        ; enable RTC interrupt
0000323F B08B                           mov   al, 8Bh ;
00003241 E670                           out   70h, al ; select register B and disable NMI
00003243 90                             nop
00003244 E471                           in    al, 71h ; read the current value of register B
00003246 88C4                           mov   ah, al  ;
00003248 B08B                           mov   al, 8Bh ;
0000324A E670                           out   70h, al ; set the index again (a read will reset the index to register B)
0000324C 88E0                           mov   al, ah  ;
0000324E 0C40                           or    al, 40h ;
00003250 E671                           out   71h, al ; write the previous value ORed with 0x40. This turns on bit 6 of register B
00003252 FB                             sti
00003253 C3                             retn
                                
                                    ; Write memory information
                                    ; 29/01/2016
                                    ; 06/11/2014
                                    ; 14/08/2015
                                    memory_info:
00003254 A1[04520100]                   mov   eax, [memory_size] ; in pages
00003259 50                             push  eax
0000325A C1E00C                         shl   eax, 12               ; in bytes
0000325D BB0A000000                     mov   ebx, 10
00003262 89D9                           mov   ecx, ebx        ; 10
00003264 BE[A5120100]                   mov   esi, mem_total_b_str
00003269 E8BD000000                     call  bintdstr
0000326E 58                             pop   eax
0000326F B107                           mov   cl, 7
00003271 BE[C9120100]                   mov   esi, mem_total_p_str
00003276 E8B0000000                     call  bintdstr
                                        ; 14/08/2015
0000327B E8C8000000                     call  calc_free_mem
                                        ; edx = calculated free pages
                                        ; ecx = 0
00003280 A1[08520100]                   mov   eax, [free_pages]
00003285 39D0                           cmp   eax, edx ; calculated free mem value
                                              ; and initial free mem value are same or not?
```

```
1998 00003287 751D                      jne    short pmim ; print mem info with '?' if not
1999 00003289 52                        push   edx ; free memory in pages
2000                                    ;mov   eax, edx
2001 0000328A C1E00C                    shl    eax, 12 ; convert page count
2002                                                   ; to byte count
2003 0000328D B10A                      mov    cl, 10
2004 0000328F BE[E9120100]              mov    esi, free_mem_b_str
2005 00003294 E892000000                call   bintdstr
2006 00003299 58                        pop    eax
2007 0000329A B107                      mov    cl, 7
2008 0000329C BE[0D130100]              mov    esi, free_mem_p_str
2009 000032A1 E885000000                call   bintdstr
2010                                pmim:
2011 000032A6 BE[93120100]              mov    esi, msg_memory_info
2012                                    ;
2013 000032AB B407                      mov    ah, 07h ; Black background,
2014                                                   ; light gray forecolor
2015                                print_kmsg: ; 29/01/2016
2016 000032AD 8825[2F520100]            mov    [ccolor], ah
2017                                pkmsg_loop:
2018 000032B3 AC                        lodsb
2019 000032B4 08C0                      or     al, al
2020 000032B6 7410                      jz     short pkmsg_ok
2021 000032B8 56                        push   esi
2022                                    ; 13/05/2016
2023 000032B9 0FB61D[2F520100]          movzx  ebx, byte [ccolor]
2024                                                   ; Video page 0 (bh=0)
2025 000032C0 E8EDE9FFFF                call   _write_tty
2026 000032C5 5E                        pop    esi
2027 000032C6 EBEB                      jmp    short pkmsg_loop
2028                                pkmsg_ok:
2029 000032C8 C3                        retn
2030
2031                                ; Convert binary number to hexadecimal string
2032                                ; 10/05/2015
2033                                ; dsectpm.s (28/02/2015)
2034                                ; Retro UNIX 386 v1 - Kernel v0.2.0.6
2035                                ; 01/12/2014
2036                                ; 25/11/2014
2037                                ;
2038                                bytetohex:
2039                                      ; INPUT ->
2040                                      ;      AL = byte (binary number)
2041                                      ; OUTPUT ->
2042                                      ;      AX = hexadecimal string
2043                                      ;
2044 000032C9 53                        push   ebx
2045 000032CA 31DB                      xor    ebx, ebx
2046 000032CC 88C3                      mov    bl, al
2047 000032CE C0EB04                    shr    bl, 4
2048 000032D1 8A9B[1B330000]            mov    bl, [ebx+hexchrs]
2049 000032D7 86D8                      xchg   bl, al
2050 000032D9 80E30F                    and    bl, 0Fh
2051 000032DC 8AA3[1B330000]            mov    ah, [ebx+hexchrs]
2052 000032E2 5B                        pop    ebx
2053 000032E3 C3                        retn
2054
2055                                wordtohex:
2056                                      ; INPUT ->
2057                                      ;      AX = word (binary number)
2058                                      ; OUTPUT ->
2059                                      ;      EAX = hexadecimal string
2060                                      ;
2061 000032E4 53                        push   ebx
2062 000032E5 31DB                      xor    ebx, ebx
2063 000032E7 86E0                      xchg   ah, al
2064 000032E9 6650                      push   ax
2065 000032EB 88E3                      mov    bl, ah
2066 000032ED C0EB04                    shr    bl, 4
2067 000032F0 8A83[1B330000]            mov    al, [ebx+hexchrs]
2068 000032F6 88E3                      mov    bl, ah
2069 000032F8 80E30F                    and    bl, 0Fh
2070 000032FB 8AA3[1B330000]            mov    ah, [ebx+hexchrs]
2071 00003301 C1E010                    shl    eax, 16
2072 00003304 6658                      pop    ax
2073 00003306 5B                        pop    ebx
2074 00003307 EBC0                      jmp    short bytetohex
2075                                    ;mov   bl, al
2076                                    ;shr   bl, 4
2077                                    ;mov   bl, [ebx+hexchrs]
2078                                    ;xchg  bl, al
2079                                    ;and   bl, 0Fh
2080                                    ;mov   ah, [ebx+hexchrs]
2081                                    ;pop   ebx
2082                                    ;retn
2083
2084                                dwordtohex:
2085                                      ; INPUT ->
2086                                      ;      EAX = dword (binary number)
2087                                      ; OUTPUT ->
2088                                      ;      EDX:EAX = hexadecimal string
2089                                      ;
2090 00003309 50                        push   eax
2091 0000330A C1E810                    shr    eax, 16
2092 0000330D E8D2FFFFFF                call   wordtohex
2093 00003312 89C2                      mov    edx, eax
2094 00003314 58                        pop    eax
2095 00003315 E8CAFFFFFF                call   wordtohex
2096 0000331A C3                        retn
2097
2098                                ; 10/05/2015
2099                                hex_digits:
```

```
2100                                    hexchrs:
2101 0000331B 303132333435363738-               db '0123456789ABCDEF'
2101 00003324 39414243444546
2102
2103                                    ; Convert binary number to decimal/numeric string
2104                                    ; 06/11/2014
2105                                    ; Temporary Code
2106                                    ;
2107
2108                                    bintdstr:
2109                                            ; EAX = binary number
2110                                            ; ESI = decimal/numeric string address
2111                                            ; EBX = divisor (10)
2112                                            ; ECX = string length (<=10)
2113 0000332B 01CE                             add     esi, ecx
2114                                    btdstr0:
2115 0000332D 4E                               dec     esi
2116 0000332E 31D2                             xor     edx, edx
2117 00003330 F7F3                             div     ebx
2118 00003332 80C230                           add     dl, 30h
2119 00003335 8816                             mov     [esi], dl
2120 00003337 FEC9                             dec     cl
2121 00003339 740C                             jz      short btdstr2 ; 08/09/2016
2122 0000333B 09C0                             or      eax, eax
2123 0000333D 75EE                             jnz     short btdstr0
2124                                    btdstr1:
2125 0000333F 4E                               dec     esi
2126 00003340 C60620                             mov     byte [esi], 20h ; blank space
2127 00003343 FEC9                             dec     cl
2128 00003345 75F8                             jnz     short btdstr1
2129                                    btdstr2:
2130 00003347 C3                               retn
2131
2132                                    ; Calculate free memory pages on M.A.T.
2133                                    ; 06/11/2014
2134                                    ; Temporary Code
2135                                    ;
2136
2137                                    calc_free_mem:
2138 00003348 31D2                             xor     edx, edx
2139                                            ;xor    ecx, ecx
2140 0000334A 668B0D[18520100]                 mov     cx, [mat_size] ; in pages
2141 00003351 C1E10A                           shl     ecx, 10       ; 1024 dwords per page
2142 00003354 BE00001000                       mov     esi, MEM_ALLOC_TBL
2143                                    cfm0:
2144 00003359 AD                               lodsd
2145 0000335A 51                               push    ecx
2146 0000335B B920000000                       mov     ecx, 32
2147                                    cfm1:
2148 00003360 D1E8                             shr     eax, 1
2149 00003362 7301                             jnc     short cfm2
2150 00003364 42                               inc     edx
2151                                    cfm2:
2152 00003365 E2F9                             loop    cfm1
2153 00003367 59                               pop     ecx
2154 00003368 E2EF                             loop    cfm0
2155 0000336A C3                               retn
2156
2157                                        %include 'diskio.s'  ; 07/03/2015
   1                                <1> ; ****************************************************************************
   2                                <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - diskio.s
   3                                <1> ; ----------------------------------------------------------------------------
   4                                <1> ; Last Update: 15/01/2017
   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 0000336B 9C                    <1>       pushfd
  31 0000336C 0E                    <1>       push   cs
  32 0000336D E809000000            <1>       call   DISKETTE_IO_1
  33 00003372 C3                    <1>       retn
  34                                <1>
  35                                <1> ;;;;;; DISKETTE I/O ;;;;;;;;;;;;;;;;;;;; 06/02/2015 ;;;
  36                                <1> ;////////////////////////////////////////////////////
  37                                <1>
  38                                <1> ; DISKETTE I/O - Erdogan Tan (Retro UNIX 386 v1 project)
  39                                <1> ; 20/02/2015
  40                                <1> ; 06/02/2015 (unix386.s)
  41                                <1> ; 16/12/2014 - 02/01/2015 (dsectrm2.s)
  42                                <1> ;
  43                                <1> ; Code (DELAY) modifications - AWARD BIOS 1999 (ADISK.EQU, COMMON.MAC)
```

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

```
146                             <1> ;BAD_SEEK    EQU    040H         ; SEEK OPERATION FAILED
147                             <1> BAD_NEC      EQU    020H         ; DISKETTE CONTROLLER HAS FAILED
148                             <1> BAD_CRC      EQU    010H         ; BAD CRC ON DISKETTE READ
149                             <1> MED_NOT_FND  EQU    00CH         ; MEDIA TYPE NOT FOUND
150                             <1> DMA_BOUNDARY EQU    009H         ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
151                             <1> BAD_DMA      EQU    008H         ; DMA OVERRUN ON OPERATION
152                             <1> MEDIA_CHANGE EQU    006H         ; MEDIA REMOVED ON DUAL ATTACH CARD
153                             <1> RECORD_NOT_FND      EQU    004H         ; REQUESTED SECTOR NOT FOUND
154                             <1> WRITE_PROTECT       EQU    003H         ; WRITE ATTEMPTED ON WRITE PROTECT DISK
155                             <1> BAD_ADDR_MARK       EQU    002H         ; ADDRESS MARK NOT FOUND
156                             <1> BAD_CMD      EQU    001H         ; BAD COMMAND PASSED TO DISKETTE I/O
157                             <1>
158                             <1> ;---------- DISK CHANGE LINE EQUATES -----------------------------------------
159                             <1> NOCHGLN      EQU    001H         ; NO DISK CHANGE LINE AVAILABLE
160                             <1> CHGLN        EQU    002H         ; DISK CHANGE LINE AVAILABLE
161                             <1>
162                             <1> ;---------- MEDIA/DRIVE STATE INDICATORS -------------------------------------
163                             <1> TRK_CAPA     EQU    00000001B    ; 80 TRACK CAPABILITY
164                             <1> FMT_CAPA     EQU    00000010B    ; MULTIPLE FORMAT CAPABILITY (1.2M)
165                             <1> DRV_DET      EQU    00000100B    ; DRIVE DETERMINED
166                             <1> MED_DET      EQU    00010000B    ; MEDIA DETERMINED BIT
167                             <1> DBL_STEP     EQU    00100000B    ; DOUBLE STEP BIT
168                             <1> RATE_MSK     EQU    11000000B    ; MASK FOR CLEARING ALL BUT RATE
169                             <1> RATE_500     EQU    00000000B    ; 500 KBS DATA RATE
170                             <1> RATE_300     EQU    01000000B    ; 300 KBS DATA RATE
171                             <1> RATE_250     EQU    10000000B    ; 250 KBS DATA RATE
172                             <1> STRT_MSK     EQU    00001100B    ; OPERATION START RATE MASK
173                             <1> SEND_MSK     EQU    11000000B    ; MASK FOR SEND RATE BITS
174                             <1>
175                             <1> ;---------- MEDIA/DRIVE STATE INDICATORS COMPATIBILITY ------------------------
176                             <1> M3D3U        EQU    00000000B    ; 360 MEDIA/DRIVE NOT ESTABLISHED
177                             <1> M3D1U        EQU    00000001B    ; 360 MEDIA,1.2DRIVE NOT ESTABLISHED
178                             <1> M1D1U        EQU    00000010B    ; 1.2 MEDIA/DRIVE NOT ESTABLISHED
179                             <1> MED_UNK      EQU    00000111B    ; NONE OF THE ABOVE
180                             <1>
181                             <1> ;---------- INTERRUPT EQUATES -------------------------------------------------
182                             <1> ;EOI         EQU    020H         ; END OF INTERRUPT COMMAND TO 8259
183                             <1> ;INTA00             EQU    020H         ; 8259 PORT
184                             <1> INTA01              EQU    021H         ; 8259 PORT
185                             <1> INTB00              EQU    0A0H         ; 2ND 8259
186                             <1> INTB01              EQU    0A1H         ;
187                             <1>
188                             <1> ;-----------------------------------------------------------------------------
189                             <1> DMA08        EQU    008H         ; DMA STATUS REGISTER PORT ADDRESS
190                             <1> DMA          EQU    000H         ; DMA CH.0 ADDRESS REGISTER PORT ADDRESS
191                             <1> DMA18        EQU    0D0H         ; 2ND DMA STATUS PORT ADDRESS
192                             <1> DMA1         EQU    0C0H         ; 2ND DMA CH.0 ADDRESS REGISTER ADDRESS
193                             <1> ;-----------------------------------------------------------------------------
194                             <1> ;TIMER              EQU    040H         ; 8254 TIMER - BASE ADDRESS
195                             <1>
196                             <1> ;-----------------------------------------------------------------------------
197                             <1> DMA_PAGE     EQU    081H         ; START OF DMA PAGE REGISTERS
198                             <1>
199                             <1> ; 06/02/2015 (unix386.s, protected mode modifications)
200                             <1> ; (unix386.s <-- dsectrm2.s)
201                             <1> ; 11/12/2014 (copy from IBM PC-XT Model 286 BIOS - DSEG.INC)
202                             <1>
203                             <1> ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)
204                             <1> ; 10/12/2014
205                             <1> ;
206                             <1> ;int40h:
207                             <1> ;    pushf
208                             <1> ;    push   cs
209                             <1> ;    ;cli
210                             <1> ;    call   DISKETTE_IO_1
211                             <1> ;    retn
212                             <1>
213                             <1> ; DSKETTE ----- 04/21/86 DISKETTE BIOS
214                             <1> ; (IBM PC XT Model 286 System BIOS Source Code, 04-21-86)
215                             <1> ;
216                             <1>
217                             <1> ;-- INT13H ----------------------------------------------------------------------
218                             <1> ; DISKETTE I/O
219                             <1> ;    THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4 INCH 360 KB,
220                             <1> ;    1.2 MB, 720 KB AND 1.44 MB DISKETTE DRIVES.
221                             <1> ; INPUT
222                             <1> ;    (AH) =  00H RESET DISKETTE SYSTEM
223                             <1> ;            HARD RESET TO NEC, PREPARE COMMAND, RECALIBRATE REQUIRED
224                             <1> ;            ON ALL DRIVES
225                             <1> ;-----------------------------------------------------------------------------
226                             <1> ;    (AH)= 01H  READ THE STATUS OF THE SYSTEM INTO (AH)
227                             <1> ;            @DISKETTE_STATUS FROM LAST OPERATION IS USED
228                             <1> ;-----------------------------------------------------------------------------
229                             <1> ;    REGISTERS FOR READ/WRITE/VERIFY/FORMAT
230                             <1> ;    (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
231                             <1> ;    (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
232                             <1> ;    (CH) - TRACK NUMBER (NOT VALUE CHECKED)
233                             <1> ;            MEDIA  DRIVE  TRACK NUMBER
234                             <1> ;            320/360      320/360          0-39
235                             <1> ;            320/360      1.2M       0-39
236                             <1> ;            1.2M   1.2M       0-79
237                             <1> ;            720K   720K       0-79
238                             <1> ;            1.44M  1.44M      0-79
239                             <1> ;    (CL) -       SECTOR NUMBER (NOT VALUE CHECKED, NOT USED FOR FORMAT)
240                             <1> ;            MEDIA  DRIVE  SECTOR NUMBER
241                             <1> ;            320/360      320/360           1-8/9
242                             <1> ;            320/360      1.2M        1-8/9
243                             <1> ;            1.2M   1.2M       1-15
244                             <1> ;            720K   720K       1-9
245                             <1> ;            1.44M  1.44M      1-18
246                             <1> ;    (AL)   NUMBER OF SECTORS (NOT VALUE CHECKED)
247                             <1> ;            MEDIA  DRIVE  MAX NUMBER OF SECTORS
```

```
248                             <1> ;           320/360      320/360              8/9
249                             <1> ;           320/360      1.2M          8/9
250                             <1> ;           1.2M   1.2M         15
251                             <1> ;           720K   720K         9
252                             <1> ;           1.44M  1.44M        18
253                             <1> ;
254                             <1> ;    (ES:BX) - ADDRESS OF BUFFER (NOT REQUIRED FOR VERIFY)
255                             <1> ;
256                             <1> ;--------------------------------------------------------------------------
257                             <1> ;    (AH)= 02H  READ THE DESIRED SECTORS INTO MEMORY
258                             <1> ;--------------------------------------------------------------------------
259                             <1> ;    (AH)= 03H  WRITE THE DESIRED SECTORS FROM MEMORY
260                             <1> ;--------------------------------------------------------------------------
261                             <1> ;    (AH)= 04H  VERIFY THE DESIRED SECTORS
262                             <1> ;--------------------------------------------------------------------------
263                             <1> ;    (AH)= 05H  FORMAT THE DESIRED TRACK
264                             <1> ;          (ES,BX) MUST POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS
265                             <1> ;          FOR THE      TRACK. EACH FIELD IS COMPOSED OF 4 BYTES, (C,H,R,N),
266                             <1> ;          WHERE C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER,
267                             <1> ;          N= NUMBER OF BYTES PER SECTOR (00=128,01=256,02=512,03=1024),
268                             <1> ;          THERE MUST BE ONE ENTRY FOR EVERY SECTOR ON THE TRACK.
269                             <1> ;          THIS INFORMATION IS USED TO FIND THE REQUESTED SECTOR DURING
270                             <1> ;          READ/WRITE ACCESS.
271                             <1> ;          PRIOR TO FORMATTING A DISKETTE, IF THERE EXISTS MORE THAN
272                             <1> ;          ONE SUPPORTED MEDIA FORMAT TYPE WITHIN THE DRIVE IN QUESTION,
273                             <1> ;          THEN "SET DASD TYPE" (INT 13H, AH = 17H) OR 'SET MEDIA TYPE'
274                             <1> ;          (INT 13H, AH =  18H) MUST BE CALLED TO SET THE DISKETTE TYPE
275                             <1> ;          THAT IS TO BE FORMATTED. IF "SET DASD TYPE" OR "SET MEDIA TYPE"
276                             <1> ;          IS NOT CALLED, THE FORMAT ROUTINE WILL ASSUME THE
277                             <1> ;          MEDIA FORMAT TO BE THE MAXIMUM CAPACITY OF THE DRIVE.
278                             <1> ;
279                             <1> ;          THESE PARAMETERS OF DISK BASE MUST BE CHANGED IN ORDER TO
280                             <1> ;          FORMAT THE FOLLOWING MEDIAS:
281                             <1> ;          -----------------------------------------
282                             <1> ;          : MEDIA  :     DRIVE      : PARM 1 : PARM 2 :
283                             <1> ;          -----------------------------------------
284                             <1> ;          : 320K  : 320K/360K/1.2M :  50H   :   8    :
285                             <1> ;          : 360K  : 320K/360K/1.2M :  50H   :   9    :
286                             <1> ;          : 1.2M  : 1.2M           :  54H   :  15    :
287                             <1> ;          : 720K  : 720K/1.44M     :  50H   :   9    :
288                             <1> ;          : 1.44M       : 1.44M          :  6CH   :  18    :
289                             <1> ;          -----------------------------------------
290                             <1> ;          NOTES: - PARM 1 = GAP LENGTH FOR FORMAT
291                             <1> ;                 - PARM 2 = EOT (LAST SECTOR ON TRACK)
292                             <1> ;                 - DISK BASE IS POINTED BY DISK POINTER LOCATED
293                             <1> ;                  AT ABSOLUTE ADDRESS 0:78.
294                             <1> ;                 - WHEN FORMAT OPERATIONS ARE COMPLETE, THE PARAMETERS
295                             <1> ;                  SHOULD BE RESTORED TO THEIR RESPECTIVE INITIAL VALUES.
296                             <1> ;--------------------------------------------------------------------------
297                             <1> ;    (AH) = 08H READ DRIVE PARAMETERS
298                             <1> ;    REGISTERS
299                             <1> ;      INPUT
300                             <1> ;        (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
301                             <1> ;         ** 27/05/2016 - TRDOS 386 (TRDOS v2.0) **
302                             <1> ;           ** EBX = Buffer address for floppy disk parameters table **
303                             <1> ;       OUTPUT
304                             <1> ;        (ES:DI) POINTS TO DRIVE PARAMETER TABLE
305                             <1> ;        *** TRDOS 386 note: floppy disk parameter table (16 bytes)
306                             <1> ;        will be returned to user in EBX, buffer address *** 27/05/2016 ***
307                             <1> ;
308                             <1> ;        (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
309                             <1> ;        (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
310                             <1> ;               BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
311                             <1> ;        (DH) - MAXIMUM HEAD NUMBER
312                             <1> ;        (DL) - NUMBER OF DISKETTE DRIVES INSTALLED
313                             <1> ;        (BH) - 0
314                             <1> ;        (BL) - BITS 7 THRU 4 - 0
315                             <1> ;               BITS 3 THRU 0 - VALID DRIVE TYPE VALUE IN CMOS
316                             <1> ;        (AX) - 0
317                             <1> ;     UNDER THE FOLLOWING CIRCUMSTANCES:
318                             <1> ;        (1) THE DRIVE NUMBER IS INVALID,
319                             <1> ;        (2) THE DRIVE TYPE IS UNKNOWN AND CMOS IS NOT PRESENT,
320                             <1> ;        (3) THE DRIVE TYPE IS UNKNOWN AND CMOS IS BAD,
321                             <1> ;        (4) OR THE DRIVE TYPE IS UNKNOWN AND THE CMOS DRIVE TYPE IS INVALID
322                             <1> ;        THEN ES,AX,BX,CX,DH,DI=0 ; DL=NUMBER OF DRIVES.
323                             <1> ;        IF NO DRIVES ARE PRESENT THEN: ES,AX,BX,CX,DX,DI=0.
324                             <1> ;        @DISKETTE_STATUS = 0 AND CY IS RESET.
325                             <1> ;--------------------------------------------------------------------------
326                             <1> ;    (AH)= 15H  READ DASD TYPE
327                             <1> ;    OUTPUT REGISTERS
328                             <1> ;    (AH) - ON RETURN IF CARRY FLAG NOT SET, OTHERWISE ERROR
329                             <1> ;           00 - DRIVE NOT PRESENT
330                             <1> ;           01 - DISKETTE, NO CHANGE LINE AVAILABLE
331                             <1> ;           02 - DISKETTE, CHANGE LINE AVAILABLE
332                             <1> ;           03 - RESERVED (FIXED DISK)
333                             <1> ;    (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
334                             <1> ;--------------------------------------------------------------------------
335                             <1> ;    (AH)= 16H  DISK CHANGE LINE STATUS
336                             <1> ;    OUTPUT REGISTERS
337                             <1> ;    (AH) - 00 - DISK CHANGE LINE NOT ACTIVE
338                             <1> ;           06 - DISK CHANGE LINE ACTIVE & CARRY BIT ON
339                             <1> ;    (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
340                             <1> ;--------------------------------------------------------------------------
341                             <1> ;    (AH)= 17H  SET DASD TYPE FOR FORMAT
342                             <1> ;    INPUT REGISTERS
343                             <1> ;    (AL) - 00 - NOT USED
344                             <1> ;           01 - DISKETTE 320/360K IN 360K DRIVE
345                             <1> ;           02 - DISKETTE 360K IN 1.2M DRIVE
346                             <1> ;           03 - DISKETTE 1.2M IN 1.2M DRIVE
347                             <1> ;           04 - DISKETTE 720K IN 720K DRIVE
348                             <1> ;    (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED:
349                             <1> ;           (DO NOT USE WHEN DISKETTE ATTACH CARD USED)
```

```
                                <1> ;--------------------------------------------------------------------------
                                <1> ;     (AH)= 18H  SET MEDIA TYPE FOR FORMAT
                                <1> ;     INPUT REGISTERS
                                <1> ;     (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM TRACKS
                                <1> ;     (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
                                <1> ;            BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
                                <1> ;     (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHACKED)
                                <1> ;     OUTPUT REGISTERS:
                                <1> ;     (ES:DI) - POINTER TO DRIVE PARAMETERS TABLE FOR THIS MEDIA TYPE,
                                <1> ;              UNCHANGED IF (AH) IS NON-ZERO
                                <1> ;     (AH) - 00H, CY = 0, TRACK AND SECTORS/TRACK COMBINATION IS SUPPORTED
                                <1> ;          - 01H, CY = 1, FUNCTION IS NOT AVAILABLE
                                <1> ;          - 0CH, CY = 1, TRACK AND SECTORS/TRACK COMBINATION IS NOT SUPPORTED
                                <1> ;          - 80H, CY = 1, TIME OUT (DISKETTE NOT PRESENT)
                                <1> ;--------------------------------------------------------------------------
                                <1> ;     DISK CHANGE STATUS IS ONLY CHECKED WHEN A MEDIA SPECIFIED IS OTHER
                                <1> ;     THAN 360 KB DRIVE. IF THE DISK CHANGE LINE IS FOUND TO BE
                                <1> ;     ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:
                                <1> ;            ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.
                                <1> ;            IF ATTEMPT SUCCEEDS SET DASD TYPE FOR FORMAT AND RETURN DISK
                                <1> ;            CHANGE ERROR CODE
                                <1> ;            IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASD TYPE
                                <1> ;            TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.
                                <1> ;     IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASD TYPE FOR FORMAT.
                                <1> ;
                                <1> ; DATA VARIABLE -- @DISK_POINTER
                                <1> ;     DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
                                <1> ;--------------------------------------------------------------------------
                                <1> ; OUTPUT FOR ALL FUNCTIONS
                                <1> ;     AH = STATUS OF OPERATION
                                <1> ;            STATUS BITS ARE DEFINED IN THE EQUATES FOR @DISKETTE_STATUS
                                <1> ;            VARIABLE IN THE DATA SEGMENT OF THIS MODULE
                                <1> ;     CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASD
                                <1> ;            TYPE AH=(15)).
                                <1> ;     CY = 1 FAILED OPERATION (AH HAS ERROR REASON)
                                <1> ;     FOR READ/WRITE/VERIFY
                                <1> ;            DS,BX,DX,CX PRESERVED
                                <1> ;     NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
                                <1> ;            ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
                                <1> ;            ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
                                <1> ;            THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
                                <1> ;            PROBLEM IS NOT DUE TO MOTOR START-UP.
                                <1> ;--------------------------------------------------------------------------
                                <1> ;
                                <1> ; DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)
                                <1> ;
                                <1> ;   ---------------------------------------------------------------
                                <1> ;   |       |       |       |       |       |       |       |       |
                                <1> ;   |   7   |   6   |   5   |   4   |   3   |   2   |   1   |   0   |
                                <1> ;   |       |       |       |       |       |       |       |       |
                                <1> ;   ---------------------------------------------------------------
                                <1> ;     |       |       |       |       |     |     |     |
                                <1> ;     |       |       |       |       |     -----------
                                <1> ;     |       |       |       |       |           |
                                <1> ;     |       |       |       |   RESERVED        |
                                <1> ;     |       |       |       |             PRESENT STATE
                                <1> ;     |       |       |       |     000: 360K IN 360K DRIVE UNESTABLISHED
                                <1> ;     |       |       |       |     001: 360K IN 1.2M DRIVE UNESTABLISHED
                                <1> ;     |       |       |       |     010: 1.2M IN 1.2M DRIVE UNESTABLISHED
                                <1> ;     |       |       |       |     011: 360K IN 360K DRIVE ESTABLISHED
                                <1> ;     |       |       |       |     100: 360K IN 1.2M DRIVE ESTABLISHED
                                <1> ;     |       |       |       |     101: 1.2M IN 1.2M DRIVE ESTABLISHED
                                <1> ;     |       |       |       |     110: RESERVED
                                <1> ;     |       |       |       |     111: NONE OF THE ABOVE
                                <1> ;     |       |       |       |
                                <1> ;     |       |       |     ------>     MEDIA/DRIVE ESTABLISHED
                                <1> ;     |       |       |
                                <1> ;     |       |     -------------->    DOUBLE STEPPING REQUIRED (360K IN 1.2M
                                <1> ;     |       |                   DRIVE)
                                <1> ;     |       |
                                <1> ;     ------------------------------>  DATA TRANSFER RATE FOR THIS DRIVE:
                                <1> ;
                                <1> ;                                       00: 500 KBS
                                <1> ;                                       01: 300 KBS
                                <1> ;                                       10: 250 KBS
                                <1> ;                                       11: RESERVED
                                <1> ;
                                <1> ;--------------------------------------------------------------------------
                                <1> ; STATE OPERATION STARTED - ABSOLUTE ADDRESS 40:92 (DRIVE A) & 93 (DRIVE B)
                                <1> ;--------------------------------------------------------------------------
                                <1> ; PRESENT CYLINDER NUMBER - ABSOLUTE ADDRESS 40:94 (DRIVE A) & 95 (DRIVE B)
                                <1> ;--------------------------------------------------------------------------
                                <1>
                                <1> struc MD
00000000 <res 00000001>         <1>       .SPEC1       resb   1     ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00000001 <res 00000001>         <1>       .SPEC2       resb   1     ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00000002 <res 00000001>         <1>       .OFF_TIM     resb   1     ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00000003 <res 00000001>         <1>       .BYT_SEC     resb   1     ; 512 BYTES/SECTOR
00000004 <res 00000001>         <1>       .SEC_TRK     resb   1     ; EOT (LAST SECTOR ON TRACK)
00000005 <res 00000001>         <1>       .GAP         resb   1     ; GAP LENGTH
00000006 <res 00000001>         <1>       .DTL         resb   1     ; DTL
00000007 <res 00000001>         <1>       .GAP3        resb   1     ; GAP LENGTH FOR FORMAT
00000008 <res 00000001>         <1>       .FIL_BYT     resb   1     ; FILL BYTE FOR FORMAT
00000009 <res 00000001>         <1>       .HD_TIM              resb   1     ; HEAD SETTLE TIME (MILLISECONDS)
0000000A <res 00000001>         <1>       .STR_TIM     resb   1     ; MOTOR START TIME (1/8 SECONDS)
0000000B <res 00000001>         <1>       .MAX_TRK     resb   1     ; MAX. TRACK NUMBER
0000000C <res 00000001>         <1>       .RATE        resb   1     ; DATA TRANSFER RATE
                                <1> endstruc
                                <1>
                                <1> BIT7OFF     EQU    7FH
                                <1> BIT7ON      EQU    80H
```

```
                                <1>
                                <1> ;;int13h: ; 16/02/2015
                                <1> ;; 16/02/2015 - 21/02/2015
                                <1> int40h:
00003373 9C                     <1>       pushfd
00003374 0E                     <1>       push   cs
00003375 E801000000             <1>       call   DISKETTE_IO_1
0000337A C3                     <1>       retn
                                <1>
                                <1> DISKETTE_IO_1:
                                <1>
0000337B FB                     <1>       STI                     ; INTERRUPTS BACK ON
0000337C 55                     <1>       PUSH   eBP              ; USER REGISTER
0000337D 57                     <1>       PUSH   eDI              ; USER REGISTER
0000337E 52                     <1>       PUSH   eDX              ; HEAD #, DRIVE # OR USER REGISTER
0000337F 53                     <1>       PUSH   eBX              ; BUFFER OFFSET PARAMETER OR REGISTER
00003380 51                     <1>       PUSH   eCX              ; TRACK #-SECTOR # OR USER REGISTER
00003381 89E5                   <1>       MOV    eBP,eSP                  ; BP     => PARAMETER LIST DEP. ON AH
                                <1>                               ; [BP]   = SECTOR #
                                <1>                               ; [BP+1] = TRACK #
                                <1>                               ; [BP+2] = BUFFER OFFSET
                                <1>                               ; FOR RETURN OF DRIVE PARAMETERS:
                                <1>                               ; CL/[BP] = BITS 7&6 HI BITS OF MAX CYL
                                <1>                               ;          BITS 0-5 MAX SECTORS/TRACK
                                <1>                               ; CH/[BP+1] = LOW 8 BITS OF MAX CYL.
                                <1>                               ; BL/[BP+2] = BITS 7-4 = 0
                                <1>                               ;            BITS 3-0 = VALID CMOS TYPE
                                <1>                               ; BH/[BP+3] = 0
                                <1>                               ; DL/[BP+4] = # DRIVES INSTALLED
                                <1>                               ; DH/[BP+5] = MAX HEAD #
                                <1>                               ; DI/[BP+6] = OFFSET TO DISK BASE
00003383 06                     <1>       push   es ; 06/02/2015
00003384 1E                     <1>       PUSH   DS               ; BUFFER SEGMENT PARM OR USER REGISTER
00003385 56                     <1>       PUSH   eSI              ; USER REGISTERS
                                <1>       ;CALL  DDS              ; SEGMENT OF BIOS DATA AREA TO DS
                                <1>       ;mov   cx, cs
                                <1>       ;mov   ds, cx
00003386 66B91000               <1>       mov    cx, KDATA
0000338A 8ED9                   <1>         mov     ds, cx
0000338C 8EC1                   <1>         mov     es, cx
                                <1>
                                <1>       ;CMP   AH,(FNC_TAE-FNC_TAB)/2    ; CHECK FOR > LARGEST FUNCTION
0000338E 80FC19                 <1>       cmp    ah,(FNC_TAE-FNC_TAB)/4  ; 18/02/2015
00003391 7202                   <1>       JB     short OK_FUNC       ; FUNCTION OK
00003393 B414                   <1>       MOV    AH,14H              ; REPLACE WITH KNOWN INVALID FUNCTION
                                <1> OK_FUNC:
00003395 80FC01                 <1>       CMP    AH,1                ; RESET OR STATUS ?
00003398 760C                   <1>       JBE    short OK_DRV        ; IF RESET OR STATUS DRIVE ALWAYS OK
0000339A 80FC08                 <1>       CMP    AH,8                ; READ DRIVE PARMS ?
0000339D 7407                   <1>       JZ     short OK_DRV        ; IF SO DRIVE CHECKED LATER
0000339F 80FA01                 <1>       CMP    DL,1                ; DRIVES 0 AND 1 OK
000033A2 7602                   <1>       JBE    short OK_DRV        ; IF 0 OR 1 THEN JUMP
000033A4 B414                   <1>       MOV    AH,14H              ; REPLACE WITH KNOWN INVALID FUNCTION
                                <1> OK_DRV:
000033A6 31C9                   <1>       xor    ecx, ecx
                                <1>       ;mov   esi, ecx ; 08/02/2015
000033A8 89CF                   <1>       mov    edi, ecx ; 08/02/2015
000033AA 88E1                   <1>       MOV    CL,AH               ; CL = FUNCTION
                                <1>       ;XOR   CH,CH               ; CX = FUNCTION
                                <1>       ;SHL   CL, 1               ; FUNCTION TIMES 2
000033AC C0E102                 <1>       SHL    CL, 2 ; 20/02/2015  ; FUNCTION TIMES 4 (for 32 bit offset)
000033AF BB[E7330000]           <1>       MOV    eBX,FNC_TAB         ; LOAD START OF FUNCTION TABLE
000033B4 01CB                   <1>       ADD    eBX,eCX                  ; ADD OFFSET INTO TABLE => ROUTINE
000033B6 88F4                   <1>       MOV    AH,DH               ; AX = HEAD #,# OF SECTORS OR DASD TYPE
000033B8 30F6                   <1>       XOR    DH,DH               ; DX = DRIVE #
000033BA 6689C6                 <1>       MOV    SI,AX               ; SI = HEAD #,# OF SECTORS OR DASD TYPE
000033BD 6689D7                 <1>       MOV     DI,DX                   ; DI = DRIVE #
                                <1>       ;
                                <1>       ; 11/12/2014
000033C0 8815[E55C0000]         <1>         mov     [cfd], dl              ; current floppy drive (for 'GET_PARM')
                                <1>       ;
000033C6 8A25[88520100]         <1>       MOV    AH, [DSKETTE_STATUS]      ; LOAD STATUS TO AH FOR STATUS FUNCTION
000033CC C605[88520100]00       <1>       MOV    byte [DSKETTE_STATUS],0   ; INITIALIZE FOR ALL OTHERS
                                <1>
                                <1> ;     THROUGHOUT THE DISKETTE BIOS, THE FOLLOWING INFORMATION IS CONTAINED IN
                                <1> ;     THE FOLLOWING MEMORY LOCATIONS AND REGISTERS. NOT ALL DISKETTE BIOS
                                <1> ;     FUNCTIONS REQUIRE ALL OF THESE PARAMETERS.
                                <1> ;
                                <1> ;           DI     : DRIVE #
                                <1> ;           SI-HI  : HEAD #
                                <1> ;           SI-LOW : # OF SECTORS OR DASD TYPE FOR FORMAT
                                <1> ;           ES     : BUFFER SEGMENT
                                <1> ;           [BP]   : SECTOR #
                                <1> ;           [BP+1] : TRACK #
                                <1> ;           [BP+2] : BUFFER OFFSET
                                <1> ;
                                <1> ;     ACROSS CALLS TO SUBROUTINES THE CARRY FLAG (CY=1), WHERE INDICATED IN
                                <1> ;     SUBROUTINE PROLOGUES, REPRESENTS AN EXCEPTION RETURN (NORMALLY AN ERROR
                                <1> ;     CONDITION). IN MOST CASES, WHEN CY = 1, @DSKETTE_STATUS CONTAINS THE
                                <1> ;     SPECIFIC ERROR CODE.
                                <1> ;
                                <1>                               ; (AH) = @DSKETTE_STATUS
000033D3 FF13                   <1>       CALL   dWORD [eBX]        ; CALL THE REQUESTED FUNCTION
000033D5 5E                     <1>       POP    eSI                ; RESTORE ALL REGISTERS
000033D6 1F                     <1>       POP    DS
000033D7 07                     <1>       pop    es     ; 06/02/2015
000033D8 59                     <1>       POP    eCX
000033D9 5B                     <1>       POP    eBX
000033DA 5A                     <1>       POP    eDX
000033DB 5F                     <1>       POP    eDI
000033DC 89E5                   <1>       MOV    eBP, eSP
000033DE 50                     <1>       PUSH   eAX
```

```
554 000033DF 9C                 <1>       PUSHFd
555 000033E0 58                 <1>       POP    eAX
556                             <1>       ;MOV   [BP+6], AX
557 000033E1 89450C             <1>       mov    [ebp+12], eax  ; 18/02/2015, flags
558 000033E4 58                 <1>       POP    eAX
559 000033E5 5D                 <1>       POP    eBP
560 000033E6 CF                 <1>       IRETd
561                             <1>
562                             <1> ;--------------------------------------------------------------------------------
563                             <1> ; DW --> dd (06/02/2015)
564 000033E7 [4B340000]         <1> FNC_TAB     dd     DSK_RESET           ; AH = 00H; RESET
565 000033EB [C4340000]         <1>       dd     DSK_STATUS          ; AH = 01H; STATUS
566 000033EF [D5340000]         <1>       dd     DSK_READ            ; AH = 02H; READ
567 000033F3 [E6340000]         <1>       dd     DSK_WRITE           ; AH = 03H; WRITE
568 000033F7 [F7340000]         <1>       dd     DSK_VERF            ; AH = 04H; VERIFY
569 000033FB [08350000]         <1>       dd     DSK_FORMAT          ; AH = 05H; FORMAT
570 000033FF [8D350000]         <1>       dd     FNC_ERR                   ; AH = 06H; INVALID
571 00003403 [8D350000]         <1>       dd     FNC_ERR                   ; AH = 07H; INVALID
572 00003407 [9A350000]         <1>       dd     DSK_PARMS           ; AH = 08H; READ DRIVE PARAMETERS
573 0000340B [8D350000]         <1>       dd     FNC_ERR                   ; AH = 09H; INVALID
574 0000340F [8D350000]         <1>       dd     FNC_ERR                   ; AH = 0AH; INVALID
575 00003413 [8D350000]         <1>       dd     FNC_ERR                   ; AH = 0BH; INVALID
576 00003417 [8D350000]         <1>       dd     FNC_ERR                   ; AH = 0CH; INVALID
577 0000341B [8D350000]         <1>       dd     FNC_ERR                   ; AH = 0DH; INVALID
578 0000341F [8D350000]         <1>       dd     FNC_ERR                   ; AH = 0EH; INVALID
579 00003423 [8D350000]         <1>       dd     FNC_ERR                   ; AH = 0FH; INVALID
580 00003427 [8D350000]         <1>       dd     FNC_ERR                   ; AH = 10H; INVALID
581 0000342B [8D350000]         <1>       dd     FNC_ERR                   ; AH = 11H; INVALID
582 0000342F [8D350000]         <1>       dd     FNC_ERR                   ; AH = 12H; INVALID
583 00003433 [8D350000]         <1>       dd     FNC_ERR                   ; AH = 13H; INVALID
584 00003437 [8D350000]         <1>       dd     FNC_ERR                   ; AH = 14H; INVALID
585 0000343B [72360000]         <1>       dd     DSK_TYPE            ; AH = 15H; READ DASD TYPE
586 0000343F [9D360000]         <1>       dd     DSK_CHANGE          ; AH = 16H; CHANGE STATUS
587 00003443 [D7360000]         <1>       dd     FORMAT_SET          ; AH = 17H; SET DASD TYPE
588 00003447 [5A370000]         <1>       dd     SET_MEDIA           ; AH = 18H; SET MEDIA TYPE
589                             <1> FNC_TAE EQU     $                         ; END
590                             <1>
591                             <1> ;--------------------------------------------------------------------------------
592                             <1> ; DISK_RESET(AH = 00H)
593                             <1> ;           RESET THE DISKETTE SYSTEM.
594                             <1> ;
595                             <1> ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
596                             <1> ;--------------------------------------------------------------------------------
597                             <1> DSK_RESET:
598 0000344B 66BAF203           <1>       MOV    DX,03F2H            ; ADAPTER CONTROL PORT
599 0000344F FA                 <1>       CLI                        ; NO INTERRUPTS
600 00003450 A0[86520100]       <1>       MOV    AL,[MOTOR_STATUS]   ; GET DIGITAL OUTPUT REGISTER REFLECTION
601 00003455 243F               <1>       AND    AL,00111111B        ; KEEP SELECTED AND MOTOR ON BITS
602 00003457 C0C004             <1>       ROL    AL,4                ; MOTOR VALUE TO HIGH NIBBLE
603                             <1>                                  ; DRIVE SELECT TO LOW NIBBLE
604 0000345A 0C08               <1>       OR     AL,00001000B        ; TURN ON INTERRUPT ENABLE
605 0000345C EE                 <1>       OUT    DX,AL               ; RESET THE ADAPTER
606 0000345D C605[85520100]00   <1>       MOV    byte [SEEK_STATUS],0      ; SET RECALIBRATE REQUIRED ON ALL DRIVES
607                             <1>       ;JMP   $+2                 ; WAIT FOR I/O
608                             <1>       ;JMP   $+2                 ; WAIT FOR I/O (TO INSURE MINIMUM
609                             <1>                                  ;      PULSE WIDTH)
610                             <1>       ; 19/12/2014
611                             <1>       NEWIODELAY
611 00003464 E6EB               <2>  out 0ebh,al
612                             <1>
613                             <1>       ; 17/12/2014
614                             <1>       ; AWARD BIOS 1999 - RESETDRIVES (ADISK.ASM)
615 00003466 B915000000         <1>       mov    ecx, WAITCPU_RESET_ON     ; cx = 21 -- Min. 14 micro seconds !?
616                             <1> wdw1:
617                             <1>       NEWIODELAY   ; 27/02/2015
617 0000346B E6EB               <2>  out 0ebh,al
618 0000346D E2FC               <1>       loop   wdw1
619                             <1>       ;
620 0000346F 0C04               <1>       OR     AL,00000100B        ; TURN OFF RESET BIT
621 00003471 EE                 <1>       OUT    DX,AL               ; RESET THE ADAPTER
622                             <1>       ; 16/12/2014
623                             <1>       IODELAY
623 00003472 EB00               <2>  jmp short $+2
623 00003474 EB00               <2>  jmp short $+2
624                             <1>       ;
625                             <1>       ;STI                        ; ENABLE THE INTERRUPTS
626 00003476 E83C0C0000         <1>       CALL   WAIT_INT            ; WAIT FOR THE INTERRUPT
627 0000347B 723E               <1>       JC     short DR_ERR        ; IF ERROR, RETURN IT
628 0000347D 66B9C000           <1>       MOV    CX,11000000B        ; CL = EXPECTED @NEC_STATUS
629                             <1> NXT_DRV:
630 00003481 6651               <1>       PUSH   CX                  ; SAVE FOR CALL
631 00003483 B8[B9340000]       <1>       MOV    eAX, DR_POP_ERR     ; LOAD NEC_OUTPUT ERROR ADDRESS
632 00003488 50                 <1>       PUSH   eAX                 ; "
633 00003489 B408               <1>       MOV    AH,08H              ; SENSE INTERRUPT STATUS COMMAND
634 0000348B E81A0B0000         <1>       CALL   NEC_OUTPUT
635 00003490 58                 <1>       POP    eAX                 ; THROW AWAY ERROR RETURN
636 00003491 E8510C0000         <1>       CALL   RESULTS                   ; READ IN THE RESULTS
637 00003496 6659               <1>       POP    CX                  ; RESTORE AFTER CALL
638 00003498 7221               <1>       JC     short DR_ERR        ; ERROR RETURN
639 0000349A 3A0D[89520100]     <1>       CMP    CL, [NEC_STATUS]    ; TEST FOR DRIVE READY TRANSITION
640 000034A0 7519               <1>       JNZ    short DR_ERR        ; EVERYTHING OK
641 000034A2 FEC1               <1>       INC    CL                  ; NEXT EXPECTED @NEC_STATUS
642 000034A4 80F9C3             <1>       CMP    CL,11000011B        ; ALL POSSIBLE DRIVES CLEARED
643 000034A7 76D8               <1>       JBE    short NXT_DRV       ; FALL THRU IF 11000100B OR >
644                             <1>       ;
645 000034A9 E869030000         <1>       CALL   SEND_SPEC           ; SEND SPECIFY COMMAND TO NEC
646                             <1> RESBAC:
647 000034AE E81D090000         <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
648 000034B3 6689F3             <1>       MOV    BX,SI               ; GET SAVED AL TO BL
649 000034B6 88D8               <1>       MOV    AL,BL               ; PUT BACK FOR RETURN
650 000034B8 C3                 <1>       RETn
651                             <1> DR_POP_ERR:
```

```
000034B9 6659                <1>       POP    CX                  ; CLEAR STACK
                             <1> DR_ERR:
000034BB 800D[88520100]20    <1>       OR     byte [DSKETTE_STATUS],BAD_NEC ; SET ERROR CODE
000034C2 EBEA                <1>       JMP    SHORT RESBAC        ; RETURN FROM RESET
                             <1>
                             <1> ;--------------------------------------------------------------------------------
                             <1> ; DISK_STATUS      (AH = 01H)
                             <1> ;     DISKETTE STATUS.
                             <1> ;
                             <1> ; ON ENTRY:  AH : STATUS OF PREVIOUS OPERATION
                             <1> ;
                             <1> ; ON EXIT:   AH, @DSKETTE_STATUS, CY REFLECT STATUS OF PREVIOUS OPERATION.
                             <1> ;--------------------------------------------------------------------------------
                             <1> DSK_STATUS:
000034C4 8825[88520100]      <1>       MOV    [DSKETTE_STATUS],AH ; PUT BACK FOR SETUP END
000034CA E801090000          <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
000034CF 6689F3              <1>       MOV    BX,SI               ; GET SAVED AL TO BL
000034D2 88D8                <1>       MOV    AL,BL               ; PUT BACK FOR RETURN
000034D4 C3                  <1>       RETn
                             <1>
                             <1> ;--------------------------------------------------------------------------------
                             <1> ; DISK_READ  (AH = 02H)
                             <1> ;     DISKETTE READ.
                             <1> ;
                             <1> ; ON ENTRY:  DI     : DRIVE #
                             <1> ;            SI-HI  : HEAD #
                             <1> ;            SI-LOW : # OF SECTORS
                             <1> ;            ES     : BUFFER SEGMENT
                             <1> ;            [BP]   : SECTOR #
                             <1> ;            [BP+1] : TRACK #
                             <1> ;            [BP+2] : BUFFER OFFSET
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;--------------------------------------------------------------------------------
                             <1>
                             <1> ; 06/02/2015, ES:BX -> EBX (unix386.s)
                             <1>
                             <1> DSK_READ:
000034D5 8025[86520100]7F    <1>       AND    byte [MOTOR_STATUS],01111111B ; INDICATE A READ OPERATION
000034DC 66B846E6            <1>       MOV    AX,0E646H           ; AX = NEC COMMAND, DMA COMMAND
000034E0 E83C040000          <1>       CALL   RD_WR_VF            ; COMMON READ/WRITE/VERIFY
000034E5 C3                  <1>       RETn
                             <1>
                             <1> ;--------------------------------------------------------------------------------
                             <1> ; DISK_WRITE(AH = 03H)
                             <1> ;     DISKETTE WRITE.
                             <1> ;
                             <1> ; ON ENTRY:  DI     : DRIVE #
                             <1> ;            SI-HI  : HEAD #
                             <1> ;            SI-LOW : # OF SECTORS
                             <1> ;            ES     : BUFFER SEGMENT
                             <1> ;            [BP]   : SECTOR #
                             <1> ;            [BP+1] : TRACK #
                             <1> ;            [BP+2] : BUFFER OFFSET
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;--------------------------------------------------------------------------------
                             <1>
                             <1> ; 06/02/2015, ES:BX -> EBX (unix386.s)
                             <1>
                             <1> DSK_WRITE:
000034E6 66B84AC5            <1>       MOV    AX,0C54AH           ; AX = NEC COMMAND, DMA COMMAND
000034EA 800D[86520100]80    <1>         OR      byte [MOTOR_STATUS],10000000B ; INDICATE WRITE OPERATION
000034F1 E82B040000          <1>       CALL   RD_WR_VF            ; COMMON READ/WRITE/VERIFY
000034F6 C3                  <1>       RETn
                             <1>
                             <1> ;--------------------------------------------------------------------------------
                             <1> ; DISK_VERF  (AH = 04H)
                             <1> ;     DISKETTE VERIFY.
                             <1> ;
                             <1> ; ON ENTRY:  DI     : DRIVE #
                             <1> ;            SI-HI  : HEAD #
                             <1> ;            SI-LOW : # OF SECTORS
                             <1> ;            ES     : BUFFER SEGMENT
                             <1> ;            [BP]   : SECTOR #
                             <1> ;            [BP+1] : TRACK #
                             <1> ;            [BP+2] : BUFFER OFFSET
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;--------------------------------------------------------------------------------
                             <1> DSK_VERF:
000034F7 8025[86520100]7F    <1>       AND    byte [MOTOR_STATUS],01111111B ; INDICATE A READ OPERATION
000034FE 66B842E6            <1>       MOV    AX,0E642H           ; AX = NEC COMMAND, DMA COMMAND
00003502 E81A040000          <1>       CALL   RD_WR_VF            ; COMMON READ/WRITE/VERIFY
00003507 C3                  <1>       RETn
                             <1>
                             <1> ;--------------------------------------------------------------------------------
                             <1> ; DISK_FORMAT      (AH = 05H)
                             <1> ;     DISKETTE FORMAT.
                             <1> ;
                             <1> ; ON ENTRY:  DI     : DRIVE #
                             <1> ;            SI-HI  : HEAD #
                             <1> ;            SI-LOW : # OF SECTORS
                             <1> ;            ES     : BUFFER SEGMENT
                             <1> ;            [BP]   : SECTOR #
                             <1> ;            [BP+1] : TRACK #
                             <1> ;            [BP+2] : BUFFER OFFSET
                             <1> ;            @DISK_POINTER POINTS TO THE PARAMETER TABLE OF THIS DRIVE
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;--------------------------------------------------------------------------------
                             <1> DSK_FORMAT:
```

```
754 00003508 E853030000          <1>       CALL   XLAT_NEW           ; TRANSLATE STATE TO PRESENT ARCH.
755 0000350D E84F050000          <1>       CALL   FMT_INIT           ; ESTABLISH STATE IF UNESTABLISHED
756 00003512 800D[86520100]80    <1>         OR      byte [MOTOR_STATUS], 10000000B ; INDICATE WRITE OPERATION
757 00003519 E897050000          <1>       CALL   MED_CHANGE         ; CHECK MEDIA CHANGE AND RESET IF SO
758 0000351E 725D                <1>         JC      short FM_DON            ; MEDIA CHANGED, SKIP
759 00003520 E8F2020000          <1>       CALL   SEND_SPEC          ; SEND SPECIFY COMMAND TO NEC
760 00003525 E8FD050000          <1>       CALL   CHK_LASTRATE       ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
761 0000352A 7405                <1>         JZ      short FM_WR             ; YES, SKIP SPECIFY COMMAND
762 0000352C E8D4050000          <1>       CALL   SEND_RATE          ; SEND DATA RATE TO CONTROLLER
763                              <1> FM_WR:
764 00003531 E88A060000          <1>       CALL   FMTDMA_SET         ; SET UP THE DMA FOR FORMAT
765 00003536 7245                <1>         JC      short FM_DON            ; RETURN WITH ERROR
766 00003538 B44D                <1>       MOV    AH,04DH                 ; ESTABLISH THE FORMAT COMMAND
767 0000353A E8E7060000          <1>       CALL   NEC_INIT           ; INITIALIZE THE NEC
768 0000353F 723C                <1>         JC      short FM_DON            ; ERROR - EXIT
769 00003541 B8[7D350000]        <1>         MOV     eAX, FM_DON             ; LOAD ERROR ADDRESS
770 00003546 50                  <1>       PUSH   eAX                ; PUSH NEC_OUT ERROR RETURN
771 00003547 B203                <1>       MOV    DL,3               ; BYTES/SECTOR VALUE TO NEC
772 00003549 E856090000          <1>       CALL   GET_PARM
773 0000354E E8570A0000          <1>       CALL   NEC_OUTPUT
774 00003553 B204                <1>       MOV    DL,4               ; SECTORS/TRACK VALUE TO NEC
775 00003555 E84A090000          <1>       CALL   GET_PARM
776 0000355A E84B0A0000          <1>       CALL   NEC_OUTPUT
777 0000355F B207                <1>       MOV    DL,7               ; GAP LENGTH VALUE TO NEC
778 00003561 E83E090000          <1>       CALL   GET_PARM
779 00003566 E83F0A0000          <1>       CALL   NEC_OUTPUT
780 0000356B B208                <1>       MOV    DL,8               ; FILLER BYTE TO NEC
781 0000356D E832090000          <1>       CALL   GET_PARM
782 00003572 E8330A0000          <1>       CALL   NEC_OUTPUT
783 00003577 58                  <1>       POP    eAX                ; THROW AWAY ERROR
784 00003578 E827070000          <1>       CALL   NEC_TERM           ; TERMINATE, RECEIVE STATUS, ETC,
785                              <1> FM_DON:
786 0000357D E80F030000          <1>       CALL   XLAT_OLD           ; TRANSLATE STATE TO COMPATIBLE MODE
787 00003582 E849080000          <1>       CALL   SETUP_END          ; VARIOUS CLEANUPS
788 00003587 6689F3              <1>       MOV    BX,SI              ; GET SAVED AL TO BL
789 0000358A 88D8                <1>       MOV    AL,BL              ; PUT BACK FOR RETURN
790 0000358C C3                  <1>       RETn
791                              <1>
792                              <1> ;--------------------------------------------------------------------------------
793                              <1> ; FNC_ERR
794                              <1> ;     INVALID FUNCTION REQUESTED OR INVALID DRIVE:
795                              <1> ;     SET BAD COMMAND IN STATUS.
796                              <1> ;
797                              <1> ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
798                              <1> ;--------------------------------------------------------------------------------
799                              <1> FNC_ERR:                       ; INVALID FUNCTION REQUEST
800 0000358D 6689F0              <1>       MOV    AX,SI              ; RESTORE AL
801 00003590 B401                <1>       MOV    AH,BAD_CMD         ; SET BAD COMMAND ERROR
802 00003592 8825[88520100]      <1>       MOV    [DSKETTE_STATUS],AH ; STORE IN DATA AREA
803 00003598 F9                  <1>       STC                       ; SET CARRY INDICATING ERROR
804 00003599 C3                  <1>       RETn
805                              <1>
806                              <1> ; 01/06/2016
807                              <1> ; 28/05/2016
808                              <1> ; 27/05/2016 - TRDOS 386 (TRDOS v.2.0)
809                              <1> ;--------------------------------------------------------------------------------
810                              <1> ; DISK_PARMS (AH = 08H)
811                              <1> ;     READ DRIVE PARAMETERS.
812                              <1> ;
813                              <1> ; ON ENTRY:  DI : DRIVE #
814                              <1> ;            ; 27/05/2016
815                              <1> ;            EBX = Buffer Address for floppy disk parameters table (16 bytes)
816                              <1> ;
817                              <1> ; ON EXIT:   CL/[BP]   = BITS 7 & 6 HI 2 BITS OF MAX CYLINDER
818                              <1> ;                        BITS 0-5 MAX SECTORS/TRACK
819                              <1> ;            CH/[BP+1] = LOW 8 BITS OF MAX CYLINDER
820                              <1> ;            BL/[BP+2] = BITS 7-4 = 0
821                              <1> ;                        BITS 3-0 = VALID CMOS DRIVE TYPE
822                              <1> ;            BH/[BP+3] = 0
823                              <1> ;            DL/[BP+4] = # DRIVES INSTALLED (VALUE CHECKED)
824                              <1> ;            DH/[BP+5] = MAX HEAD #
825                              <1> ;           ** 27/05/2016 - TRDOS 386 (TRDOS v2.0) **
826                              <1> ;             ** EBX = Buffer address for floppy disk parameters table **
827                              <1> ;            ;DI/[BP+6] = OFFSET TO DISK_BASE
828                              <1> ;            ;ES        = SEGMENT OF DISK_BASE
829                              <1> ;
830                              <1> ;            AX        = 0
831                              <1> ;
832                              <1> ;            NOTE : THE ABOVE INFORMATION IS STORED IN THE USERS STACK AT
833                              <1> ;                   THE LOCATIONS WHERE THE MAIN ROUTINE WILL POP THEM
834                              <1> ;                   INTO THE APPROPRIATE REGISTERS BEFORE RETURNING TO THE
835                              <1> ;                   CALLER.
836                              <1> ;--------------------------------------------------------------------------------
837                              <1> DSK_PARMS:
838 0000359A E8C1020000          <1>        CALL   XLAT_NEW           ; TRANSLATE STATE TO PRESENT ARCH,
839                              <1>      ;        MOV    WORD [BP+2],0       ; DRIVE TYPE = 0
840                              <1>      ;  MOV     AX, [EQUIP_FLAG]         ; LOAD EQUIPMENT FLAG FOR # DISKETTES
841                              <1>      ;  AND     AL,11000001B             ; KEEP DISKETTE DRIVE BITS
842                              <1>      ;  MOV     DL,2                     ; DISKETTE DRIVES = 2
843                              <1>      ;  CMP     AL,01000001B             ; 2 DRIVES INSTALLED ?
844                              <1>      ;  JZ      short STO_DL             ; IF YES JUMP
845                              <1>      ;  DEC     DL                       ; DISKETTE DRIVES = 1
846                              <1>      ;  CMP     AL,00000001B             ; 1 DRIVE INSTALLED ?
847                              <1>      ;  JNZ     short NON_DRV            ; IF NO JUMP
848 0000359F 29D2                <1>       sub    edx, edx
849 000035A1 66A1[F65C0000]      <1>       mov     ax, [fd0_type]
850 000035A7 6621C0              <1>       and     ax, ax
851 000035AA 0F848A000000        <1>         jz      NON_DRV
852 000035B0 FEC2                <1>       inc     dl
853 000035B2 20E4                <1>       and     ah, ah
854 000035B4 7402                <1>       jz      short STO_DL
855 000035B6 FEC2                <1>       inc     dl
```

```
                        <1> STO_DL:
                        <1>       ;MOV   [BP+4],DL          ; STORE NUMBER OF DRIVES
000035B8 895508         <1>       mov    [ebp+8], edx ; 20/02/2015
000035BB 6683FF01       <1>       CMP    DI,1               ; CHECK FOR VALID DRIVE
000035BF 777C           <1>       JA     short NON_DRV1           ; DRIVE INVALID
                        <1>       ;MOV   BYTE [BP+5],1      ; MAXIMUM HEAD NUMBER =   1
000035C1 C6450901       <1>       mov    byte [ebp+9], 1  ; 20/02/2015
000035C5 E8D1080000     <1>       CALL   CMOS_TYPE          ; RETURN DRIVE TYPE IN AL
                        <1>       ;;20/02/2015
                        <1>       ;;JC   short CHK_EST      ; IF CMOS BAD CHECKSUM ESTABLISHED
                        <1>       ;;OR   AL,AL              ; TEST FOR NO DRIVE TYPE
000035CA 740F           <1>       JZ     short CHK_EST      ; JUMP IF SO
000035CC E81B020000     <1>       CALL   DR_TYPE_CHECK      ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
000035D1 7208           <1>       JC     short CHK_EST      ; TYPE NOT IN TABLE (POSSIBLE BAD CMOS)
                        <1>       ;MOV   [BP+2],AL          ; STORE VALID CMOS DRIVE TYPE
                        <1>         ;mov [ebp+4], al ; 06/02/2015
000035D3 8A4B04         <1>       MOV     CL, [eBX+MD.SEC_TRK]     ; GET SECTOR/TRACK
000035D6 8A6B0B         <1>         MOV     CH, [eBX+MD.MAX_TRK]     ; GET MAX. TRACK NUMBER
000035D9 EB36           <1>       JMP    SHORT STO_CX       ; CMOS GOOD, USE CMOS
                        <1> CHK_EST:
000035DB 8AA7[95520100] <1>       MOV    AH, [DSK_STATE+eDI] ; LOAD STATE FOR THIS DRIVE
000035E1 F6C410         <1>       TEST   AH,MED_DET         ; CHECK FOR ESTABLISHED STATE
000035E4 7457           <1>       JZ     short NON_DRV1           ; CMOS BAD/INVALID OR UNESTABLISHED
                        <1> USE_EST:
000035E6 80E4C0         <1>       AND    AH,RATE_MSK        ; ISOLATE STATE
000035E9 80FC80         <1>       CMP    AH,RATE_250        ; RATE 250 ?
000035EC 7570           <1>       JNE    short USE_EST2           ; NO, GO CHECK OTHER RATE
                        <1>
                        <1> ;-----      DATA RATE IS 250 KBS, TRY 360 KB TABLE FIRST
                        <1>
000035EE B001           <1>       MOV    AL,01              ; DRIVE TYPE 1 (360KB)
000035F0 E8F7010000     <1>       CALL   DR_TYPE_CHECK      ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
000035F5 8A4B04         <1>         MOV     CL, [eBX+MD.SEC_TRK]    ; GET SECTOR/TRACK
000035F8 8A6B0B         <1>         MOV     CH, [eBX+MD.MAX_TRK]    ; GET MAX. TRACK NUMBER
000035FB F687[95520100]01 <1>     TEST   byte [DSK_STATE+eDI],TRK_CAPA ; 80 TRACK ?
00003602 740D           <1>       JZ     short STO_CX       ; MUST BE 360KB DRIVE
                        <1>
                        <1> ;-----      IT IS 1.44 MB DRIVE
                        <1>
                        <1> PARM144:
00003604 B004           <1>       MOV    AL,04              ; DRIVE TYPE 4 (1.44MB)
00003606 E8E1010000     <1>       CALL   DR_TYPE_CHECK      ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
0000360B 8A4B04         <1>         MOV     CL, [eBX+MD.SEC_TRK]    ; GET SECTOR/TRACK
0000360E 8A6B0B         <1>         MOV     CH, [eBX+MD.MAX_TRK]    ; GET MAX. TRACK NUMBER
                        <1> STO_CX:
00003611 894D00         <1>       MOV    [eBP],eCX          ; SAVE POINTER IN STACK FOR RETURN
                        <1> ES_DI:
                        <1>       ;MOV   [BP+6],BX          ; ADDRESS OF MEDIA/DRIVE PARM TABLE
                        <1>       ;mov   [ebp+12], ebx ; 06/02/2015
                        <1>       ;MOV   AX,CS              ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
                        <1>       ;MOV   ES,AX              ; ES IS SEGMENT OF TABLE
                        <1>       ;
                        <1>       ; 28/05/2016
                        <1>       ; 27/05/2016
                        <1>       ; return floppy disk parameters table to user
                        <1>       ; in user's buffer, which is pointed by EBX
                        <1>       ;
00003614 57             <1>       push   edi
00003615 8B7D04         <1>       mov    edi, [ebp+4]             ; ebx (input), user's buffer address
00003618 0FB6C0         <1>       movzx  eax, al
0000361B 894504         <1>         mov  [ebp+4], eax   ; ebx      ; drive type (for floppy drives)
                        <1>       ; 01/06/2016 (INT 33h, disk type return for floppy disks, in BL)
0000361E A3[905F0100]   <1>       mov    [user_buffer], eax  ; 01/06/2016 (overwrite ebx return value)
                        <1>       ;(INT 33h, Function 08h will replace user's buffer addr with disk type!)
                        <1>       ;
00003623 89DE           <1>       mov    esi, ebx           ; floppy disk parameter table (16 bytes)
00003625 B910000000     <1>       mov    ecx, 16 ; 16 bytes
0000362A E857B20000     <1>         call    transfer_to_user_buffer ; trdosk6.s (16/05/2016)
0000362F 5F             <1>       pop    edi
                        <1> DP_OUT:
00003630 E85C020000     <1>       CALL   XLAT_OLD           ; TRANSLATE STATE TO COMPATIBLE MODE
00003635 6631C0         <1>       XOR    AX,AX              ; CLEAR
00003638 F8             <1>       CLC
00003639 C3             <1>       RETn
                        <1>
                        <1> ;-----      NO DRIYE PRESENT HANDLER
                        <1>
                        <1> NON_DRV:
                        <1>       ;MOV   BYTE [BP+4],0      ; CLEAR NUMBER OF DRIVES
0000363A 895508         <1>       mov    [ebp+8], edx ; 0 ; 20/02/2015
                        <1> NON_DRV1:
0000363D 6681FF8000     <1>       CMP    DI,80H             ; CHECK FOR FIXED MEDIA TYPE REQUEST
00003642 720C           <1>       JB     short NON_DRV2           ; CONTINUE IF NOT REQUEST FALL THROUGH
                        <1>
                        <1> ;-----      FIXED DISK REQUEST FALL THROUGH ERROR
                        <1>
00003644 E848020000     <1>       CALL   XLAT_OLD           ; ELSE TRANSLATE TO COMPATIBLE MODE
00003649 6689F0         <1>       MOV    AX,SI              ; RESTORE AL
0000364C B401           <1>       MOV    AH,BAD_CMD         ; SET BAD COMMAND ERROR
0000364E F9             <1>       STC
0000364F C3             <1>       RETn
                        <1>
                        <1> NON_DRV2:
                        <1>       ;XOR   AX,AX              ; CLEAR PARMS IF NO DRIVES OR CMOS BAD
00003650 31C0           <1>       xor    eax, eax
00003652 66894500       <1>       MOV    [eBP],AX           ; TRACKS, SECTORS/TRACK = 0
                        <1>       ;MOV   [BP+5],AH          ; HEAD = 0
00003656 886509         <1>       mov    [ebp+9], ah ; 06/02/2015
                        <1>       ;MOV   [BP+6],AX          ; OFFSET TO DISK_BASE = 0
00003659 89450C         <1>       mov    [ebp+12], eax
                        <1>       ;MOV   ES,AX              ; ES IS SEGMENT OF TABLE
0000365C EBD2           <1>       JMP    SHORT DP_OUT
```

```
                            <1>
                            <1> ;-----      DATA RATE IS EITHER 300 KBS OR 500 KBS, TRY 1.2 MB TABLE FIRST
                            <1>
                            <1> USE_EST2:
0000365E B002               <1>       MOV    AL,02               ; DRIVE TYPE 2 (1.2MB)
00003660 E887010000         <1>       CALL   DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
00003665 8A4B04             <1>         MOV     CL, [eBX+MD.SEC_TRK]    ; GET SECTOR/TRACK
00003668 8A6B0B             <1>         MOV     CH, [eBX+MD.MAX_TRK]    ; GET MAX. TRACK NUMBER
0000366B 80FC40             <1>       CMP    AH,RATE_300         ; RATE 300 ?
0000366E 74A1               <1>       JZ     short STO_CX        ; MUST BE 1.2MB DRIVE
00003670 EB92               <1>       JMP    SHORT PARM144       ; ELSE, IT IS 1.44MB DRIVE
                            <1>
                            <1> ;--------------------------------------------------------------------------
                            <1> ; DISK_TYPE (AH = 15H)
                            <1> ;     THIS ROUTINE RETURNS THE TYPE OF MEDIA INSTALLED.
                            <1> ;
                            <1> ;  ON ENTRY: DI = DRIVE #
                            <1> ;
                            <1> ;  ON EXIT:  AH = DRIVE TYPE, CY=0
                            <1> ;--------------------------------------------------------------------------
                            <1> DSK_TYPE:
00003672 E8E9010000         <1>       CALL   XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
00003677 8A87[95520100]     <1>       MOV    AL, [DSK_STATE+eDI] ; GET PRESENT STATE INFORMATION
0000367D 08C0               <1>       OR     AL,AL               ; CHECK FOR NO DRIVE
0000367F 7418               <1>       JZ     short NO_DRV
00003681 B401               <1>       MOV    AH,NOCHGLN          ; NO CHANGE LINE FOR 40 TRACK DRIVE
00003683 A801               <1>       TEST   AL,TRK_CAPA         ; IS THIS DRIVE AN 80 TRACK DRIVE?
00003685 7402               <1>       JZ     short DT_BACK             ; IF NO JUMP
00003687 B402               <1>       MOV    AH,CHGLN            ; CHANGE LINE FOR 80 TRACK DRIVE
                            <1> DT_BACK:
00003689 6650               <1>       PUSH   AX                  ; SAVE RETURN VALUE
0000368B E801020000         <1>       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
00003690 6658               <1>       POP    AX                  ; RESTORE RETURN VALUE
00003692 F8                 <1>       CLC                        ; NO ERROR
00003693 6689F3             <1>       MOV    BX,SI               ; GET SAVED AL TO BL
00003696 88D8               <1>       MOV    AL,BL               ; PUT BACK FOR RETURN
00003698 C3                 <1>       RETn
                            <1> NO_DRV:
00003699 30E4               <1>       XOR    AH,AH               ; NO DRIVE PRESENT OR UNKNOWN
0000369B EBEC               <1>       JMP    SHORT DT_BACK
                            <1>
                            <1> ;--------------------------------------------------------------------------
                            <1> ; DISK_CHANGE      (AH = 16H)
                            <1> ;     THIS ROUTINE RETURNS THE STATE OF THE DISK CHANGE LINE.
                            <1> ;
                            <1> ; ON ENTRY:  DI = DRIVE #
                            <1> ;
                            <1> ; ON EXIT:   AH = @DSKETTE_STATUS
                            <1> ;                 00 - DISK CHANGE LINE INACTIVE, CY = 0
                            <1> ;                 06 - DISK CHANGE LINE ACTIVE, CY = 1
                            <1> ;--------------------------------------------------------------------------
                            <1> DSK_CHANGE:
0000369D E8BE010000         <1>       CALL   XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
000036A2 8A87[95520100]     <1>       MOV    AL, [DSK_STATE+eDI] ; GET MEDIA STATE INFORMATION
000036A8 08C0               <1>       OR     AL,AL               ; DRIVE PRESENT ?
000036AA 7422               <1>       JZ     short DC_NON        ; JUMP IF NO DRIVE
000036AC A801               <1>       TEST   AL,TRK_CAPA         ; 80 TRACK DRIVE ?
000036AE 7407               <1>       JZ     short SETIT         ; IF SO , CHECK CHANGE LINE
                            <1> DC0:
000036B0 E88D0A0000         <1>         CALL    READ_DSKCHNG            ; GO CHECK STATE OF DISK CHANGE LINE
000036B5 7407               <1>       JZ     short FINIS         ; CHANGE LINE NOT ACTIVE
                            <1>
000036B7 C605[88520100]06   <1> SETIT:       MOV    byte [DSKETTE_STATUS], MEDIA_CHANGE ; INDICATE MEDIA REMOVED
                            <1>
000036BE E8CE010000         <1> FINIS:       CALL   XLAT_OLD           ; TRANSLATE STATE TO COMPATIBLE MODE
000036C3 E808070000         <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
000036C8 6689F3             <1>       MOV    BX,SI               ; GET SAVED AL TO BL
000036CB 88D8               <1>       MOV    AL,BL               ; PUT BACK FOR RETURN
000036CD C3                 <1>       RETn
                            <1> DC_NON:
000036CE 800D[88520100]80   <1>       OR     byte [DSKETTE_STATUS], TIME_OUT ; SET TIMEOUT, NO DRIVE
000036D5 EBE7               <1>       JMP    SHORT FINIS
                            <1>
                            <1> ;--------------------------------------------------------------------------
                            <1> ; FORMAT_SET(AH = 17H)
                            <1> ;     THIS ROUTINE IS USED TO ESTABLISH THE TYPE OF MEDIA TO BE USED
                            <1> ;     FOR THE FOLLOWING FORMAT OPERATION.
                            <1> ;
                            <1> ; ON ENTRY:  SI LOW = DASD TYPE FOR FORMAT
                            <1> ;            DI     = DRIVE #
                            <1> ;
                            <1> ; ON EXIT:   @DSKETTE_STATUS REFLECTS STATUS
                            <1> ;            AH = @DSKETTE_STATUS
                            <1> ;            CY = 1 IF ERROR
                            <1> ;--------------------------------------------------------------------------
                            <1> FORMAT_SET:
000036D7 E884010000         <1>       CALL   XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
000036DC 6656               <1>       PUSH   SI                  ; SAVE DASD TYPE
000036DE 6689F0             <1>       MOV    AX,SI               ; AH = ? , AL , DASD TYPE
000036E1 30E4               <1>       XOR    AH,AH               ; AH , 0 , AL , DASD TYPE
000036E3 6689C6             <1>       MOV    SI,AX               ; SI = DASD TYPE
000036E6 80A7[95520100]0F   <1>       AND    byte [DSK_STATE+eDI], ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR STATE
000036ED 664E               <1>       DEC    SI                  ; CHECK FOR 320/360K MEDIA & DRIVE
000036EF 7509               <1>       JNZ    short NOT_320       ; BYPASS IF NOT
000036F1 808F[95520100]90   <1>       OR     byte [DSK_STATE+eDI], MED_DET+RATE_250 ; SET TO 320/360
000036F8 EB48               <1>       JMP    SHORT S0
                            <1>
                            <1> NOT_320:
000036FA E8B6030000         <1>       CALL   MED_CHANGE          ; CHECK FOR TIME_OUT
000036FF 803D[88520100]80   <1>       CMP    byte [DSKETTE_STATUS], TIME_OUT
00003706 743A               <1>       JZ     short S0            ; IF TIME OUT TELL CALLER
                            <1> S3:
```

```
1060 00003708 664E                <1>       DEC    SI                  ; CHECK FOR 320/360K IN 1.2M DRIVE
1061 0000370A 7509                <1>       JNZ    short NOT_320_12    ; BYPASS IF NOT
1062 0000370C 808F[95520100]70    <1>       OR     byte [DSK_STATE+eDI], MED_DET+DBL_STEP+RATE_300 ; SET STATE
1063 00003713 EB2D                <1>       JMP    SHORT S0
1064                              <1>
1065                              <1> NOT_320_12:
1066 00003715 664E                <1>       DEC    SI                  ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
1067 00003717 7509                <1>       JNZ    short NOT_12        ; BYPASS IF NOT
1068 00003719 808F[95520100]10    <1>       OR     byte [DSK_STATE+eDI], MED_DET+RATE_500 ; SET STATE VARIABLE
1069 00003720 EB20                <1>       JMP    SHORT S0            ; RETURN TO CALLER
1070                              <1>
1071                              <1> NOT_12:
1072 00003722 664E                <1>       DEC    SI                  ; CHECK FOR SET DASD TYPE 04
1073 00003724 752B                <1>       JNZ    short FS_ERR        ; BAD COMMAND EXIT IF NOT VALID TYPE
1074                              <1>
1075 00003726 F687[95520100]04    <1>       TEST   byte [DSK_STATE+eDI], DRV_DET ; DRIVE DETERMINED ?
1076 0000372D 740B                <1>       JZ     short ASSUME        ; IF STILL NOT DETERMINED ASSUME
1077 0000372F B050                <1>       MOV    AL,MED_DET+RATE_300
1078 00003731 F687[95520100]02    <1>         TEST    byte [DSK_STATE+eDI], FMT_CAPA ; MULTIPLE FORMAT CAPABILITY ?
1079 00003738 7502                <1>       JNZ    short OR_IT_IN             ; IF 1.2 M THEN DATA RATE 300
1080                              <1>
1081                              <1> ASSUME:
1082 0000373A B090                <1>       MOV    AL,MED_DET+RATE_250 ; SET UP
1083                              <1>
1084                              <1> OR_IT_IN:
1085 0000373C 0887[95520100]      <1>       OR     [DSK_STATE+eDI], AL ; OR IN THE CORRECT STATE
1086                              <1> S0:
1087 00003742 E84A010000          <1>       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
1088 00003747 E884060000          <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
1089 0000374C 665B                <1>       POP    BX                  ; GET SAVED AL TO BL
1090 0000374E 88D8                <1>       MOV    AL,BL               ; PUT BACK FOR RETURN
1091 00003750 C3                  <1>       RETn
1092                              <1>
1093                              <1> FS_ERR:
1094 00003751 C605[88520100]01    <1>       MOV    byte [DSKETTE_STATUS], BAD_CMD ; UNKNOWN STATE,BAD COMMAND
1095 00003758 EBE8                <1>       JMP    SHORT S0
1096                              <1>
1097                              <1> ;--------------------------------------------------------------------------
1098                              <1> ; SET_MEDIA  (AH = 18H)
1099                              <1> ;     THIS ROUTINE SETS THE TYPE OF MEDIA AND DATA RATE
1100                              <1> ;     TO BE USED FOR THE FOLLOWING FORMAT OPERATION.
1101                              <1> ;
1102                              <1> ; ON ENTRY:
1103                              <1> ;     [BP]   = SECTOR PER TRACK
1104                              <1> ;     [BP+1] = TRACK #
1105                              <1> ;     DI     = DRIVE #
1106                              <1> ;
1107                              <1> ; ON EXIT:
1108                              <1> ;     @DSKETTE_STATUS REFLECTS STATUS
1109                              <1> ;     IF NO ERROR:
1110                              <1> ;           AH = 0
1111                              <1> ;           CY = 0
1112                              <1> ;           ES = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
1113                              <1> ;           DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE
1114                              <1> ;     IF ERROR:
1115                              <1> ;           AH = @DSKETTE_STATUS
1116                              <1> ;           CY = 1
1117                              <1> ;--------------------------------------------------------------------------
1118                              <1> SET_MEDIA:
1119 0000375A E801010000          <1>       CALL   XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
1120 0000375F F687[95520100]01    <1>         TEST    byte [DSK_STATE+eDI], TRK_CAPA ; CHECK FOR CHANGE LINE AVAILABLE
1121 00003766 7415                <1>       JZ     short SM_CMOS       ; JUMP IF 40 TRACK DRIVE
1122 00003768 E848030000          <1>       CALL   MED_CHANGE          ; RESET CHANGE LINE
1123 0000376D 803D[88520100]80    <1>       CMP    byte [DSKETTE_STATUS], TIME_OUT ; IF TIME OUT TELL CALLER
1124 00003774 746B                <1>       JE     short SM_RTN
1125 00003776 C605[88520100]00    <1>       MOV    byte [DSKETTE_STATUS], 0 ; CLEAR STATUS
1126                              <1> SM_CMOS:
1127 0000377D E819070000          <1>       CALL   CMOS_TYPE           ; RETURN DRIVE TYPE IN (AL)
1128                              <1>       ;;20/02/2015
1129                              <1>       ;;JC   short MD_NOT_FND    ; ERROR IN CMOS
1130                              <1>       ;;OR   AL,AL               ; TEST FOR NO DRIVE
1131 00003782 745D                <1>       JZ     short SM_RTN        ; RETURN IF SO
1132 00003784 E863000000          <1>       CALL   DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
1133 00003789 7231                <1>       JC     short MD_NOT_FND    ; TYPE NOT IN TABLE (BAD CMOS)
1134 0000378B 57                  <1>       PUSH   eDI                 ; SAVE REG.
1135 0000378C 31DB                <1>       XOR    eBX,eBX                  ; BX = INDEX TO DR. TYPE TABLE
1136 0000378E B906000000          <1>       MOV    eCX,DR_CNT          ; CX = LOOP COUNT
1137                              <1> DR_SEARCH:
1138 00003793 8AA3[705C0000]      <1>       MOV    AH, [DR_TYPE+eBX]   ; GET DRIVE TYPE
1139 00003799 80E47F              <1>       AND    AH,BIT7OFF          ; MASK OUT MSB
1140 0000379C 38E0                <1>       CMP    AL,AH               ; DRIVE TYPE MATCH ?
1141 0000379E 7516                <1>       JNE    short NXT_MD        ; NO, CHECK NEXT DRIVE TYPE
1142                              <1> DR_FND:
1143 000037A0 8BBB[715C0000]      <1>       MOV    eDI, [DR_TYPE+eBX+1]      ; DI = MEDIA/DRIVE PARAM TABLE
1144                              <1> MD_SEARCH:
1145 000037A6 8A6704              <1>         MOV     AH, [eDI+MD.SEC_TRK]    ; GET SECTOR/TRACK
1146 000037A9 386500              <1>       CMP    [eBP],AH            ; MATCH?
1147 000037AC 7508                <1>       JNE    short NXT_MD        ; NO, CHECK NEXT MEDIA
1148 000037AE 8A670B              <1>         MOV     AH, [eDI+MD.MAX_TRK]    ; GET MAX. TRACK #
1149 000037B1 386501              <1>       CMP    [eBP+1],AH          ; MATCH?
1150 000037B4 740F                <1>       JE     short MD_FND        ; YES, GO GET RATE
1151                              <1> NXT_MD:
1152                              <1>       ;ADD   BX,3                ; CHECK NEXT DRIVE TYPE
1153 000037B6 83C305              <1>         add  ebx, 5 ; 18/02/2015
1154 000037B9 E2D8                <1>       LOOP    DR_SEARCH
1155 000037BB 5F                  <1>       POP    eDI                 ; RESTORE REG.
1156                              <1> MD_NOT_FND:
1157 000037BC C605[88520100]0C    <1>       MOV    byte [DSKETTE_STATUS], MED_NOT_FND ; ERROR, MEDIA TYPE NOT FOUND
1158 000037C3 EB1C                <1>       JMP    SHORT SM_RTN        ; RETURN
1159                              <1> MD_FND:
1160 000037C5 8A470C              <1>         MOV     AL, [eDI+MD.RATE]       ; GET RATE
1161 000037C8 3C40                <1>       CMP    AL,RATE_300         ; DOUBLE STEP REQUIRED FOR RATE 300
```

```
1162 000037CA 7502                <1>       JNE    short MD_SET
1163 000037CC 0C20                <1>       OR     AL,DBL_STEP
1164                              <1> MD_SET:
1165                              <1>       ;MOV   [BP+6],DI           ; SAVE TABLE POINTER IN STACK
1166 000037CE 897D0C              <1>       mov    [ebp+12], edi ; 18/02/2015
1167 000037D1 0C10                <1>       OR     AL,MED_DET          ; SET MEDIA ESTABLISHED
1168 000037D3 5F                  <1>       POP    eDI
1169 000037D4 80A7[95520100]0F    <1>       AND    byte [DSK_STATE+eDI], ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR STATE
1170 000037DB 0887[95520100]      <1>       OR     [DSK_STATE+eDI], AL
1171                              <1>       ;MOV   AX, CS              ; SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
1172                              <1>       ;MOV   ES, AX              ; ES IS SEGMENT OF TABLE
1173                              <1> SM_RTN:
1174 000037E1 E8AB000000          <1>       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
1175 000037E6 E8E5050000          <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
1176 000037EB C3                  <1>       RETn
1177                              <1>
1178                              <1> ;--------------------------------------------------------------
1179                              <1> ; DR_TYPE_CHECK                                          :
1180                              <1> ;     CHECK IF THE GIVEN DRIVE TYPE IN REGISTER (AL)           :
1181                              <1> ;     IS SUPPORTED IN BIOS DRIVE TYPE TABLE            :
1182                              <1> ; ON ENTRY:                                      :
1183                              <1> ;     AL = DRIVE TYPE                                  :
1184                              <1> ; ON EXIT:                                       :
1185                              <1> ;     CS = SEGMENT MEDIA/DRIVE PARAMETER TABLE (CODE)          :
1186                              <1> ;     CY = 0       DRIVE TYPE SUPPORTED                        :
1187                              <1> ;          BX = OFFSET TO MEDIA/DRIVE PARAMETER TABLE          :
1188                              <1> ;     CY = 1 DRIVE TYPE NOT SUPPORTED              :
1189                              <1> ; REGISTERS ALTERED: eBX                                 :
1190                              <1> ;--------------------------------------------------------------
1191                              <1> DR_TYPE_CHECK:
1192 000037EC 6650                <1>       PUSH   AX
1193 000037EE 51                  <1>       PUSH   eCX
1194 000037EF 31DB                <1>       XOR    eBX,eBX                  ; BX = INDEX TO DR_TYPE TABLE
1195 000037F1 B906000000          <1>       MOV    eCX,DR_CNT          ; CX = LOOP COUNT
1196                              <1> TYPE_CHK:
1197 000037F6 8AA3[705C0000]      <1>       MOV    AH,[DR_TYPE+eBX]    ; GET DRIVE TYPE
1198 000037FC 38E0                <1>       CMP    AL,AH               ; DRIVE TYPE MATCH?
1199 000037FE 740D                <1>       JE     short DR_TYPE_VALID ; YES, RETURN WITH CARRY RESET
1200                              <1>       ;ADD   BX,3                ; CHECK NEXT DRIVE TYPE
1201 00003800 83C305              <1>         add  ebx, 5 ; 16/02/2015 (32 bit address modification)
1202 00003803 E2F1                <1>       LOOP    TYPE_CHK
1203                              <1>       ;
1204 00003805 BB[CF5C0000]        <1>       mov    ebx, MD_TBL6        ; 1.44MB fd parameter table
1205                              <1>                                  ; Default for GET_PARM (11/12/2014)
1206                              <1>       ;
1207 0000380A F9                  <1>       STC                        ; DRIVE TYPE NOT FOUND IN TABLE
1208 0000380B EB06                <1>       JMP    SHORT TYPE_RTN
1209                              <1> DR_TYPE_VALID:
1210 0000380D 8B9B[715C0000]      <1>       MOV    eBX,[DR_TYPE+eBX+1]       ; BX = MEDIA TABLE
1211                              <1> TYPE_RTN:
1212 00003813 59                  <1>       POP    eCX
1213 00003814 6658                <1>       POP    AX
1214 00003816 C3                  <1>       RETn
1215                              <1>
1216                              <1> ;--------------------------------------------------------------
1217                              <1> ; SEND_SPEC                                              :
1218                              <1> ;     SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM     :
1219                              <1> ;     THE DRIVE PARAMETER TABLE POINTED BY @DISK_POINTER   :
1220                              <1> ; ON ENTRY:  @DISK_POINTER = DRIVE PARAMETER TABLE         :
1221                              <1> ; ON EXIT:   NONE                                        :
1222                              <1> ; REGISTERS ALTERED: CX, DX                              :
1223                              <1> ;--------------------------------------------------------------
1224                              <1> SEND_SPEC:
1225 00003817 50                  <1>       PUSH   eAX                 ; SAVE AX
1226 00003818 B8[3E380000]        <1>       MOV    eAX, SPECBAC        ; LOAD ERROR ADDRESS
1227 0000381D 50                  <1>       PUSH   eAX                 ; PUSH NEC_OUT ERROR RETURN
1228 0000381E B403                <1>       MOV    AH,03H              ; SPECIFY COMMAND
1229 00003820 E885070000          <1>       CALL   NEC_OUTPUT          ; OUTPUT THE COMMAND
1230 00003825 28D2                <1>       SUB    DL,DL               ; FIRST SPECIFY BYTE
1231 00003827 E878060000          <1>       CALL   GET_PARM            ; GET PARAMETER TO AH
1232 0000382C E879070000          <1>       CALL   NEC_OUTPUT          ; OUTPUT THE COMMAND
1233 00003831 B201                <1>       MOV    DL,1                ; SECOND SPECIFY BYTE
1234 00003833 E86C060000          <1>       CALL   GET_PARM            ; GET PARAMETER TO AH
1235 00003838 E86D070000          <1>       CALL   NEC_OUTPUT          ; OUTPUT THE COMMAND
1236 0000383D 58                  <1>       POP    eAX                 ; POP ERROR RETURN
1237                              <1> SPECBAC:
1238 0000383E 58                  <1>       POP    eAX                 ; RESTORE ORIGINAL AX VALUE
1239 0000383F C3                  <1>       RETn
1240                              <1>
1241                              <1> ;--------------------------------------------------------------
1242                              <1> ; SEND_SPEC_MD                                           :
1243                              <1> ;     SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM     :
1244                              <1> ;     THE MEDIA/DRIVE PARAMETER TABLE POINTED BY (CS:BX)   :
1245                              <1> ; ON ENTRY:  CS:BX = MEDIA/DRIVE PARAMETER TABLE         :
1246                              <1> ; ON EXIT:   NONE                                :
1247                              <1> ; REGISTERS ALTERED: AX                                  :
1248                              <1> ;--------------------------------------------------------------
1249                              <1> SEND_SPEC_MD:
1250 00003840 50                  <1>       PUSH   eAX                 ; SAVE RATE DATA
1251 00003841 B8[5E380000]        <1>       MOV    eAX, SPEC_ESBAC           ; LOAD ERROR ADDRESS
1252 00003846 50                  <1>       PUSH   eAX                 ; PUSH NEC_OUT ERROR RETURN
1253 00003847 B403                <1>       MOV    AH,03H              ; SPECIFY COMMAND
1254 00003849 E85C070000          <1>       CALL   NEC_OUTPUT          ; OUTPUT THE COMMAND
1255 0000384E 8A23                <1>         MOV     AH, [eBX+MD.SPEC1]      ; GET 1ST SPECIFY BYTE
1256 00003850 E855070000          <1>       CALL   NEC_OUTPUT          ; OUTPUT THE COMMAND
1257 00003855 8A6301              <1>         MOV     AH, [eBX+MD.SPEC2]      ; GET SECOND SPECIFY BYTE
1258 00003858 E84D070000          <1>       CALL   NEC_OUTPUT          ; OUTPUT THE COMMAND
1259 0000385D 58                  <1>       POP    eAX                 ; POP ERROR RETURN
1260                              <1> SPEC_ESBAC:
1261 0000385E 58                  <1>       POP    eAX                 ; RESTORE ORIGINAL AX VALUE
1262 0000385F C3                  <1>       RETn
1263                              <1>
```

```
1264                            <1> ;--------------------------------------------------------------------------
1265                            <1> ; XLAT_NEW
1266                            <1> ;     TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE
1267                            <1> ;     MODE TO NEW ARCHITECTURE.
1268                            <1> ;
1269                            <1> ; ON ENTRY:  DI = DRIVE #
1270                            <1> ;--------------------------------------------------------------------------
1271                            <1> XLAT_NEW:
1272 00003860 83FF01           <1>       CMP    eDI,1                    ; VALID DRIVE
1273 00003863 7725             <1>       JA     short XN_OUT             ; IF INVALID BACK
1274 00003865 80BF[95520100]00 <1>       CMP    byte [DSK_STATE+eDI], 0          ; NO DRIVE ?
1275 0000386C 741D             <1>       JZ     short DO_DET             ; IF NO DRIVE ATTEMPT DETERMINE
1276 0000386E 6689F9           <1>       MOV    CX,DI                    ; CX = DRIVE NUMBER
1277 00003871 C0E102           <1>       SHL    CL,2                     ; CL = SHIFT COUNT, A=0, B=4
1278 00003874 A0[94520100]     <1>       MOV    AL, [HF_CNTRL]                ; DRIVE INFORMATION
1279 00003879 D2C8             <1>       ROR    AL,CL                    ; TO LOW NIBBLE
1280 0000387B 2407             <1>       AND    AL,DRV_DET+FMT_CAPA+TRK_CAPA     ; KEEP DRIVE BITS
1281 0000387D 80A7[95520100]F8 <1>         AND     byte [DSK_STATE+eDI], ~(DRV_DET+FMT_CAPA+TRK_CAPA)
1282 00003884 0887[95520100]   <1>       OR     [DSK_STATE+eDI], AL      ; UPDATE DRIVE STATE
1283                            <1> XN_OUT:
1284 0000388A C3               <1>       RETn
1285                            <1> DO_DET:
1286 0000388B E8BF080000       <1>       CALL   DRIVE_DET                ; TRY TO DETERMINE
1287 00003890 C3               <1>       RETn
1288                            <1>
1289                            <1> ;--------------------------------------------------------------------------
1290                            <1> ; XLAT_OLD
1291                            <1> ;     TRANSLATES DISKETTE STATE LOCATIONS FROM NEW
1292                            <1> ;     ARCHITECTURE TO COMPATIBLE MODE.
1293                            <1> ;
1294                            <1> ; ON ENTRY:  DI = DRIVE
1295                            <1> ;--------------------------------------------------------------------------
1296                            <1> XLAT_OLD:
1297 00003891 83FF01           <1>       CMP    eDI,1              ; VALID DRIVE ?
1298                            <1>         ;JA     short XO_OUT            ; IF INVALID BACK
1299 00003894 0F8786000000     <1>         ja      XO_OUT
1300 0000389A 80BF[95520100]00 <1>         CMP  byte [DSK_STATE+eDI],0    ; NO DRIVE ?
1301 000038A1 747D             <1>       JZ     short XO_OUT        ; IF NO DRIVE TRANSLATE DONE
1302                            <1>
1303                            <1> ;-----      TEST FOR SAVED DRIVE INFORMATION ALREADY SET
1304                            <1>
1305 000038A3 6689F9           <1>       MOV    CX,DI               ; CX = DRIVE NUMBER
1306 000038A6 C0E102           <1>       SHL    CL,2                ; CL = SHIFT COUNT, A=0, B=4
1307 000038A9 B402             <1>       MOV    AH,FMT_CAPA         ; LOAD MULTIPLE DATA RATE BIT MASK
1308 000038AB D2CC             <1>       ROR    AH,CL               ; ROTATE BY MASK
1309 000038AD 8425[94520100]   <1>       TEST   [HF_CNTRL], AH            ; MULTIPLE-DATA RATE DETERMINED ?
1310 000038B3 751C             <1>       JNZ    short SAVE_SET            ; IF SO, NO NEED TO RE-SAVE
1311                            <1>
1312                            <1> ;-----      ERASE DRIVE BITS IN @HF_CNTRL FOR THIS DRIVE
1313                            <1>
1314 000038B5 B407             <1>       MOV    AH,DRV_DET+FMT_CAPA+TRK_CAPA ; MASK TO KEEP
1315 000038B7 D2CC             <1>       ROR    AH,CL               ; FIX MASK TO KEEP
1316 000038B9 F6D4             <1>       NOT    AH                  ; TRANSLATE MASK
1317 000038BB 2025[94520100]   <1>       AND    [HF_CNTRL], AH            ; KEEP BITS FROM OTHER DRIVE INTACT
1318                            <1>
1319                            <1> ;-----      ACCESS CURRENT DRIVE BITS AND STORE IN @HF_CNTRL
1320                            <1>
1321 000038C1 8A87[95520100]   <1>       MOV    AL, [DSK_STATE+eDI] ; ACCESS STATE
1322 000038C7 2407             <1>       AND    AL,DRV_DET+FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
1323 000038C9 D2C8             <1>       ROR    AL,CL               ; FIX FOR THIS DRIVE
1324 000038CB 0805[94520100]   <1>       OR     [HF_CNTRL], AL            ; UPDATE SAVED DRIVE STATE
1325                            <1>
1326                            <1> ;-----      TRANSLATE TO COMPATIBILITY MODE
1327                            <1>
1328                            <1> SAVE_SET:
1329 000038D1 8AA7[95520100]   <1>       MOV    AH, [DSK_STATE+eDI] ; ACCESS STATE
1330 000038D7 88E7             <1>       MOV    BH,AH               ; TO BH FOR LATER
1331 000038D9 80E4C0           <1>       AND    AH,RATE_MSK         ; KEEP ONLY RATE
1332 000038DC 80FC00           <1>       CMP    AH,RATE_500         ; RATE 500 ?
1333 000038DF 7410             <1>       JZ     short CHK_144       ; YES 1.2/1.2 OR 1.44/1.44
1334 000038E1 B001             <1>       MOV    AL,M3D1U            ; AL = 360 IN 1.2 UNESTABLISHED
1335 000038E3 80FC40           <1>       CMP    AH,RATE_300         ; RATE 300 ?
1336 000038E6 7518             <1>       JNZ    short CHK_250       ; NO, 360/360, 720/720 OR 720/1.44
1337 000038E8 F6C720           <1>       TEST   BH,DBL_STEP         ; CHECK FOR DOUBLE STEP
1338 000038EB 751F             <1>       JNZ    short TST_DET       ; MUST BE 360 IN 1.2
1339                            <1> UNKNO:
1340 000038ED B007             <1>       MOV    AL,MED_UNK          ; NONE OF THE ABOVE
1341 000038EF EB22             <1>       JMP    SHORT AL_SET        ; PROCESS COMPLETE
1342                            <1> CHK_144:
1343 000038F1 E8A5050000       <1>       CALL   CMOS_TYPE           ; RETURN DRIVE TYPE IN (AL)
1344                            <1>       ;;20/02/2015
1345                            <1>       ;;JC   short UNKNO         ; ERROR, SET 'NONE OF ABOVE'
1346 000038F6 74F5             <1>       jz     short UNKNO ;; 20/02/2015
1347 000038F8 3C02             <1>       CMP    AL,2                ; 1.2MB DRIVE ?
1348 000038FA 75F1             <1>       JNE    short UNKNO         ; NO, GO SET 'NONE OF ABOVE'
1349 000038FC B002             <1>       MOV    AL,M1D1U            ; AL = 1.2 IN 1.2 UNESTABLISHED
1350 000038FE EB0C             <1>       JMP    SHORT TST_DET
1351                            <1> CHK_250:
1352 00003900 B000             <1>       MOV    AL,M3D3U            ; AL = 360 IN 360 UNESTABLISHED
1353 00003902 80FC80           <1>       CMP    AH,RATE_250         ; RATE 250 ?
1354 00003905 75E6             <1>       JNZ    short UNKNO         ; IF SO FALL IHRU
1355 00003907 F6C701           <1>       TEST   BH,TRK_CAPA         ; 80 TRACK CAPABILITY ?
1356 0000390A 75E1             <1>       JNZ    short UNKNO         ; IF SO JUMP, FALL THRU TEST DET
1357                            <1> TST_DET:
1358 0000390C F6C710           <1>       TEST   BH,MED_DET          ; DETERMINED ?
1359 0000390F 7402             <1>       JZ     short AL_SET        ; IF NOT THEN SET
1360 00003911 0403             <1>       ADD    AL,3                ; MAKE DETERMINED/ESTABLISHED
1361                            <1> AL_SET:
1362 00003913 80A7[95520100]F8 <1>       AND    byte [DSK_STATE+eDI], ~(DRV_DET+FMT_CAPA+TRK_CAPA) ; CLEAR DRIVE
1363 0000391A 0887[95520100]   <1>       OR     [DSK_STATE+eDI], AL ; REPLACE WITH COMPATIBLE MODE
1364                            <1> XO_OUT:
1365 00003920 C3               <1>       RETn
```

```
1366                            <1>
1367                            <1> ;--------------------------------------------------------------------------------
1368                            <1> ; RD_WR_VF
1369                            <1> ;     COMMON READ, WRITE AND VERIFY:
1370                            <1> ;     MAIN LOOP FOR STATE RETRIES.
1371                            <1> ;
1372                            <1> ; ON ENTRY:  AH = READ/WRITE/VERIFY NEC PARAMETER
1373                            <1> ;            AL = READ/WRITE/VERIFY DMA PARAMETER
1374                            <1> ;
1375                            <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1376                            <1> ;--------------------------------------------------------------------------------
1377                            <1> RD_WR_VF:
1378 00003921 6650             <1>       PUSH   AX                  ; SAVE DMA, NEC PARAMETERS
1379 00003923 E838FFFFFF       <1>       CALL   XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
1380 00003928 E8F3000000       <1>       CALL   SETUP_STATE         ; INITIALIZE START AND END RATE
1381 0000392D 6658             <1>       POP    AX                  ; RESTORE READ/WRITE/VERIFY
1382                            <1> DO_AGAIN:
1383 0000392F 6650             <1>       PUSH   AX                  ; SAVE READ/WRITE/VERIFY PARAMETER
1384 00003931 E87F010000       <1>       CALL   MED_CHANGE          ; MEDIA CHANGE AND RESET IF CHANGED
1385 00003936 6658             <1>       POP    AX                  ; RESTORE READ/WRITE/VERIFY
1386 00003938 0F82C9000000     <1>         JC      RWV_END                ; MEDIA CHANGE ERROR OR TIME-OUT
1387                            <1> RWV:
1388 0000393E 6650             <1>       PUSH   AX                  ; SAVE READ/WRITE/VERIFY PARAMETER
1389 00003940 8AB7[95520100]   <1>       MOV    DH, [DSK_STATE+eDI] ; GET RATE STATE OF THIS DRIVE
1390 00003946 80E6C0           <1>       AND    DH,RATE_MSK         ; KEEP ONLY RATE
1391 00003949 E84D050000       <1>       CALL   CMOS_TYPE           ; RETURN DRIVE TYPE IN AL (AL)
1392                            <1>       ;;20/02/2015
1393                            <1>       ;;JC   short RWV_ASSUME    ; ERROR IN CMOS
1394 0000394E 7451             <1>       jz     short RWV_ASSUME ; 20/02/2015
1395 00003950 3C01             <1>       CMP    AL,1                ; 40 TRACK DRIVE?
1396 00003952 750D             <1>       JNE    short RWV_1         ; NO, BYPASS CMOS VALIDITY CHECK
1397 00003954 F687[95520100]01 <1>       TEST   byte [DSK_STATE+eDI], TRK_CAPA ; CHECK FOR 40 TRACK DRIVE
1398 0000395B 7413             <1>       JZ     short RWV_2         ; YES, CMOS IS CORRECT
1399 0000395D B002             <1>       MOV    AL,2                ; CHANGE TO 1.2M
1400 0000395F EB0F             <1>       JMP    SHORT RWV_2
1401                            <1> RWV_1:
1402 00003961 720D             <1>       JB     short RWV_2         ; NO DRIVE SPECIFIED, CONTINUE
1403 00003963 F687[95520100]01 <1>       TEST    byte [DSK_STATE+eDI], TRK_CAPA ; IS IT REALLY 40 TRACK?
1404 0000396A 7504             <1>       JNZ    short RWV_2         ; NO, 80 TRACK
1405 0000396C B001             <1>       MOV    AL,1                ; IT IS 40 TRACK, FIX CMOS VALUE
1406 0000396E EB04             <1>       jmp    short rwv_3
1407                            <1> RWV_2:
1408 00003970 08C0             <1>       OR     AL,AL               ; TEST FOR NO DRIVE
1409 00003972 742D             <1>       JZ     short RWV_ASSUME    ; ASSUME TYPE, USE MAX TRACK
1410                            <1> rwv_3:
1411 00003974 E873FEFFFF       <1>       CALL   DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL.
1412 00003979 7226             <1>       JC     short RWV_ASSUME    ; TYPE NOT IN TABLE (BAD CMOS)
1413                            <1>
1414                            <1> ;-----       SEARCH FOR MEDIA/DRIVE PARAMETER TABLE
1415                            <1>
1416 0000397B 57               <1>       PUSH   eDI                 ; SAVE DRIVE #
1417 0000397C 31DB             <1>       XOR    eBX,eBX                  ; BX = INDEX TO DR_TYPE TABLE
1418 0000397E B906000000       <1>       MOV    eCX,DR_CNT          ; CX = LOOP COUNT
1419                            <1> RWV_DR_SEARCH:
1420 00003983 8AA3[705C0000]   <1>       MOV    AH, [DR_TYPE+eBX]   ; GET DRIVE TYPE
1421 00003989 80E47F           <1>       AND    AH,BIT7OFF          ; MASK OUT MSB
1422 0000398C 38E0             <1>       CMP    AL,AH               ; DRIVE TYPE MATCH?
1423 0000398E 750B             <1>       JNE    short RWV_NXT_MD    ; NO, CHECK NEXT DRIVE TYPE
1424                            <1> RWV_DR_FND:
1425 00003990 8BBB[715C0000]   <1>       MOV    eDI, [DR_TYPE+eBX+1]      ; DI = MEDIA/DRIVE PARAMETER TABLE
1426                            <1> RWV_MD_SEARH:
1427 00003996 3A770C           <1>         CMP     DH, [eDI+MD.RATE]       ; MATCH?
1428 00003999 741B             <1>       JE     short RWV_MD_FND    ; YES, GO GET 1ST SPECIFY BYTE
1429                            <1> RWV_NXT_MD:
1430                            <1>       ;ADD   BX,3                ; CHECK NEXT DRIVE TYPE
1431 0000399B 83C305           <1>       add    eBX, 5
1432 0000399E E2E3             <1>       LOOP   RWV_DR_SEARCH
1433 000039A0 5F               <1>       POP    eDI                 ; RESTORE DRIVE #
1434                            <1>
1435                            <1> ;-----       ASSUME PRIMARY DRIVE IS INSTALLED AS SHIPPED
1436                            <1>
1437                            <1> RWV_ASSUME:
1438 000039A1 BB[8E5C0000]     <1>       MOV    eBX, MD_TBL1        ; POINT TO 40 TRACK 250 KBS
1439 000039A6 F687[95520100]01 <1>       TEST   byte [DSK_STATE+eDI], TRK_CAPA ; TEST FOR 80 TRACK
1440 000039AD 740A             <1>       JZ     short RWV_MD_FND1   ; MUST BE 40 TRACK
1441 000039AF BB[A85C0000]     <1>       MOV    eBX, MD_TBL3        ; POINT TO 80 TRACK 500 KBS
1442 000039B4 EB03             <1>       JMP    short RWV_MD_FND1   ; GO SPECIFY PARAMTERS
1443                            <1>
1444                            <1> ;-----       CS:BX POINTS TO MEDIA/DRIVE PARAMETER TABLE
1445                            <1>
1446                            <1> RWV_MD_FND:
1447 000039B6 89FB             <1>       MOV    eBX,eDI                  ; BX = MEDIA/DRIVE PARAMETER TABLE
1448 000039B8 5F               <1>       POP    eDI                 ; RESTORE DRIVE #
1449                            <1>
1450                            <1> ;-----       SEND THE SPECIFY COMMAND TO THE CONTROLLER
1451                            <1>
1452                            <1> RWV_MD_FND1:
1453 000039B9 E882FEFFFF       <1>       CALL   SEND_SPEC_MD
1454 000039BE E864010000       <1>       CALL   CHK_LASTRATE        ; ZF=1 ATTEMP RATE IS SAME AS LAST RATE
1455 000039C3 7405             <1>       JZ     short RWV_DBL       ; YES,SKIP SEND RATE COMMAND
1456 000039C5 E83B010000       <1>       CALL   SEND_RATE           ; SEND DATA RATE TO NEC
1457                            <1> RWV_DBL:
1458 000039CA 53               <1>       PUSH   eBX                 ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
1459 000039CB E822040000       <1>       CALL   SETUP_DBL           ; CHECK FOR DOUBLE STEP
1460 000039D0 5B               <1>       POP    eBX                 ; RESTORE ADDRESS
1461 000039D1 7226             <1>       JC     short CHK_RET       ; ERROR FROM READ ID, POSSIBLE RETRY
1462 000039D3 6658             <1>       POP    AX                  ; RESTORE NEC, DMA COMMAND
1463 000039D5 6650             <1>       PUSH   AX                  ; SAVE NEC COMMAND
1464 000039D7 53               <1>       PUSH   eBX                 ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
1465 000039D8 E861010000       <1>       CALL   DMA_SETUP           ; SET UP THE DMA
1466 000039DD 5B               <1>       POP    eBX
1467 000039DE 6658             <1>       POP    AX                  ; RESTORE NEC COMMAND
```

```
1468 000039E0 722F              <1>       JC     short RWV_BAC       ; CHECK FOR DMA BOUNDARY ERROR
1469 000039E2 6650              <1>       PUSH   AX                  ; SAVE NEC COMMAND
1470 000039E4 53                <1>       PUSH   eBX                 ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
1471 000039E5 E83C020000        <1>       CALL   NEC_INIT            ; INITIALIZE NEC
1472 000039EA 5B                <1>       POP    eBX                 ; RESTORE ADDRESS
1473 000039EB 720C              <1>       JC     short CHK_RET       ; ERROR - EXIT
1474 000039ED E866020000        <1>       CALL   RWV_COM                   ; OP CODE COMMON TO READ/WRITE/VERIFY
1475 000039F2 7205              <1>       JC     short CHK_RET       ; ERROR - EXIT
1476 000039F4 E8AB020000        <1>       CALL   NEC_TERM            ; TERMINATE, GET STATUS, ETC.
1477                            <1> CHK_RET:
1478 000039F9 E84A030000        <1>       CALL   RETRY               ; CHECK FOR, SETUP RETRY
1479 000039FE 6658              <1>       POP    AX                  ; RESTORE READ/WRITE/VERIFY PARAMETER
1480 00003A00 7305              <1>       JNC    short RWV_END       ; CY = 0 NO RETRY
1481 00003A02 E928FFFFFF        <1>         JMP     DO_AGAIN                ; CY = 1 MEANS RETRY
1482                            <1> RWV_END:
1483 00003A07 E8F4020000        <1>       CALL   DSTATE              ; ESTABLISH STATE IF SUCCESSFUL
1484 00003A0C E887030000        <1>       CALL   NUM_TRANS           ; AL = NUMBER TRANSFERRED
1485                            <1> RWV_BAC:                         ; BAD DMA ERROR ENTRY
1486 00003A11 6650              <1>       PUSH   AX                  ; SAVE NUMBER TRANSFERRED
1487 00003A13 E879FEFFFF        <1>       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
1488 00003A18 6658              <1>       POP    AX                  ; RESTORE NUMBER TRANSFERRED
1489 00003A1A E8B1030000        <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
1490 00003A1F C3                <1>       RETn
1491                            <1>
1492                            <1> ;--------------------------------------------------------------------------
1493                            <1> ; SETUP_STATE:     INITIALIZES START AND END RATES.
1494                            <1> ;--------------------------------------------------------------------------
1495                            <1> SETUP_STATE:
1496 00003A20 F687[95520100]10  <1>       TEST   byte [DSK_STATE+eDI], MED_DET ; MEDIA DETERMINED ?
1497 00003A27 7537              <1>       JNZ    short J1C           ; NO STATES IF DETERMINED
1498 00003A29 66B84000          <1>         MOV     AX,(RATE_500*256)+RATE_300  ; AH = START RATE, AL = END RATE
1499 00003A2D F687[95520100]04  <1>       TEST   byte [DSK_STATE+eDI],DRV_DET ; DRIVE ?
1500 00003A34 740D              <1>       JZ     short AX_SET        ; DO NOT KNOW DRIVE
1501 00003A36 F687[95520100]02  <1>       TEST   byte [DSK_STATE+eDI], FMT_CAPA ; MULTI-RATE?
1502 00003A3D 7504              <1>       JNZ    short AX_SET        ; JUMP IF YES
1503 00003A3F 66B88080          <1>         MOV     AX,RATE_250*257         ; START A END RATE 250 FOR 360 DRIVE
1504                            <1> AX_SET:
1505 00003A43 80A7[95520100]1F  <1>       AND    byte [DSK_STATE+eDI], ~(RATE_MSK+DBL_STEP) ; TURN OFF THE RATE
1506 00003A4A 08A7[95520100]    <1>       OR     [DSK_STATE+eDI], AH ; RATE FIRST TO TRY
1507 00003A50 8025[90520100]F3  <1>       AND    byte [LASTRATE], ~STRT_MSK ; ERASE LAST TO TRY RATE BITS
1508 00003A57 C0C804            <1>       ROR    AL,4                ; TO OPERATION LAST RATE LOCATION
1509 00003A5A 0805[90520100]    <1>       OR     [LASTRATE], AL             ; LAST RATE
1510                            <1> J1C:
1511 00003A60 C3                <1>       RETn
1512                            <1>
1513                            <1> ;--------------------------------------------------------------------------
1514                            <1> ;  FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.
1515                            <1> ;--------------------------------------------------------------------------
1516                            <1> FMT_INIT:
1517 00003A61 F687[95520100]10  <1>       TEST   byte [DSK_STATE+eDI], MED_DET ; IS MEDIA ESTABLISHED
1518 00003A68 7546              <1>       JNZ    short F1_OUT        ; IF SO RETURN
1519 00003A6A E82C040000        <1>       CALL   CMOS_TYPE           ; RETURN DRIVE TYPE IN AL
1520                            <1>       ;; 20/02/2015
1521                            <1>       ;;JC   short CL_DRV        ; ERROR IN CMOS ASSUME NO DRIVE
1522 00003A6F 7440              <1>       jz     short CL_DRV ;; 20/02/2015
1523 00003A71 FEC8              <1>       DEC    AL                  ; MAKE ZERO ORIGIN
1524                            <1>       ;;JS   short CL_DRV        ; NO DRIVE IF AL 0
1525 00003A73 8AA7[95520100]    <1>       MOV    AH, [DSK_STATE+eDI] ; AH = CURRENT STATE
1526 00003A79 80E40F            <1>       AND    AH, ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR
1527 00003A7C 08C0              <1>       OR     AL,AL               ; CHECK FOR 360
1528 00003A7E 7505              <1>       JNZ    short N_360         ; IF 360 WILL BE 0
1529 00003A80 80CC90            <1>       OR     AH,MED_DET+RATE_250 ; ESTABLISH MEDIA
1530 00003A83 EB25              <1>       JMP    SHORT SKP_STATE            ; SKIP OTHER STATE PROCESSING
1531                            <1> N_360:
1532 00003A85 FEC8              <1>       DEC    AL                  ; 1.2 M DRIVE
1533 00003A87 7505              <1>       JNZ    short N_12          ; JUMP IF NOT
1534                            <1> F1_RATE:
1535 00003A89 80CC10            <1>       OR     AH,MED_DET+RATE_500 ; SET FORMAT RATE
1536 00003A8C EB1C              <1>       JMP    SHORT SKP_STATE            ; SKIP OTHER STATE PROCESSING
1537                            <1> N_12:
1538 00003A8E FEC8              <1>       DEC    AL                  ; CHECK FOR TYPE 3
1539 00003A90 750F              <1>       JNZ    short N_720         ; JUMP IF NOT
1540 00003A92 F6C404            <1>       TEST   AH,DRV_DET          ; IS DRIVE DETERMINED
1541 00003A95 7410              <1>       JZ     short ISNT_12       ; TREAT AS NON 1.2 DRIVE
1542 00003A97 F6C402            <1>       TEST   AH,FMT_CAPA         ; IS 1.2M
1543 00003A9A 740B              <1>       JZ     short ISNT_12       ; JUMP IF NOT
1544 00003A9C 80CC50            <1>       OR     AH,MED_DET+RATE_300 ; RATE 300
1545 00003A9F EB09              <1>       JMP    SHORT SKP_STATE            ; CONTINUE
1546                            <1> N_720:
1547 00003AA1 FEC8              <1>       DEC    AL                  ; CHECK FOR TYPE 4
1548 00003AA3 750C              <1>       JNZ    short CL_DRV        ; NO DRIVE, CMOS BAD
1549 00003AA5 EBE2              <1>       JMP    SHORT F1_RATE
1550                            <1> ISNT_12:
1551 00003AA7 80CC90            <1>       OR     AH,MED_DET+RATE_250 ; MUST BE RATE 250
1552                            <1>
1553                            <1> SKP_STATE:
1554 00003AAA 88A7[95520100]    <1>       MOV    [DSK_STATE+eDI], AH ; STORE AWAY
1555                            <1> F1_OUT:
1556 00003AB0 C3                <1>       RETn
1557                            <1> CL_DRV:
1558 00003AB1 30E4              <1>       XOR    AH,AH               ; CLEAR STATE
1559 00003AB3 EBF5              <1>       JMP    SHORT SKP_STATE            ; SAVE IT
1560                            <1>
1561                            <1> ;--------------------------------------------------------------------------
1562                            <1> ; MED_CHANGE
1563                            <1> ;     CHECKS FOR MEDIA CHANGE, RESETS MEDIA CHANGE,
1564                            <1> ;     CHECKS MEDIA CHANGE AGAIN.
1565                            <1> ;
1566                            <1> ; ON EXIT:   CY = 1 MEANS MEDIA CHANGE OR TIMEOUT
1567                            <1> ;            @DSKETTE_STATUS = ERROR CODE
1568                            <1> ;--------------------------------------------------------------------------
1569                            <1> MED_CHANGE:
```

```
1570 00003AB5 E888060000          <1>       CALL   READ_DSKCHNG        ; READ DISK CHANCE LINE STATE
1571 00003ABA 7447                <1>       JZ     short MC_OUT        ; BYPASS HANDLING DISK CHANGE LINE
1572 00003ABC 80A7[95520100]EF    <1>       AND    byte [DSK_STATE+eDI], ~MED_DET ; CLEAR STATE FOR THIS DRIVE
1573                              <1>
1574                              <1> ;     THIS SEQUENCE ENSURES WHENEVER A DISKETTE IS CHANGED THAT
1575                              <1> ;     ON THE NEXT OPERATION THE REQUIRED MOTOR START UP TIME WILL
1576                              <1> ;     BE WAITED. (DRIVE MOTOR MAY GO OFF UPON DOOR OPENING).
1577                              <1>
1578 00003AC3 6689F9              <1>       MOV    CX,DI               ; CL = DRIVE 0
1579 00003AC6 B001                <1>       MOV    AL,1                ; MOTOR ON BIT MASK
1580 00003AC8 D2E0                <1>       SHL    AL,CL               ; TO APPROPRIATE POSITION
1581 00003ACA F6D0                <1>       NOT    AL                  ; KEEP ALL BUT MOTOR ON
1582 00003ACC FA                  <1>       CLI                        ; NO INTERRUPTS
1583 00003ACD 2005[86520100]      <1>       AND    [MOTOR_STATUS], AL  ; TURN MOTOR OFF INDICATOR
1584 00003AD3 FB                  <1>       STI                        ; INTERRUPTS ENABLED
1585 00003AD4 E810040000          <1>       CALL   MOTOR_ON            ; TURN MOTOR ON
1586                              <1>
1587                              <1> ;-----      THIS SEQUENCE OF SEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL
1588                              <1>
1589 00003AD9 E86DF9FFFF          <1>       CALL   DSK_RESET           ; RESET NEC
1590 00003ADE B501                <1>       MOV    CH,01H              ; MOVE TO CYLINDER 1
1591 00003AE0 E8FF040000          <1>       CALL   SEEK                ; ISSUE SEEK
1592 00003AE5 30ED                <1>       XOR    CH,CH               ; MOVE TO CYLINDER 0
1593 00003AE7 E8F8040000          <1>       CALL   SEEK                ; ISSUE SEEK
1594 00003AEC C605[88520100]06    <1>       MOV    byte [DSKETTE_STATUS], MEDIA_CHANGE ; STORE IN STATUS
1595                              <1> OK1:
1596 00003AF3 E84A060000          <1>       CALL   READ_DSKCHNG        ; CHECK MEDIA CHANGED AGAIN
1597 00003AF8 7407                <1>       JZ     short OK2           ; IF ACTIVE, NO DISKETTE, TIMEOUT
1598                              <1> OK4:
1599 00003AFA C605[88520100]80    <1>       MOV    byte [DSKETTE_STATUS], TIME_OUT ; TIMEOUT IF DRIVE EMPTY
1600                              <1> OK2:
1601 00003B01 F9                  <1>       STC                        ; MEDIA CHANGED, SET CY
1602 00003B02 C3                  <1>       RETn
1603                              <1> MC_OUT:
1604 00003B03 F8                  <1>       CLC                        ; NO MEDIA CHANGED, CLEAR CY
1605 00003B04 C3                  <1>       RETn
1606                              <1>
1607                              <1> ;--------------------------------------------------------------------------
1608                              <1> ; SEND_RATE
1609                              <1> ;     SENDS DATA RATE COMMAND TO NEC
1610                              <1> ; ON ENTRY:  DI = DRIVE #
1611                              <1> ; ON EXIT:   NONE
1612                              <1> ; REGISTERS ALTERED: DX
1613                              <1> ;--------------------------------------------------------------------------
1614                              <1> SEND_RATE:
1615 00003B05 6650                <1>       PUSH   AX                  ; SAVE REG.
1616 00003B07 8025[90520100]3F    <1>       AND    byte [LASTRATE], ~SEND_MSK ; ELSE CLEAR LAST RATE ATTEMPTED
1617 00003B0E 8A87[95520100]      <1>       MOV    AL, [DSK_STATE+eDI] ; GET RATE STATE OF THIS DRIVE
1618 00003B14 24C0                <1>       AND    AL,SEND_MSK         ; KEEP ONLY RATE BITS
1619 00003B16 0805[90520100]      <1>       OR     [LASTRATE], AL            ; SAVE NEW RATE FOR NEXT CHECK
1620 00003B1C C0C002              <1>       ROL    AL,2                ; MOVE TO BIT OUTPUT POSITIONS
1621 00003B1F 66BAF703            <1>       MOV    DX,03F7H            ; OUTPUT NEW DATA RATE
1622 00003B23 EE                  <1>       OUT    DX,AL
1623 00003B24 6658                <1>       POP    AX                  ; RESTORE REG.
1624 00003B26 C3                  <1>       RETn
1625                              <1>
1626                              <1> ;--------------------------------------------------------------------------
1627                              <1> ; CHK_LASTRATE
1628                              <1> ;     CHECK PREVIOUS DATE RATE SNT TO THE CONTROLLER.
1629                              <1> ; ON ENTRY:
1630                              <1> ;     DI = DRIVE #
1631                              <1> ; ON EXIT:
1632                              <1> ;     ZF =  1 DATA RATE IS THE SAME AS THE LAST RATE SENT TO NEC
1633                              <1> ;     ZF =  0 DATA RATE IS DIFFERENT FROM LAST RATE
1634                              <1> ; REGISTERS ALTERED: DX
1635                              <1> ;--------------------------------------------------------------------------
1636                              <1> CHK_LASTRATE:
1637 00003B27 6650                <1>       PUSH   AX                  ; SAVE REG
1638 00003B29 2225[90520100]      <1>       AND    AH, [LASTRATE]            ; GET LAST DATA RATE SELECTED
1639 00003B2F 8A87[95520100]      <1>       MOV    AL, [DSK_STATE+eDI] ; GET RATE STATE OF THIS DRIVE
1640 00003B35 6625C0C0            <1>         AND     AX, SEND_MSK*257        ; KEEP ONLY RATE BITS OF BOTH
1641 00003B39 38E0                <1>       CMP    AL, AH              ; COMPARE TO PREVIOUSLY TRIED
1642                              <1>                                  ; ZF = 1 RATE IS THE SAME
1643 00003B3B 6658                <1>       POP    AX                  ; RESTORE REG.
1644 00003B3D C3                  <1>       RETn
1645                              <1>
1646                              <1> ;--------------------------------------------------------------------------
1647                              <1> ; DMA_SETUP
1648                              <1> ;     THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY OPERATIONS.
1649                              <1> ;
1650                              <1> ; ON ENTRY:  AL = DMA COMMAND
1651                              <1> ;
1652                              <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1653                              <1> ;--------------------------------------------------------------------------
1654                              <1>
1655                              <1> ; SI = Head #, # of Sectors or DASD Type
1656                              <1>
1657                              <1> ; 22/08/2015
1658                              <1> ; 08/02/2015 - Protected Mode Modification
1659                              <1> ; 06/02/2015 - 07/02/2015
1660                              <1> ; NOTE: Buffer address must be in 1st 16MB of Physical Memory (24 bit limit).
1661                              <1> ; (DMA Addres = Physical Address)
1662                              <1> ; (Retro UNIX 386 v1 Kernel/System Mode Virtual Address = Physical Address)
1663                              <1> ;
1664                              <1>
1665                              <1>
1666                              <1> ; 04/02/2016 (clc)
1667                              <1> ; 20/02/2015 modification (source: AWARD BIOS 1999, DMA_SETUP)
1668                              <1> ; 16/12/2014 (IODELAY)
1669                              <1>
1670                              <1> DMA_SETUP:
1671                              <1>
```

```
                                <1> ;; 20/02/2015
00003B3E 8B5504                 <1>       mov    edx, [ebp+4]        ; Buffer address
00003B41 F7C2000000FF           <1>       test   edx, 0FF000000h           ; 16 MB limit (22/08/2015, bugfix)
00003B47 756E                   <1>       jnz    short dma_bnd_err_stc
                                <1>       ;
00003B49 6650                   <1>       push   ax                  ; DMA command
00003B4B 52                     <1>       push   edx                 ; *
00003B4C B203                   <1>       mov    dl, 3               ; GET BYTES/SECTOR PARAMETER
00003B4E E851030000             <1>       call   GET_PARM            ;
00003B53 88E1                   <1>       mov    cl, ah                   ; SHIFT COUNT (0=128, 1=256, 2=512 ETC)
00003B55 6689F0                 <1>       mov    ax, si              ; Sector count
00003B58 88C4                   <1>       mov    ah, al              ; AH =  # OF SECTORS
00003B5A 28C0                   <1>       sub    al, al              ; AL = 0, AX = # SECTORS * 256
00003B5C 66D1E8                 <1>       shr    ax, 1               ; AX = # SECTORS * 128
00003B5F 66D3E0                 <1>       shl    ax, cl              ; SHIFT BY PARAMETER VALUE
00003B62 6648                   <1>       dec    ax                  ; -1 FOR DMA VALUE
00003B64 6689C1                 <1>       mov    cx, ax
00003B67 5A                     <1>       pop    edx                 ; *
00003B68 6658                   <1>       pop    ax
00003B6A 3C42                   <1>       cmp    al, 42h
00003B6C 7507                   <1>         jne     short NOT_VERF
00003B6E BA0000FF00             <1>       mov    edx, 0FF0000h
00003B73 EB08                   <1>       jmp    short J33
                                <1> NOT_VERF:
00003B75 6601CA                 <1>       add    dx, cx              ; check for overflow
00003B78 723E                   <1>       jc     short dma_bnd_err
                                <1>       ;
00003B7A 6629CA                 <1>       sub    dx, cx              ; Restore start address
                                <1> J33:
00003B7D FA                     <1>       CLI                        ; DISABLE INTERRUPTS DURING DMA SET-UP
00003B7E E60C                   <1>       OUT    DMA+12,AL           ; SET THE FIRST/LA5T F/F
                                <1>       IODELAY                          ; WAIT FOR I/O
00003B80 EB00                   <2>  jmp short $+2
00003B82 EB00                   <2>  jmp short $+2
00003B84 E60B                   <1>       OUT    DMA+11,AL           ; OUTPUT THE MODE BYTE
00003B86 89D0                   <1>       mov    eax, edx            ; Buffer address
00003B88 E604                   <1>       OUT    DMA+4,AL            ; OUTPUT LOW ADDRESS
                                <1>       IODELAY                          ; WAIT FOR I/O
00003B8A EB00                   <2>  jmp short $+2
00003B8C EB00                   <2>  jmp short $+2
00003B8E 88E0                   <1>       MOV    AL,AH
00003B90 E604                   <1>       OUT    DMA+4,AL            ; OUTPUT HIGH ADDRESS
00003B92 C1E810                 <1>       shr    eax, 16
                                <1>       IODELAY                          ; I/O WAIT STATE
00003B95 EB00                   <2>  jmp short $+2
00003B97 EB00                   <2>  jmp short $+2
00003B99 E681                   <1>       OUT    081H,AL                  ; OUTPUT highest BITS TO PAGE REGISTER
                                <1>       IODELAY
00003B9B EB00                   <2>  jmp short $+2
00003B9D EB00                   <2>  jmp short $+2
00003B9F 6689C8                 <1>       mov    ax, cx              ; Byte count - 1
00003BA2 E605                   <1>       OUT    DMA+5,AL            ; LOW BYTE OF COUNT
                                <1>       IODELAY                          ; WAIT FOR I/O
00003BA4 EB00                   <2>  jmp short $+2
00003BA6 EB00                   <2>  jmp short $+2
00003BA8 88E0                   <1>       MOV    AL, AH
00003BAA E605                   <1>       OUT    DMA+5,AL            ; HIGH BYTE OF COUNT
                                <1>       IODELAY
00003BAC EB00                   <2>  jmp short $+2
00003BAE EB00                   <2>  jmp short $+2
00003BB0 FB                     <1>       STI                        ; RE-ENABLE INTERRUPTS
00003BB1 B002                   <1>       MOV    AL, 2               ; MODE FOR 8237
00003BB3 E60A                   <1>       OUT    DMA+10, AL          ; INITIALIZE THE DISKETTE CHANNEL
                                <1>
00003BB5 F8                     <1>       clc    ; 04/02/2016
00003BB6 C3                     <1>       retn
                                <1>
                                <1> dma_bnd_err_stc:
00003BB7 F9                     <1>       stc
                                <1> dma_bnd_err:
00003BB8 C605[88520100]09       <1>       MOV    byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
00003BBF C3                     <1>       RETn                       ; CY SET BY ABOVE IF ERROR
                                <1>
                                <1> ;; 16/12/2014
                                <1> ;;    CLI                        ; DISABLE INTERRUPTS DURING DMA SET-UP
                                <1> ;;    OUT    DMA+12,AL           ; SET THE FIRST/LA5T F/F
                                <1> ;;    ;JMP   $+2                 ; WAIT FOR I/O
                                <1> ;;    IODELAY
                                <1> ;;    OUT    DMA+11,AL           ; OUTPUT THE MODE BYTE
                                <1> ;;    ;SIODELAY
                                <1> ;;      ;CMP AL, 42H                  ; DMA VERIFY COMMAND
                                <1> ;;      ;JNE short NOT_VERF           ; NO
                                <1> ;;      ;XOR AX, AX             ; START ADDRESS
                                <1> ;;      ;JMP SHORT J33
                                <1> ;;;NOT_VERF:
                                <1> ;;    ;MOV   AX,ES               ; GET THE ES VALUE
                                <1> ;;    ;ROL   AX,4                ; ROTATE LEFT
                                <1> ;;    ;MOV   CH,AL               ; GET HIGHEST NIBBLE OF ES TO CH
                                <1> ;;    ;AND   AL,11110000B        ; ZERO THE LOW NIBBLE FROM SEGMENT
                                <1> ;;    ;ADD   AX,[BP+2]           ; TEST FOR CARRY FROM ADDITION
                                <1> ;;    mov    eax, [ebp+4] ; 06/02/2015
                                <1> ;;    ;JNC   short J33
                                <1> ;;    ;INC   CH                  ; CARRY MEANS HIGH 4 BITS MUST BE INC
                                <1> ;;;J33:
                                <1> ;;    PUSH   eAX                 ; SAVE START ADDRESS
                                <1> ;;    OUT    DMA+4,AL            ; OUTPUT LOW ADDRESS
                                <1> ;;    ;JMP   $+2                 ; WAIT FOR I/O
                                <1> ;;    IODELAY
                                <1> ;;    MOV    AL,AH
                                <1> ;;    OUT    DMA+4,AL            ; OUTPUT HIGH ADDRESS
                                <1> ;;    shr    eax, 16           ; 07/02/2015
                                <1> ;;    ;MOV   AL,CH               ; GET HIGH 4 BITS
```

```
1762                            <1> ;;    ;JMP   $+2                ; I/O WAIT STATE
1763                            <1> ;;    IODELAY
1764                            <1> ;;    ;AND   AL,00001111B
1765                            <1> ;;    OUT    081H,AL                  ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
1766                            <1> ;;    ;SIODELAY
1767                            <1> ;;
1768                            <1> ;;;----- DETERMINE COUNT
1769                            <1> ;;    sub    eax, eax ; 08/02/2015
1770                            <1> ;;    MOV    AX, SI             ; AL =  # OF SECTORS
1771                            <1> ;;    XCHG   AL, AH             ; AH =  # OF SECTORS
1772                            <1> ;;    SUB    AL, AL             ; AL = 0, AX = # SECTORS * 256
1773                            <1> ;;    SHR    AX, 1              ; AX = # SECTORS * 128
1774                            <1> ;;    PUSH   AX                 ; SAVE # OF SECTORS * 128
1775                            <1> ;;    MOV    DL, 3              ; GET BYTES/SECTOR PARAMETER
1776                            <1> ;;    CALL   GET_PARM           ; "
1777                            <1> ;;    MOV    CL,AH              ; SHIFT COUNT (0=128, 1=256, 2=512 ETC)
1778                            <1> ;;    POP    AX                 ; AX = # SECTORS * 128
1779                            <1> ;;    SHL    AX,CL              ; SHIFT BY PARAMETER VALUE
1780                            <1> ;;    DEC    AX                 ; -1 FOR DMA VALUE
1781                            <1> ;;    PUSH   eAX  ; 08/02/2015  ; SAVE COUNT VALUE
1782                            <1> ;;    OUT    DMA+5,AL           ; LOW BYTE OF COUNT
1783                            <1> ;;    ;JMP   $+2                ; WAIT FOR I/O
1784                            <1> ;;    IODELAY
1785                            <1> ;;    MOV    AL, AH
1786                            <1> ;;    OUT    DMA+5,AL           ; HIGH BYTE OF COUNT
1787                            <1> ;;    ;IODELAY
1788                            <1> ;;    STI                       ; RE-ENABLE INTERRUPTS
1789                            <1> ;;    POP    eCX  ; 08/02/2015  ; RECOVER COUNT VALUE
1790                            <1> ;;    POP    eAX  ; 08/02/2015  ; RECOVER ADDRESS VALUE
1791                            <1> ;;    ;ADD   AX, CX             ; ADD, TEST FOR 64K OVERFLOW
1792                            <1> ;;    add    ecx, eax ; 08/02/2015
1793                            <1> ;;    MOV    AL, 2              ; MODE FOR 8237
1794                            <1> ;;    ;JMP   $+2                ; WAIT FOR I/O
1795                            <1> ;;    SIODELAY
1796                            <1> ;;    OUT    DMA+10, AL         ; INITIALIZE THE DISKETTE CHANNEL
1797                            <1> ;;    ;JNC   short NO_BAD       ; CHECK FOR ERROR
1798                            <1> ;;    jc     short dma_bnd_err ; 08/02/2015
1799                            <1> ;;    and    ecx, 0FFF00000h ; 16 MB limit
1800                            <1> ;;    jz     short NO_BAD
1801                            <1> ;;dma_bnd_err:
1802                            <1> ;;    MOV    byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
1803                            <1> ;;NO_BAD:
1804                            <1> ;;    RETn                      ; CY SET BY ABOVE IF ERROR
1805                            <1>
1806                            <1> ;--------------------------------------------------------------------------
1807                            <1> ; FMTDMA_SET
1808                            <1> ;     THIS ROUTINE SETS UP THE DMA CONTROLLER FOR A FORMAT OPERATION.
1809                            <1> ;
1810                            <1> ; ON ENTRY:  NOTHING REQUIRED
1811                            <1> ;
1812                            <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1813                            <1> ;--------------------------------------------------------------------------
1814                            <1>
1815                            <1> FMTDMA_SET:
1816                            <1> ;; 20/02/2015 modification
1817 00003BC0 8B5504           <1>       mov    edx, [ebp+4]       ; Buffer address
1818 00003BC3 F7C20000F0FF     <1>       test   edx, 0FFF00000h          ; 16 MB limit
1819 00003BC9 75EC             <1>       jnz    short dma_bnd_err_stc
1820                           <1>       ;
1821 00003BCB 6652             <1>       push   dx                 ; *
1822 00003BCD B204             <1>       mov    DL, 4              ; SECTORS/TRACK VALUE IN PARM TABLE
1823 00003BCF E8D0020000       <1>       call   GET_PARM           ; "
1824 00003BD4 88E0             <1>       mov    al, ah             ; AL = SECTORS/TRACK VALUE
1825 00003BD6 28E4             <1>       sub    ah, ah             ; AX = SECTORS/TRACK VALUE
1826 00003BD8 66C1E002         <1>       shl    ax, 2              ; AX = SEC/TRK * 4 (OFFSET C,H,R,N)
1827 00003BDC 6648             <1>       dec    ax                 ; -1 FOR DMA VALUE
1828 00003BDE 6689C1           <1>       mov    cx, ax
1829 00003BE1 665A             <1>       pop    dx                 ; *
1830 00003BE3 6601CA           <1>       add    dx, cx             ; check for overflow
1831 00003BE6 72D0             <1>       jc     short dma_bnd_err
1832                           <1>       ;
1833 00003BE8 6629CA           <1>       sub    dx, cx             ; Restore start address
1834                           <1>       ;
1835 00003BEB B04A             <1>       MOV    AL, 04AH           ; WILL WRITE TO THE DISKETTE
1836 00003BED FA               <1>       CLI                       ; DISABLE INTERRUPTS DURING DMA SET-UP
1837 00003BEE E60C             <1>       OUT    DMA+12,AL          ; SET THE FIRST/LA5T F/F
1838                           <1>       IODELAY                         ; WAIT FOR I/O
1838 00003BF0 EB00             <2>  jmp short $+2
1838 00003BF2 EB00             <2>  jmp short $+2
1839 00003BF4 E60B             <1>       OUT    DMA+11,AL          ; OUTPUT THE MODE BYTE
1840 00003BF6 89D0             <1>       mov    eax, edx           ; Buffer address
1841 00003BF8 E604             <1>       OUT    DMA+4,AL           ; OUTPUT LOW ADDRESS
1842                           <1>       IODELAY                         ; WAIT FOR I/O
1842 00003BFA EB00             <2>  jmp short $+2
1842 00003BFC EB00             <2>  jmp short $+2
1843 00003BFE 88E0             <1>       MOV    AL,AH
1844 00003C00 E604             <1>       OUT    DMA+4,AL           ; OUTPUT HIGH ADDRESS
1845 00003C02 C1E810           <1>       shr    eax, 16
1846                           <1>       IODELAY                         ; I/O WAIT STATE
1846 00003C05 EB00             <2>  jmp short $+2
1846 00003C07 EB00             <2>  jmp short $+2
1847 00003C09 E681             <1>       OUT    081H,AL                  ; OUTPUT highest BITS TO PAGE REGISTER
1848                           <1>       IODELAY
1848 00003C0B EB00             <2>  jmp short $+2
1848 00003C0D EB00             <2>  jmp short $+2
1849 00003C0F 6689C8           <1>       mov    ax, cx             ; Byte count - 1
1850 00003C12 E605             <1>       OUT    DMA+5,AL           ; LOW BYTE OF COUNT
1851                           <1>       IODELAY                         ; WAIT FOR I/O
1851 00003C14 EB00             <2>  jmp short $+2
1851 00003C16 EB00             <2>  jmp short $+2
1852 00003C18 88E0             <1>       MOV    AL, AH
1853 00003C1A E605             <1>       OUT    DMA+5,AL           ; HIGH BYTE OF COUNT
```

```
1854                                    <1>       IODELAY
1854 00003C1C EB00                      <2>  jmp short $+2
1854 00003C1E EB00                      <2>  jmp short $+2
1855 00003C20 FB                        <1>       STI                     ; RE-ENABLE INTERRUPTS
1856 00003C21 B002                      <1>       MOV    AL, 2            ; MODE FOR 8237
1857 00003C23 E60A                      <1>       OUT    DMA+10, AL       ; INITIALIZE THE DISKETTE CHANNEL
1858 00003C25 C3                        <1>       retn
1859                                    <1>
1860                                    <1> ;; 08/02/2015 - Protected Mode Modification
1861                                    <1> ;;    MOV    AL, 04AH         ; WILL WRITE TO THE DISKETTE
1862                                    <1> ;;    CLI                     ; DISABLE INTERRUPTS DURING DMA SET-UP
1863                                    <1> ;;    OUT    DMA+12,AL        ; SET THE FIRST/LA5T F/F
1864                                    <1> ;;    ;JMP   $+2              ; WAIT FOR I/O
1865                                    <1> ;;    IODELAY
1866                                    <1> ;;    OUT    DMA+11,AL        ; OUTPUT THE MODE BYTE
1867                                    <1> ;;    ;MOV   AX,ES            ; GET THE ES VALUE
1868                                    <1> ;;    ;ROL   AX,4             ; ROTATE LEFT
1869                                    <1> ;;    ;MOV   CH,AL            ; GET HIGHEST NIBBLE OF ES TO CH
1870                                    <1> ;;    ;AND   AL,11110000B     ; ZERO THE LOW NIBBLE FROM SEGMENT
1871                                    <1> ;;    ;ADD   AX,[BP+2]        ; TEST FOR CARRY FROM ADDITION
1872                                    <1> ;;    ;JNC   short J33A
1873                                    <1> ;;    ;INC   CH               ; CARRY MEANS HIGH 4 BITS MUST BE INC
1874                                    <1> ;;    mov    eax, [ebp+4] ; 08/02/2015
1875                                    <1> ;;;J33A:
1876                                    <1> ;;    PUSH   eAX ; 08/02/2015 ; SAVE START ADDRESS
1877                                    <1> ;;    OUT    DMA+4,AL         ; OUTPUT LOW ADDRESS
1878                                    <1> ;;    ;JMP   $+2              ; WAIT FOR I/O
1879                                    <1> ;;    IODELAY
1880                                    <1> ;;    MOV    AL,AH
1881                                    <1> ;;    OUT    DMA+4,AL         ; OUTPUT HIGH ADDRESS
1882                                    <1> ;;    shr    eax, 16 ; 08/02/2015
1883                                    <1> ;;    ;MOV   AL,CH            ; GET HIGH 4 BITS
1884                                    <1> ;;    ;JMP   $+2              ; I/O WAIT STATE
1885                                    <1> ;;    IODELAY
1886                                    <1> ;;    ;AND   AL,00001111B
1887                                    <1> ;;    OUT    081H,AL                  ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
1888                                    <1> ;;
1889                                    <1> ;;;----- DETERMINE COUNT
1890                                    <1> ;;    sub    eax, eax ; 08/02/2015
1891                                    <1> ;;    MOV    DL, 4            ; SECTORS/TRACK VALUE IN PARM TABLE
1892                                    <1> ;;    CALL   GET_PARM         ; "
1893                                    <1> ;;    XCHG   AL, AH           ; AL = SECTORS/TRACK VALUE
1894                                    <1> ;;    SUB    AH, AH           ; AX = SECTORS/TRACK VALUE
1895                                    <1> ;;    SHL    AX, 2            ; AX = SEC/TRK * 4 (OFFSET C,H,R,N)
1896                                    <1> ;;    DEC    AX               ; -1 FOR DMA VALUE
1897                                    <1> ;;    PUSH   eAX    ; 08/02/2015 ; SAVE # OF BYTES TO BE TRANSFERED
1898                                    <1> ;;    OUT    DMA+5,AL         ; LOW BYTE OF COUNT
1899                                    <1> ;;    ;JMP   $+2              ; WAIT FOR I/O
1900                                    <1> ;;    IODELAY
1901                                    <1> ;;    MOV    AL, AH
1902                                    <1> ;;    OUT    DMA+5,AL         ; HIGH BYTE OF COUNT
1903                                    <1> ;;    STI                     ; RE-ENABLE INTERRUPTS
1904                                    <1> ;;    POP    eCX    ; 08/02/2015 ; RECOVER COUNT VALUE
1905                                    <1> ;;    POP    eAX    ; 08/02/2015 ; RECOVER ADDRESS VALUE
1906                                    <1> ;;    ;ADD   AX, CX           ; ADD, TEST FOR 64K OVERFLOW
1907                                    <1> ;;    add    ecx, eax ; 08/02/2015
1908                                    <1> ;;    MOV    AL, 2            ; MODE FOR 8237
1909                                    <1> ;;    ;JMP   $+2              ; WAIT FOR I/O
1910                                    <1> ;;    SIODELAY
1911                                    <1> ;;    OUT    DMA+10, AL       ; INITIALIZE THE DISKETTE CHANNEL
1912                                    <1> ;;    ;JNC   short FMTDMA_OK          ; CHECK FOR ERROR
1913                                    <1> ;;    jc     short fmtdma_bnd_err ; 08/02/2015
1914                                    <1> ;;    and    ecx, 0FFF00000h  ; 16 MB limit
1915                                    <1> ;;    jz     short FMTDMA_OK
1916                                    <1> ;;    stc    ; 20/02/2015
1917                                    <1> ;;fmtdma_bnd_err:
1918                                    <1> ;;    MOV    byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
1919                                    <1> ;;FMTDMA_OK:
1920                                    <1> ;;    RETn                    ; CY SET BY ABOVE IF ERROR
1921                                    <1>
1922                                    <1> ;--------------------------------------------------------------------------------
1923                                    <1> ; NEC_INIT
1924                                    <1> ;     THIS ROUTINE SEEKS TO THE REQUESTED TRACK AND INITIALIZES
1925                                    <1> ;     THE NEC FOR THE READ/WRITE/VERIFY/FORMAT OPERATION.
1926                                    <1> ;
1927                                    <1> ; ON ENTRY:  AH = NEC COMMAND TO BE PERFORMED
1928                                    <1> ;
1929                                    <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1930                                    <1> ;--------------------------------------------------------------------------------
1931                                    <1> NEC_INIT:
1932 00003C26 6650                     <1>       PUSH   AX               ; SAVE NEC COMMAND
1933 00003C28 E8BC020000               <1>       CALL   MOTOR_ON         ; TURN MOTOR ON FOR SPECIFIC DRIVE
1934                                    <1>
1935                                    <1> ;-----      DO THE SEEK OPERATION
1936                                    <1>
1937 00003C2D 8A6D01                   <1>       MOV    CH,[eBP+1]       ; CH = TRACK #
1938 00003C30 E8AF030000               <1>       CALL   SEEK             ; MOVE TO CORRECT TRACK
1939 00003C35 6658                     <1>       POP    AX               ; RECOVER COMMAND
1940 00003C37 721E                     <1>       JC     short ER_1       ; ERROR ON SEEK
1941 00003C39 BB[573C0000]             <1>       MOV    eBX, ER_1        ; LOAD ERROR ADDRESS
1942 00003C3E 53                       <1>       PUSH   eBX              ; PUSH NEC_OUT ERROR RETURN
1943                                    <1>
1944                                    <1> ;-----      SEND OUT THE PARAMETERS TO THE CONTROLLER
1945                                    <1>
1946 00003C3F E866030000               <1>       CALL   NEC_OUTPUT       ; OUTPUT THE OPERATION COMMAND
1947 00003C44 6689F0                   <1>       MOV    AX,SI            ; AH = HEAD #
1948 00003C47 89FB                     <1>       MOV    eBX,eDI                ; BL = DRIVE #
1949 00003C49 C0E402                   <1>       SAL    AH,2             ; MOVE IT TO BIT 2
1950 00003C4C 80E404                   <1>       AND    AH,00000100B     ; ISOLATE THAT BIT
1951 00003C4F 08DC                     <1>       OR     AH,BL            ; OR IN THE DRIVE NUMBER
1952 00003C51 E854030000               <1>       CALL   NEC_OUTPUT       ; FALL THRU CY SET IF ERROR
1953 00003C56 5B                       <1>       POP    eBX              ; THROW AWAY ERROR RETURN
```

```
1954                                    <1> ER_1:
1955 00003C57 C3                        <1>         RETn
1956                                    <1>
1957                                    <1> ;--------------------------------------------------------------------------------
1958                                    <1> ; RWV_COM
1959                                    <1> ;       THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC TO THE
1960                                    <1> ;       READ/WRITE/VERIFY OPERATIONS.
1961                                    <1> ;
1962                                    <1> ; ON ENTRY:  CS:BX = ADDRESS OF MEDIA/DRIVE PARAMETER TABLE
1963                                    <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1964                                    <1> ;--------------------------------------------------------------------------------
1965                                    <1> RWV_COM:
1966 00003C58 B8[A33C0000]              <1>         MOV     eAX, ER_2           ; LOAD ERROR ADDRESS
1967 00003C5D 50                        <1>         PUSH    eAX                 ; PUSH NEC_OUT ERROR RETURN
1968 00003C5E 8A6501                    <1>         MOV     AH,[eBP+1]          ; OUTPUT TRACK #
1969 00003C61 E844030000                <1>         CALL    NEC_OUTPUT
1970 00003C66 6689F0                    <1>         MOV     AX,SI               ; OUTPUT HEAD #
1971 00003C69 E83C030000                <1>         CALL    NEC_OUTPUT
1972 00003C6E 8A6500                    <1>           MOV     AH,[eBP]                ; OUTPUT SECTOR #
1973 00003C71 E834030000                <1>         CALL    NEC_OUTPUT
1974 00003C76 B203                      <1>         MOV     DL,3                ; BYTES/SECTOR PARAMETER FROM BLOCK
1975 00003C78 E827020000                <1>         CALL    GET_PARM            ; ... TO THE NEC
1976 00003C7D E828030000                <1>         CALL    NEC_OUTPUT          ; OUTPUT TO CONTROLLER
1977 00003C82 B204                      <1>         MOV     DL,4                ; EOT PARAMETER FROM BLOCK
1978 00003C84 E81B020000                <1>         CALL    GET_PARM            ; ... TO THE NEC
1979 00003C89 E81C030000                <1>         CALL    NEC_OUTPUT          ; OUTPUT TO CONTROLLER
1980 00003C8E 8A6305                    <1>           MOV     AH, [eBX+MD.GAP]        ; GET GAP LENGTH
1981                                    <1> _R15:
1982 00003C91 E814030000                <1>         CALL    NEC_OUTPUT
1983 00003C96 B206                      <1>         MOV     DL,6                ; DTL PARAMETER PROM BLOCK
1984 00003C98 E807020000                <1>         CALL    GET_PARM            ;  TO THE NEC
1985 00003C9D E808030000                <1>         CALL    NEC_OUTPUT          ; OUTPUT TO CONTROLLER
1986 00003CA2 58                        <1>         POP     eAX                 ; THROW AWAY ERROR EXIT
1987                                    <1> ER_2:
1988 00003CA3 C3                        <1>         RETn
1989                                    <1>
1990                                    <1> ;--------------------------------------------------------------------------------
1991                                    <1> ; NEC_TERM
1992                                    <1> ;       THIS ROUTINE WAITS FOR THE OPERATION THEN ACCEPTS THE STATUS
1993                                    <1> ;       FROM THE NEC FOR THE READ/WRITE/VERIFY/FORWAT OPERATION.
1994                                    <1> ;
1995                                    <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1996                                    <1> ;--------------------------------------------------------------------------------
1997                                    <1> NEC_TERM:
1998                                    <1>
1999                                    <1> ;-----      LET THE OPERATION HAPPEN
2000                                    <1>
2001 00003CA4 56                        <1>         PUSH    eSI                 ; SAVE HEAD #, # OF SECTORS
2002 00003CA5 E80D040000                <1>         CALL    WAIT_INT            ; WAIT FOR THE INTERRUPT
2003 00003CAA 9C                        <1>         PUSHF
2004 00003CAB E837040000                <1>         CALL    RESULTS                 ; GET THE NEC STATUS
2005 00003CB0 724B                      <1>         JC      short SET_END_POP
2006 00003CB2 9D                        <1>         POPF
2007 00003CB3 723E                      <1>         JC      short SET_END       ; LOOK FOR ERROR
2008                                    <1>
2009                                    <1> ;-----      CHECK THE RESULTS RETURNED BY THE CONTROLLER
2010                                    <1>
2011 00003CB5 FC                        <1>         CLD                         ; SET THE CORRECT DIRECTION
2012 00003CB6 BE[89520100]              <1>         MOV     eSI, NEC_STATUS             ; POINT TO STATUS FIELD
2013 00003CBB AC                        <1>         lodsb                       ; GET ST0
2014 00003CBC 24C0                      <1>         AND     AL,11000000B        ; TEST FOR NORMAL TERMINATION
2015 00003CBE 7433                      <1>         JZ      short SET_END
2016 00003CC0 3C40                      <1>         CMP     AL,01000000B        ; TEST FOR ABNORMAL TERMINATION
2017 00003CC2 7527                      <1>         JNZ     short J18           ; NOT ABNORMAL, BAD NEC
2018                                    <1>
2019                                    <1> ;-----      ABNORMAL TERMINATION, FIND OUT WHY
2020                                    <1>
2021 00003CC4 AC                        <1>         lodsb                       ; GET ST1
2022 00003CC5 D0E0                      <1>         SAL     AL,1                ; TEST FOR EDT FOUND
2023 00003CC7 B404                      <1>         MOV     AH,RECORD_NOT_FND
2024 00003CC9 7222                      <1>         JC      short J19
2025 00003CCB C0E002                    <1>         SAL     AL,2
2026 00003CCE B410                      <1>         MOV     AH,BAD_CRC
2027 00003CD0 721B                      <1>         JC      short J19
2028 00003CD2 D0E0                      <1>         SAL     AL,1                ; TEST FOR DMA OVERRUN
2029 00003CD4 B408                      <1>         MOV     AH,BAD_DMA
2030 00003CD6 7215                      <1>         JC      short J19
2031 00003CD8 C0E002                    <1>         SAL     AL,2                ; TEST FOR RECORD NOT FOUND
2032 00003CDB B404                      <1>         MOV     AH,RECORD_NOT_FND
2033 00003CDD 720E                      <1>         JC      short J19
2034 00003CDF D0E0                      <1>         SAL     AL,1
2035 00003CE1 B403                      <1>         MOV     AH,WRITE_PROTECT    ; TEST FOR WRITE_PROTECT
2036 00003CE3 7208                      <1>         JC      short J19
2037 00003CE5 D0E0                      <1>         SAL     AL,1                ; TEST MISSING ADDRESS MARK
2038 00003CE7 B402                      <1>         MOV     AH,BAD_ADDR_MARK
2039 00003CE9 7202                      <1>         JC      short J19
2040                                    <1>
2041                                    <1> ;-----      NEC MUST HAVE FAILED
2042                                    <1> J18:
2043 00003CEB B420                      <1>         MOV     AH,BAD_NEC
2044                                    <1> J19:
2045 00003CED 0825[88520100]            <1>         OR      [DSKETTE_STATUS], AH
2046                                    <1> SET_END:
2047 00003CF3 803D[88520100]01          <1>         CMP     byte [DSKETTE_STATUS], 1 ; SET ERROR CONDITION
2048 00003CFA F5                        <1>         CMC
2049 00003CFB 5E                        <1>         POP     eSI
2050 00003CFC C3                        <1>         RETn                        ; RESTORE HEAD #, # OF SECTORS
2051                                    <1>
2052                                    <1> SET_END_POP:
2053 00003CFD 9D                        <1>         POPF
2054 00003CFE EBF3                      <1>         JMP     SHORT SET_END
2055                                    <1>
```

```
                        <1> ;--------------------------------------------------------------------------
                        <1> ; DSTATE:   ESTABLISH STATE UPON SUCCESSFUL OPERATION.
                        <1> ;--------------------------------------------------------------------------
                        <1> DSTATE:
00003D00 803D[88520100]00   <1>       CMP   byte [DSKETTE_STATUS],0   ; CHECK FOR ERROR
00003D07 753E               <1>       JNZ   short SETBAC            ; IF ERROR JUMP
00003D09 808F[95520100]10   <1>       OR    byte [DSK_STATE+eDI],MED_DET ; NO ERROR, MARK MEDIA AS DETERMINED
00003D10 F687[95520100]04   <1>       TEST  byte [DSK_STATE+eDI],DRV_DET ; DRIVE DETERMINED ?
00003D17 752E               <1>       JNZ   short SETBAC        ; IF DETERMINED NO TRY TO DETERMINE
00003D19 8A87[95520100]     <1>       MOV   AL,[DSK_STATE+eDI]  ; LOAD STATE
00003D1F 24C0               <1>       AND   AL,RATE_MSK         ; KEEP ONLY RATE
00003D21 3C80               <1>       CMP   AL,RATE_250         ; RATE 250 ?
00003D23 751B               <1>       JNE   short M_12          ; NO, MUST BE 1.2M OR 1.44M DRIVE
                        <1>
                        <1> ;-----      CHECK IF IT IS 1.44M
                        <1>
00003D25 E871010000         <1>       CALL  CMOS_TYPE           ; RETURN DRIVE TYPE IN (AL)
                        <1>       ;;20/02/2015
                        <1>       ;;JC  short M_12          ; CMOS BAD
00003D2A 7414               <1>       jz    short M_12 ;; 20/02/2015
00003D2C 3C04               <1>       CMP   AL, 4               ; 1.44MB DRIVE ?
00003D2E 7410               <1>       JE    short M_12          ; YES
                        <1> M_720:
00003D30 80A7[95520100]FD   <1>       AND   byte [DSK_STATE+eDI], ~FMT_CAPA ; TURN OFF FORMAT CAPABILITY
00003D37 808F[95520100]04   <1>       OR    byte [DSK_STATE+eDI],DRV_DET  ; MARK DRIVE DETERMINED
00003D3E EB07               <1>       JMP   SHORT SETBAC        ; BACK
                        <1> M_12:
00003D40 808F[95520100]06   <1>       OR    byte [DSK_STATE+eDI],DRV_DET+FMT_CAPA
                        <1>                                     ; TURN ON DETERMINED & FMT CAPA
                        <1> SETBAC:
00003D47 C3                 <1>       RETn
                        <1>
                        <1> ;--------------------------------------------------------------------------
                        <1> ; RETRY
                        <1> ;     DETERMINES WHETHER A RETRY IS NECESSARY.
                        <1> ;     IF RETRY IS REQUIRED THEN STATE INFORMATION IS UPDATED FOR RETRY.
                        <1> ;
                        <1> ; ON EXIT:  CY = 1 FOR RETRY, CY = 0 FOR NO RETRY
                        <1> ;--------------------------------------------------------------------------
                        <1> RETRY:
00003D48 803D[88520100]00   <1>       CMP   byte [DSKETTE_STATUS],0   ; GET STATUS OF OPERATION
00003D4F 7445               <1>       JZ    short NO_RETRY          ; SUCCESSFUL OPERATION
00003D51 803D[88520100]80   <1>       CMP   byte [DSKETTE_STATUS],TIME_OUT ; IF TIME OUT NO RETRY
00003D58 743C               <1>       JZ    short NO_RETRY
00003D5A 8AA7[95520100]     <1>       MOV   AH,[DSK_STATE+eDI]  ; GET MEDIA STATE OF DRIVE
00003D60 F6C410             <1>       TEST  AH,MED_DET          ; ESTABLISHED/DETERMINED ?
00003D63 7531               <1>       JNZ   short NO_RETRY          ; IF ESTABLISHED STATE THEN TRUE ERROR
00003D65 80E4C0             <1>       AND   AH,RATE_MSK         ; ISOLATE RATE
00003D68 8A2D[90520100]     <1>       MOV   CH,[LASTRATE]       ; GET START OPERATION STATE
00003D6E C0C504             <1>       ROL   CH,4                ; TO CORRESPONDING BITS
00003D71 80E5C0             <1>       AND   CH,RATE_MSK         ; ISOLATE RATE BITS
00003D74 38E5               <1>       CMP   CH,AH               ; ALL RATES TRIED
00003D76 741E               <1>       JE    short NO_RETRY          ; IF YES, THEN TRUE ERROR
                        <1>
                        <1> ;     SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE
                        <1> ;      00000000B (500) -> 10000000B    (250)
                        <1> ;      10000000B (250) -> 01000000B    (300)
                        <1> ;      01000000B (300) -> 00000000B    (500)
                        <1>
00003D78 80FC01             <1>       CMP   AH,RATE_500+1       ; SET CY FOR RATE 500
00003D7B D0DC               <1>       RCR   AH,1                ; TO NEXT STATE
00003D7D 80E4C0             <1>       AND   AH,RATE_MSK         ; KEEP ONLY RATE BITS
00003D80 80A7[95520100]1F   <1>       AND   byte [DSK_STATE+eDI], ~(RATE_MSK+DBL_STEP)
                        <1>                                     ; RATE, DBL STEP OFF
00003D87 08A7[95520100]     <1>       OR    [DSK_STATE+eDI],AH  ; TURN ON NEW RATE
00003D8D C605[88520100]00   <1>       MOV   byte [DSKETTE_STATUS],0   ; RESET STATUS FOR RETRY
00003D94 F9                 <1>       STC                       ; SET CARRY FOR RETRY
00003D95 C3                 <1>       RETn                      ; RETRY RETURN
                        <1>
                        <1> NO_RETRY:
00003D96 F8                 <1>       CLC                       ; CLEAR CARRY NO RETRY
00003D97 C3                 <1>       RETn                      ; NO RETRY RETURN
                        <1>
                        <1> ;--------------------------------------------------------------------------
                        <1> ; NUM_TRANS
                        <1> ;     THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT WERE
                        <1> ;     ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.
                        <1> ;
                        <1> ; ON ENTRY:  [BP+1] = TRACK
                        <1> ;            SI-HI  = HEAD
                        <1> ;            [BP]   = START SECTOR
                        <1> ;
                        <1> ; ON EXIT:   AL = NUMBER ACTUALLY TRANSFERRED
                        <1> ;--------------------------------------------------------------------------
                        <1> NUM_TRANS:
00003D98 30C0               <1>       XOR   AL,AL               ; CLEAR FOR ERROR
00003D9A 803D[88520100]00   <1>       CMP   byte [DSKETTE_STATUS],0   ; CHECK FOR ERROR
00003DA1 752C               <1>       JNZ   NT_OUT              ; IF ERROR 0 TRANSFERRED
00003DA3 B204               <1>       MOV   DL,4                ; SECTORS/TRACK OFFSET TO DL
00003DA5 E8FA000000         <1>       CALL  GET_PARM            ; AH = SECTORS/TRACK
00003DAA 8A1D[8E520100]     <1>       MOV   BL, [NEC_STATUS+5]  ; GET ENDING SECTOR
00003DB0 6689F1             <1>       MOV   CX,SI               ; CH = HEAD # STARTED
00003DB3 3A2D[8D520100]     <1>       CMP   CH, [NEC_STATUS+4]  ; GET HEAD ENDED UP ON
00003DB9 750D               <1>       JNZ   DIF_HD              ; IF ON SAME HEAD, THEN NO ADJUST
00003DBB 8A2D[8C520100]     <1>       MOV   CH, [NEC_STATUS+3]  ; GET TRACK ENDED UP ON
00003DC1 3A6D01             <1>       CMP   CH,[eBP+1]          ; IS IT ASKED FOR TRACK
00003DC4 7404               <1>       JZ    short SAME_TRK          ; IF SAME TRACK NO INCREASE
00003DC6 00E3               <1>       ADD   BL,AH               ; ADD SECTORS/TRACK
                        <1> DIF_HD:
00003DC8 00E3               <1>       ADD   BL,AH               ; ADD SECTORS/TRACK
                        <1> SAME_TRK:
00003DCA 2A5D00             <1>       SUB   BL,[eBP]            ; SUBTRACT START FROM END
```

```
2158 00003DCD 88D8               <1>       MOV    AL,BL               ; TO AL
2159                             <1> NT_OUT:
2160 00003DCF C3                 <1>       RETn
2161                             <1>
2162                             <1> ;--------------------------------------------------------------------------
2163                             <1> ; SETUP_END
2164                             <1> ;     RESTORES @MOTOR_COUNT TO PARAMETER PROVIDED IN TABLE
2165                             <1> ;     AND LOADS @DSKETTE_STATUS TO AH, AND SETS CY.
2166                             <1> ;
2167                             <1> ; ON EXIT:
2168                             <1> ;     AH, @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
2169                             <1> ;--------------------------------------------------------------------------
2170                             <1> SETUP_END:
2171 00003DD0 B202               <1>       MOV    DL,2                ; GET THE MOTOR WAIT PARAMETER
2172 00003DD2 6650               <1>       PUSH   AX                  ; SAVE NUMBER TRANSFERRED
2173 00003DD4 E8CB000000         <1>       CALL   GET_PARM
2174 00003DD9 8825[87520100]     <1>       MOV    [MOTOR_COUNT],AH    ; STORE UPON RETURN
2175 00003DDF 6658               <1>       POP    AX                  ; RESTORE NUMBER TRANSFERRED
2176 00003DE1 8A25[88520100]     <1>       MOV    AH, [DSKETTE_STATUS]      ; GET STATUS OF OPERATION
2177 00003DE7 08E4               <1>       OR     AH,AH               ; CHECK FOR ERROR
2178 00003DE9 7402               <1>       JZ     short NUN_ERR       ; NO ERROR
2179 00003DEB 30C0               <1>       XOR    AL,AL               ; CLEAR NUMBER RETURNED
2180                             <1> NUN_ERR:
2181 00003DED 80FC01             <1>       CMP    AH,1                ; SET THE CARRY FLAG TO INDICATE
2182 00003DF0 F5                 <1>       CMC                        ; SUCCESS OR FAILURE
2183 00003DF1 C3                 <1>       RETn
2184                             <1>
2185                             <1> ;--------------------------------------------------------------------------
2186                             <1> ; SETUP_DBL
2187                             <1> ;     CHECK DOUBLE STEP.
2188                             <1> ;
2189                             <1> ; ON ENTRY : DI = DRIVE
2190                             <1> ;
2191                             <1> ; ON EXIT :  CY = 1 MEANS ERROR
2192                             <1> ;--------------------------------------------------------------------------
2193                             <1> SETUP_DBL:
2194 00003DF2 8AA7[95520100]     <1>       MOV    AH, [DSK_STATE+eDI] ; ACCESS STATE
2195 00003DF8 F6C410             <1>       TEST   AH,MED_DET          ; ESTABLISHED STATE ?
2196 00003DFB 757E               <1>       JNZ    short NO_DBL              ; IF ESTABLISHED THEN DOUBLE DONE
2197                             <1>
2198                             <1> ;-----      CHECK FOR TRACK 0 TO SPEED UP ACKNOWLEDGE OF UNFORMATTED DISKETTE
2199                             <1>
2200 00003DFD C605[85520100]00   <1>       MOV    byte [SEEK_STATUS],0      ; SET RECALIBRATE REQUIRED ON ALL DRIVES
2201 00003E04 E8E0000000         <1>       CALL   MOTOR_ON            ; ENSURE MOTOR STAY ON
2202 00003E09 B500               <1>       MOV    CH,0                ; LOAD TRACK 0
2203 00003E0B E8D4010000         <1>       CALL   SEEK                ; SEEK TO TRACK 0
2204 00003E10 E868000000         <1>       CALL   READ_ID                   ; READ ID FUNCTION
2205 00003E15 7249               <1>       JC     short SD_ERR        ; IF ERROR NO TRACK 0
2206                             <1>
2207                             <1> ;-----      INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)
2208                             <1>
2209 00003E17 66B95004           <1>       MOV    CX,0450H            ; START, MAX TRACKS
2210 00003E1B F687[95520100]01   <1>       TEST   byte [DSK_STATE+eDI],TRK_CAPA ; TEST FOR 80 TRACK CAPABILITY
2211 00003E22 7402               <1>       JZ     short CNT_OK        ; IF NOT COUNT IS SETUP
2212 00003E24 B1A0               <1>       MOV    CL,0A0H                   ; MAXIMUM TRACK 1.2 MB
2213                             <1>
2214                             <1> ;     ATTEMPT READ ID OF ALL TRACKS, ALL HEADS UNTIL SUCCESS; UPON SUCCESS,
2215                             <1> ;     MUST SEE IF ASKED FOR TRACK IN SINGLE STEP MODE = TRACK ID READ; IF NOT
2216                             <1> ;     THEN SET DOUBLE STEP ON.
2217                             <1>
2218                             <1> CNT_OK:
2219 00003E26 C605[87520100]FF   <1>         MOV     byte [MOTOR_COUNT], 0FFH ; ENSURE MOTOR STAYS ON FOR OPERATION
2220 00003E2D 6651               <1>       PUSH   CX                  ; SAVE TRACK, COUNT
2221 00003E2F C605[88520100]00   <1>       MOV    byte [DSKETTE_STATUS],0   ; CLEAR STATUS, EXPECT ERRORS
2222 00003E36 6631C0             <1>       XOR    AX,AX               ; CLEAR AX
2223 00003E39 D0ED               <1>       SHR    CH,1                ; HALVE TRACK, CY = HEAD
2224 00003E3B C0D003             <1>       RCL    AL,3                ; AX = HEAD IN CORRECT BIT
2225 00003E3E 6650               <1>       PUSH   AX                  ; SAVE HEAD
2226 00003E40 E89F010000         <1>       CALL   SEEK                ; SEEK TO TRACK
2227 00003E45 6658               <1>       POP    AX                  ; RESTORE HEAD
2228 00003E47 6609C7             <1>       OR     DI,AX               ; DI = HEAD OR'ED DRIVE
2229 00003E4A E82E000000         <1>       CALL   READ_ID                   ; READ ID HEAD 0
2230 00003E4F 9C                 <1>       PUSHF                      ; SAVE RETURN FROM READ_ID
2231 00003E50 6681E7FB00         <1>       AND    DI,11111011B        ; TURN OFF HEAD 1 BIT
2232 00003E55 9D                 <1>       POPF                       ; RESTORE ERROR RETURN
2233 00003E56 6659               <1>       POP    CX                  ; RESTORE COUNT
2234 00003E58 7308               <1>       JNC    short DO_CHK        ; IF OK, ASKED = RETURNED TRACK ?
2235 00003E5A FEC5               <1>       INC    CH                  ; INC FOR NEXT TRACK
2236 00003E5C 38CD               <1>       CMP    CH,CL               ; REACHED MAXIMUM YET
2237 00003E5E 75C6               <1>       JNZ    short CNT_OK        ; CONTINUE TILL ALL TRIED
2238                             <1>
2239                             <1> ;-----      FALL THRU, READ ID FAILED FOR ALL TRACKS
2240                             <1>
2241                             <1> SD_ERR:
2242 00003E60 F9                 <1>       STC                        ; SET CARRY FOR ERROR
2243 00003E61 C3                 <1>       RETn                       ; SETUP_DBL ERROR EXIT
2244                             <1>
2245                             <1> DO_CHK:
2246 00003E62 8A0D[8C520100]     <1>       MOV    CL, [NEC_STATUS+3]  ; LOAD RETURNED TRACK
2247 00003E68 888F[99520100]     <1>       MOV    [DSK_TRK+eDI], CL   ; STORE TRACK NUMBER
2248 00003E6E D0ED               <1>       SHR    CH,1                ; HALVE TRACK
2249 00003E70 38CD               <1>       CMP    CH,CL               ; IS IT THE SAME AS ASKED FOR TRACK
2250 00003E72 7407               <1>       JZ     short NO_DBL        ; IF SAME THEN NO DOUBLE STEP
2251 00003E74 808F[95520100]20   <1>       OR     byte [DSK_STATE+eDI],DBL_STEP ; TURN ON DOUBLE STEP REQUIRED
2252                             <1> NO_DBL:
2253 00003E7B F8                 <1>       CLC                        ; CLEAR ERROR FLAG
2254 00003E7C C3                 <1>       RETn
2255                             <1>
2256                             <1> ;--------------------------------------------------------------------------
2257                             <1> ; READ_ID
2258                             <1> ;     READ ID FUNCTION.
2259                             <1> ;
```

```
                        <1> ; ON ENTRY:  DI : BIT 2 = HEAD; BITS 1,0 = DRIVE
                        <1> ;
                        <1> ; ON EXIT:   DI : BIT 2 IS RESET, BITS 1,0 = DRIVE
                        <1> ;            @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                        <1> ;--------------------------------------------------------------------------
                        <1> READ_ID:
00003E7D B8[9A3E0000]   <1>       MOV    eAX, ER_3           ; MOVE NEC OUTPUT ERROR ADDRESS
00003E82 50             <1>       PUSH   eAX
00003E83 B44A           <1>       MOV    AH,4AH              ; READ ID COMMAND
00003E85 E820010000     <1>       CALL   NEC_OUTPUT          ; TO CONTROLLER
00003E8A 6689F8         <1>       MOV    AX,DI               ; DRIVE # TO AH, HEAD 0
00003E8D 88C4           <1>       MOV    AH,AL
00003E8F E816010000     <1>       CALL   NEC_OUTPUT          ; TO CONTROLLER
00003E94 E80BFEFFFF     <1>       CALL   NEC_TERM            ; WAIT FOR OPERATION, GET STATUS
00003E99 58             <1>       POP    eAX                 ; THROW AWAY ERROR ADDRESS
                        <1> ER_3:
00003E9A C3             <1>       RETn
                        <1>
                        <1> ;--------------------------------------------------------------------------
                        <1> ; CMOS_TYPE
                        <1> ;     RETURNS DISKETTE TYPE FROM CMOS
                        <1> ;
                        <1> ; ON ENTRY:  DI = DRIVE #
                        <1> ;
                        <1> ; ON EXIT:   AL = TYPE; CY REFLECTS STATUS
                        <1> ;--------------------------------------------------------------------------
                        <1>
                        <1> CMOS_TYPE: ; 11/12/2014
00003E9B 8A87[F65C0000] <1> mov   al, [eDI+fd0_type]
00003EA1 20C0           <1> and   al, al ; 18/12/2014
00003EA3 C3             <1> retn
                        <1>
                        <1> ;CMOS_TYPE:
                        <1> ;     MOV    AL, CMOS_DIAG       ; CMOS DIAGNOSTIC STATUS BYTE ADDRESS
                        <1> ;     CALL   CMOS_READ           ; GET CMOS STATUS
                        <1> ;     TEST   AL,BAD_BAT+BAD_CKSUM      ; BATTERY GOOD AND CHECKSUM VALID
                        <1> ;     STC                        ; SET CY = 1 INDICATING ERROR FOR RETURN
                        <1> ;     JNZ    short BAD_CM        ; ERROR IF EITHER BIT ON
                        <1> ;     MOV    AL,CMOS_DISKETTE    ; ADDRESS OF DISKETTE BYTE IN CMOS
                        <1> ;     CALL   CMOS_READ           ; GET DISKETTE BYTE
                        <1> ;     OR     DI,DI               ; SEE WHICH DRIVE IN QUESTION
                        <1> ;     JNZ    short TB            ; IF DRIVE 1, DATA IN LOW NIBBLE
                        <1> ;     ROR    AL,4                ; EXCHANGE NIBBLES IF SECOND DRIVE
                        <1> ;TB:
                        <1> ;     AND    AL,0FH              ; KEEP ONLY DRIVE DATA, RESET CY, 0
                        <1> ;BAD_CM:
                        <1> ;     RETn                       ; CY, STATUS OF READ
                        <1>
                        <1> ;--------------------------------------------------------------------------
                        <1> ; GET_PARM
                        <1> ;     THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE DISK_BASE
                        <1> ;     BLOCK POINTED TO BY THE DATA VARIABLE @DISK_POINTER. A BYTE FROM
                        <1> ;     THAT TABLE IS THEN MOVED INTO AH, THE INDEX OF THAT BYTE BEING
                        <1> ;     THE PARAMETER IN DL.
                        <1> ;
                        <1> ; ON ENTRY:  DL = INDEX OF BYTE TO BE FETCHED
                        <1> ;
                        <1> ; ON EXIT:   AH = THAT BYTE FROM BLOCK
                        <1> ;            AL,DH DESTROYED
                        <1> ;--------------------------------------------------------------------------
                        <1> GET_PARM:
                        <1>       ;PUSH  DS
00003EA4 56             <1>       PUSH   eSI
                        <1>       ;SUB   AX,AX               ; DS = 0, BIOS DATA AREA
                        <1>       ;MOV   DS,AX
                        <1>       ;;mov  ax, cs
                        <1>       ;;mov  ds, ax
                        <1>       ; 08/02/2015 (protected mode modifications, bx -> ebx)
00003EA5 87D3           <1>       XCHG   eDX,eBX                    ; BL = INDEX
                        <1>       ;SUB   BH,BH               ; BX = INDEX
00003EA7 81E3FF000000   <1>       and    ebx, 0FFh
                        <1>       ;LDS   SI, [DISK_POINTER]  ; POINT TO BLOCK
                        <1>       ;
                        <1>       ; 17/12/2014
00003EAD 66A1[E55C0000] <1>       mov    ax, [cfd] ; current (AL) and previous fd (AH)
00003EB3 38E0           <1>       cmp    al, ah
00003EB5 7425           <1>       je     short gpndc
00003EB7 A2[E65C0000]   <1>       mov    [pfd], al ; current drive -> previous drive
00003EBC 53             <1>       push   ebx ; 08/02/2015
00003EBD 88C3           <1>       mov    bl, al
                        <1>       ; 11/12/2014
00003EBF 8A83[F65C0000] <1>       mov    al, [eBX+fd0_type]  ; Drive type (0,1,2,3,4)
                        <1>       ; 18/12/2014
00003EC5 20C0           <1>       and    al, al
00003EC7 7507           <1>       jnz    short gpdtc
00003EC9 BB[CF5C0000]   <1>       mov    ebx, MD_TBL6        ; 1.44 MB param. tbl. (default)
00003ECE EB05           <1>         jmp     short gpdpu
                        <1> gpdtc:
00003ED0 E817F9FFFF     <1>       call   DR_TYPE_CHECK
                        <1>       ; cf = 1 -> eBX points to 1.44MB fd parameter table (default)
                        <1> gpdpu:
00003ED5 891D[6C5C0000] <1>       mov    [DISK_POINTER], ebx
00003EDB 5B             <1>       pop    ebx
                        <1> gpndc:
00003EDC 8B35[6C5C0000] <1>       mov    esi, [DISK_POINTER] ; 08/02/2015, si -> esi
00003EE2 8A241E         <1>       MOV    AH, [eSI+eBX]       ; GET THE WORD
00003EE5 87D3           <1>       XCHG   eDX,eBX                   ; RESTORE BX
00003EE7 5E             <1>       POP    eSI
                        <1>       ;POP   DS
00003EE8 C3             <1>       RETn
                        <1>
                        <1> ;--------------------------------------------------------------------------
```

```
                        <1> ; MOTOR_ON
                        <1> ;     TURN MOTOR ON AND WAIT FOR MOTOR START UP TIME. THE @MOTOR_COUNT
                        <1> ;     IS REPLACED WITH A SUFFICIENTLY HIGH NUMBER (0FFH) TO ENSURE
                        <1> ;     THAT THE MOTOR DOES NOT GO OFF DURING THE OPERATION. IF THE
                        <1> ;     MOTOR NEEDED TO BE TURNED ON, THE MULTI-TASKING HOOK FUNCTION
                        <1> ;     (AX=90FDH, INT 15) IS CALLED TELLING THE OPERATING SYSTEM
                        <1> ;     THAT THE BIOS IS ABOUT TO WAIT FOR MOTOR START UP. IF THIS
                        <1> ;     FUNCTION RETURNS WITH CY = 1, IT MEANS THAT THE MINIMUM WAIT
                        <1> ;     HAS BEEN COMPLETED. AT THIS POINT A CHECK IS MADE TO ENSURE
                        <1> ;     THAT THE MOTOR WASN'T TURNED OFF BY THE TIMER. IF THE HOOK DID
                        <1> ;     NOT WAIT, THE WAIT FUNCTION (AH=086H) IS CALLED TO WAIT THE
                        <1> ;     PRESCRIBED AMOUNT OF TIME. IF THE CARRY FLAG IS SET ON RETURN,
                        <1> ;     IT MEANS THAT THE FUNCTION IS IN USE AND DID NOT PERFORM THE
                        <1> ;     WAIT. A TIMER 1 WAIT LOOP WILL THEN DO THE WAIT.
                        <1> ;
                        <1> ; ON ENTRY:  DI = DRIVE #
                        <1> ; ON EXIT:   AX,CX,DX DESTROYED
                        <1> ;--------------------------------------------------------------------------
                        <1> MOTOR_ON:
00003EE9 53             <1>       PUSH   eBX                ; SAVE REG.
00003EEA E82A000000     <1>       CALL   TURN_ON                  ; TURN ON MOTOR
00003EEF 7226           <1>       JC     short MOT_IS_ON          ; IF CY=1 NO WAIT
00003EF1 E89BF9FFFF     <1>       CALL   XLAT_OLD           ; TRANSLATE STATE TO COMPATIBLE MODE
00003EF6 E865F9FFFF     <1>       CALL   XLAT_NEW           ; TRANSLATE STATE TO PRESENT ARCH,
                        <1>       ;CALL  TURN_ON            ; CHECK AGAIN IF MOTOR ON
                        <1>       ;JC    MOT_IS_ON          ; IF NO WAIT MEANS IT IS ON
                        <1> M_WAIT:
00003EFB B20A           <1>       MOV    DL,10              ; GET THE MOTOR WAIT PARAMETER
00003EFD E8A2FFFFFF     <1>       CALL   GET_PARM
                        <1>       ;MOV   AL,AH              ; AL = MOTOR WAIT PARAMETER
                        <1>       ;XOR   AH,AH              ; AX = MOTOR WAIT PARAMETER
                        <1>       ;CMP   AL,8               ; SEE IF AT LEAST A SECOND IS SPECIFIED
00003F02 80FC08         <1>       cmp    ah, 8
                        <1>       ;JAE   short GP2          ; IF YES, CONTINUE
00003F05 7702           <1>       ja     short J13
                        <1>       ;MOV   AL,8               ; ONE SECOND WAIT FOR MOTOR START UP
00003F07 B408           <1>       mov    ah, 8
                        <1>
                        <1> ;-----      AS CONTAINS NUMBER OF 1/8 SECONDS (125000 MICROSECONDS) TO WAIT
                        <1> GP2:
                        <1> ;-----      FOLLOWING LOOPS REQUIRED WHEN RTC WAIT FUNCTION IS ALREADY IN USE
                        <1> J13:                          ; WAIT FOR 1/8 SECOND PER (AL)
00003F09 B95E200000     <1>       MOV    eCX,8286           ; COUNT FOR 1/8 SECOND AT 15.085737 US
00003F0E E8DADEFFFF     <1>       CALL   WAITF              ; GO TO FIXED WAIT ROUTINE
                        <1>       ;DEC   AL                 ; DECREMENT TIME VALUE
00003F13 FECC           <1>       dec    ah
00003F15 75F2           <1>       JNZ    short J13          ; ARE WE DONE YET
                        <1> MOT_IS_ON:
00003F17 5B             <1>       POP    eBX                ; RESTORE REG.
00003F18 C3             <1>       RETn
                        <1>
                        <1> ;--------------------------------------------------------------------------
                        <1> ; TURN_ON
                        <1> ;     TURN MOTOR ON AND RETURN WAIT STATE.
                        <1> ;
                        <1> ; ON ENTRY:  DI = DRIVE #
                        <1> ;
                        <1> ; ON EXIT:   CY = 0 MEANS WAIT REQUIRED
                        <1> ;            CY = 1 MEANS NO WAIT REQUIRED
                        <1> ;            AX,BX,CX,DX DESTROYED
                        <1> ;--------------------------------------------------------------------------
                        <1> TURN_ON:
00003F19 89FB           <1>       MOV    eBX,eDI                  ; BX = DRIVE #
00003F1B 88D9           <1>       MOV    CL,BL              ; CL = DRIVE #
00003F1D C0C304         <1>       ROL    BL,4               ; BL = DRIVE SELECT
00003F20 FA             <1>       CLI                       ; NO INTERRUPTS WHILE DETERMINING STATUS
00003F21 C605[87520100]FF <1>     MOV    byte [MOTOR_COUNT],0FFH   ; ENSURE MOTOR STAYS ON FOR OPERATION
00003F28 A0[86520100]   <1>       MOV    AL, [MOTOR_STATUS] ; GET DIGITAL OUTPUT REGISTER REFLECTION
00003F2D 2430           <1>       AND    AL,00110000B       ; KEEP ONLY DRIVE SELECT BITS
00003F2F B401           <1>       MOV    AH,1               ; MASK FOR DETERMINING MOTOR BIT
00003F31 D2E4           <1>       SHL    AH,CL              ; AH = MOTOR ON, A=00000001, B=00000010
                        <1>
                        <1> ;  AL = DRIVE SELECT FROM @MOTOR_STATUS
                        <1> ;  BL = DRIVE SELECT DESIRED
                        <1> ;  AH = MOTOR ON MASK DESIRED
                        <1>
00003F33 38D8           <1>       CMP    AL,BL              ; REQUESTED DRIVE ALREADY SELECTED ?
00003F35 7508           <1>       JNZ    short TURN_IT_ON   ; IF NOT SELECTED JUMP
00003F37 8425[86520100] <1>       TEST   AH, [MOTOR_STATUS] ; TEST MOTOR ON BIT
00003F3D 7535           <1>       JNZ    short NO_MOT_WAIT  ; JUMP IF MOTOR ON AND SELECTED
                        <1>
                        <1> TURN_IT_ON:
00003F3F 08DC           <1>       OR     AH,BL              ; AH = DRIVE SELECT AND MOTOR ON
00003F41 8A3D[86520100] <1>       MOV    BH,[MOTOR_STATUS]  ; SAVE COPY OF @MOTOR_STATUS BEFORE
00003F47 80E70F         <1>       AND    BH,00001111B       ; KEEP ONLY MOTOR BITS
00003F4A 8025[86520100]CF <1>     AND    byte [MOTOR_STATUS],11001111B ; CLEAR OUT DRIVE SELECT
00003F51 0825[86520100] <1>       OR     [MOTOR_STATUS],AH  ; OR IN DRIVE SELECTED AND MOTOR ON
00003F57 A0[86520100]   <1>       MOV    AL,[MOTOR_STATUS]  ; GET DIGITAL OUTPUT REGISTER REFLECTION
00003F5C 88C3           <1>       MOV    BL,AL              ; BL=@MOTOR_STATUS AFTER, BH=BEFORE
00003F5E 80E30F         <1>       AND    BL,00001111B       ; KEEP ONLY MOTOR BITS
00003F61 FB             <1>       STI                       ; ENABLE INTERRUPTS AGAIN
00003F62 243F           <1>       AND    AL,00111111B       ; STRIP AWAY UNWANTED BITS
00003F64 C0C004         <1>       ROL    AL,4               ; PUT BITS IN DESIRED POSITIONS
00003F67 0C0C           <1>       OR     AL,00001100B       ; NO RESET, ENABLE DMA/INTERRUPT
00003F69 66BAF203       <1>       MOV    DX,03F2H           ; SELECT DRIVE AND TURN ON MOTOR
00003F6D EE             <1>       OUT    DX,AL
00003F6E 38FB           <1>       CMP    BL,BH              ; NEW MOTOR TURNED ON ?
                        <1>       ;JZ    short NO_MOT_WAIT  ; NO WAIT REQUIRED IF JUST SELECT
00003F70 7403           <1>       je     short no_mot_w1 ; 27/02/2015
00003F72 F8             <1>       CLC                       ; (re)SET CARRY MEANING WAIT
00003F73 C3             <1>       RETn
                        <1>
```

```
2464                                    <1> NO_MOT_WAIT:
2465 00003F74 FB                       <1>         sti
2466                                    <1> no_mot_w1: ; 27/02/2015
2467 00003F75 F9                       <1>         STC                         ; SET NO WAIT REQUIRED
2468                                    <1>         ;STI                        ; INTERRUPTS BACK ON
2469 00003F76 C3                       <1>         RETn
2470                                    <1>
2471                                    <1> ;--------------------------------------------------------------------------------
2472                                    <1> ; HD_WAIT
2473                                    <1> ;     WAIT FOR HEAD SETTLE TIME.
2474                                    <1> ;
2475                                    <1> ; ON ENTRY:  DI = DRIVE #
2476                                    <1> ;
2477                                    <1> ; ON EXIT:   AX,BX,CX,DX DESTROYED
2478                                    <1> ;--------------------------------------------------------------------------------
2479                                    <1> HD_WAIT:
2480 00003F77 B209                     <1>         MOV    DL,9                 ; GET HEAD SETTLE PARAMETER
2481 00003F79 E826FFFFFF               <1>         CALL   GET_PARM
2482 00003F7E 08E4                     <1>         or     ah, ah ; 17/12/2014
2483 00003F80 7519                     <1>         jnz    short DO_WAT
2484 00003F82 F605[86520100]80         <1>           TEST    byte [MOTOR_STATUS],10000000B ; SEE IF A WRITE OPERATION
2485                                    <1>         ;JZ    short ISNT_WRITE     ; IF NOT, DO NOT ENFORCE ANY VALUES
2486                                    <1>         ;OR    AH,AH                ; CHECK FOR ANY WAIT?
2487                                    <1>         ;JNZ   short DO_WAT         ; IF THERE DO NOT ENFORCE
2488 00003F89 741E                     <1>         jz     short HW_DONE
2489 00003F8B B40F                     <1>         MOV    AH,HD12_SETTLE             ; LOAD 1.2M HEAD SETTLE MINIMUM
2490 00003F8D 8A87[95520100]           <1>         MOV    AL,[DSK_STATE+eDI]  ; LOAD STATE
2491 00003F93 24C0                     <1>         AND    AL,RATE_MSK         ; KEEP ONLY RATE
2492 00003F95 3C80                     <1>         CMP    AL,RATE_250         ; 1.2 M DRIVE ?
2493 00003F97 7502                     <1>         JNZ    short DO_WAT        ; DEFAULT HEAD SETTLE LOADED
2494                                    <1> ;GP3:
2495 00003F99 B414                     <1>         MOV    AH,HD320_SETTLE            ; USE 320/360 HEAD SETTLE
2496                                    <1> ;       JMP    SHORT DO_WAT
2497                                    <1>
2498                                    <1> ;ISNT_WRITE:
2499                                    <1> ;       OR     AH,AH               ; CHECK FOR NO WAIT
2500                                    <1> ;       JZ     short HW_DONE       ; IF NOT WRITE AND 0 ITS OK
2501                                    <1>
2502                                    <1> ;-----      AH CONTAINS NUMBER OF MILLISECONDS TO WAIT
2503                                    <1> DO_WAT:
2504                                    <1> ;       MOV    AL,AH               ; AL = # MILLISECONDS
2505                                    <1> ;       ;XOR   AH,AH               ; AX = # MILLISECONDS
2506                                    <1> J29:                               ;      1 MILLISECOND LOOP
2507                                    <1>         ;mov   cx, WAIT_FDU_HEAD_SETTLE ; 33 ; 1 ms in 30 micro units.
2508 00003F9B B942000000               <1>         MOV    eCX,66              ; COUNT AT 15.085737 US PER COUNT
2509 00003FA0 E848DEFFFF               <1>         CALL   WAITF               ; DELAY FOR 1 MILLISECOND
2510                                    <1>         ;DEC   AL                  ; DECREMENT THE COUNT
2511 00003FA5 FECC                     <1>         dec    ah
2512 00003FA7 75F2                     <1>         JNZ    short J29           ; DO AL MILLISECOND # OF TIMES
2513                                    <1> HW_DONE:
2514 00003FA9 C3                       <1>         RETn
2515                                    <1>
2516                                    <1> ;--------------------------------------------------------------------------------
2517                                    <1> ; NEC_OUTPUT
2518                                    <1> ;     THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER TESTING
2519                                    <1> ;     FOR CORRECT DIRECTION AND CONTROLLER READY THIS ROUTINE WILL
2520                                    <1> ;     TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN A REASONABLE AMOUNT
2521                                    <1> ;     OF TIME, SETTING THE DISKETTE STATUS ON COMPLETION.
2522                                    <1> ;
2523                                    <1> ; ON ENTRY:  AH = BYTE TO BE OUTPUT
2524                                    <1> ;
2525                                    <1> ; ON EXIT:   CY = 0  SUCCESS
2526                                    <1> ;            CY = 1  FAILURE -- DISKETTE STATUS UPDATED
2527                                    <1> ;                    IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL
2528                                    <1> ;                    HIGHER THAN THE CALLER OF NEC OUTPUT. THIS REMOVES THE
2529                                    <1> ;                    REQUIREMENT OF TESTING AFTER EVERY CALL OF NEC_OUTPUT.
2530                                    <1> ;            AX,CX,DX DESTROYED
2531                                    <1> ;--------------------------------------------------------------------------------
2532                                    <1>
2533                                    <1> ; 09/12/2014 [Erdogan Tan]
2534                                    <1> ;     (from 'PS2 Hardware Interface Tech. Ref. May 88', Page 09-05.)
2535                                    <1> ; Diskette Drive Controller Status Register (3F4h)
2536                                    <1> ;     This read only register facilitates the transfer of data between
2537                                    <1> ;     the system microprocessor and the controller.
2538                                    <1> ; Bit 7 - When set to 1, the Data register is ready to transfer data
2539                                    <1> ;       with the system micrprocessor.
2540                                    <1> ; Bit 6 - The direction of data transfer. If this bit is set to 0,
2541                                    <1> ;       the transfer is to the controller.
2542                                    <1> ; Bit 5 - When this bit is set to 1, the controller is in the non-DMA mode.
2543                                    <1> ; Bit 4 - When this bit is set to 1, a Read or Write command is being executed.
2544                                    <1> ; Bit 3 - Reserved.
2545                                    <1> ; Bit 2 - Reserved.
2546                                    <1> ; Bit 1 - When this bit is set to 1, dskette drive 1 is in the seek mode.
2547                                    <1> ; Bit 0 - When this bit is set to 1, dskette drive 1 is in the seek mode.
2548                                    <1>
2549                                    <1> ; Data Register (3F5h)
2550                                    <1> ; This read/write register passes data, commands and parameters, and provides
2551                                    <1> ; diskette status information.
2552                                    <1>
2553                                    <1> NEC_OUTPUT:
2554                                    <1>         ;PUSH  BX                  ; SAVE REG.
2555 00003FAA 66BAF403                 <1>         MOV    DX,03F4H            ; STATUS PORT
2556                                    <1>         ;MOV   BL,2                ; HIGH ORDER COUNTER
2557                                    <1>         ;XOR   CX,CX               ; COUNT FOR TIME OUT
2558                                    <1>         ; 16/12/2014
2559                                    <1>         ; waiting for (max.) 0.5 seconds
2560                                    <1>           ;;mov     byte [wait_count], 0 ;; 27/02/2015
2561                                    <1>         ;
2562                                    <1>         ; 17/12/2014
2563                                    <1>         ; Modified from AWARD BIOS 1999 - ADISK.ASM - SEND_COMMAND
2564                                    <1>         ;
2565                                    <1>         ;WAIT_FOR_PORT:    Waits for a bit at a port pointed to by DX to
```

```
2566                            <1>       ;            go on.
2567                            <1>       ;INPUT:
2568                            <1>       ;      AH=Mask for isolation bits.
2569                            <1>       ;      AL=pattern to look for.
2570                            <1>       ;      DX=Port to test for
2571                            <1>       ;      BH:CX=Number of memory refresh periods to delay.
2572                            <1>       ;           (normally 30 microseconds per period.)
2573                            <1>       ;
2574                            <1>       ;WFP_SHORT:
2575                            <1>       ;      Wait for port if refresh cycle is short (15-80 Us range).
2576                            <1>       ;
2577                            <1>
2578                            <1> ;     mov    bl, WAIT_FDU_SEND_HI+1    ; 0+1
2579                            <1> ;     mov    cx, WAIT_FDU_SEND_LO      ; 16667
2580 00003FAE B91B410000       <1>       mov    ecx, WAIT_FDU_SEND_LH   ; 16667 (27/02/2015)
2581                            <1> ;
2582                            <1> ;WFPS_OUTER_LP:
2583                            <1> ;     ;
2584                            <1> ;WFPS_CHECK_PORT:
2585                            <1> J23:
2586 00003FB3 EC               <1>       IN     AL,DX              ; GET STATUS
2587 00003FB4 24C0             <1>       AND    AL,11000000B       ; KEEP STATUS AND DIRECTION
2588 00003FB6 3C80             <1>       CMP    AL,10000000B       ; STATUS 1 AND DIRECTION 0 ?
2589 00003FB8 7418             <1>       JZ     short J27          ; STATUS AND DIRECTION OK
2590                            <1> WFPS_HI:
2591 00003FBA E461             <1>       IN     AL, PORT_B   ;061h  ; SYS1 ; wait for hi to lo
2592 00003FBC A810             <1>       TEST   AL,010H                   ; transition on memory
2593 00003FBE 75FA             <1>       JNZ    SHORT WFPS_HI       ; refresh.
2594                            <1> WFPS_LO:
2595 00003FC0 E461             <1>       IN     AL, PORT_B          ; SYS1
2596 00003FC2 A810             <1>       TEST   AL,010H
2597 00003FC4 74FA             <1>       JZ     SHORT WFPS_LO
2598                            <1>       ;LOOP  SHORT WFPS_CHECK_PORT
2599 00003FC6 E2EB             <1>       loop   J23    ; 27/02/2015
2600                            <1> ;     ;
2601                            <1> ;     dec    bl
2602                            <1> ;     jnz    short WFPS_OUTER_LP
2603                            <1> ;     jmp    short WFPS_TIMEOUT  ; fail
2604                            <1> ;J23:
2605                            <1> ;     IN     AL,DX              ; GET STATUS
2606                            <1> ;     AND    AL,11000000B       ; KEEP STATUS AND DIRECTION
2607                            <1> ;     CMP    AL,10000000B       ; STATUS 1 AND DIRECTION 0 ?
2608                            <1> ;     JZ     short J27          ; STATUS AND DIRECTION OK
2609                            <1>       ;LOOP  J23                ; CONTINUE TILL CX EXHAUSTED
2610                            <1>       ;DEC   BL                 ; DECREMENT COUNTER
2611                            <1>       ;JNZ   short J23          ; REPEAT TILL DELAY FINISHED, CX = 0
2612                            <1>
2613                            <1>       ;;27/02/2015
2614                            <1>       ;16/12/2014
2615                            <1>         ;;cmp     byte [wait_count], 10   ; (10/18.2 seconds)
2616                            <1>       ;;jb   short J23
2617                            <1>
2618                            <1> ;WFPS_TIMEOUT:
2619                            <1>
2620                            <1> ;-----      FALL THRU TO ERROR RETURN
2621                            <1>
2622 00003FC8 800D[88520100]80 <1>       OR     byte [DSKETTE_STATUS],TIME_OUT
2623                            <1>       ;POP   BX                 ; RESTORE REG.
2624 00003FCF 58               <1>       POP    eAX ; 08/02/2015   ; DISCARD THE RETURN ADDRESS
2625 00003FD0 F9               <1>       STC                       ; INDICATE ERROR TO CALLER
2626 00003FD1 C3               <1>       RETn
2627                            <1>
2628                            <1> ;-----      DIRECTION AND STATUS OK; OUTPUT BYTE
2629                            <1>
2630                            <1> J27:
2631 00003FD2 88E0             <1>       MOV    AL,AH              ; GET BYTE TO OUTPUT
2632 00003FD4 6642             <1>       INC    DX                 ; DATA PORT = STATUS PORT + 1
2633 00003FD6 EE               <1>       OUT    DX,AL              ; OUTPUT THE BYTE
2634                            <1>       ;;NEWIODELAY  ;; 27/02/2015
2635                            <1>       ; 27/02/2015
2636 00003FD7 9C               <1>       PUSHF                     ; SAVE FLAGS
2637 00003FD8 B903000000       <1>       MOV    eCX, 3             ; 30 TO 45 MICROSECONDS WAIT FOR
2638 00003FDD E80BDEFFFF       <1>       CALL   WAITF              ; NEC FLAGS UPDATE CYCLE
2639 00003FE2 9D               <1>       POPF                      ; RESTORE FLAGS FOR EXIT
2640                            <1>       ;POP   BX                 ; RESTORE REG
2641 00003FE3 C3               <1>       RETn                      ; CY = 0 FROM TEST INSTRUCTION
2642                            <1>
2643                            <1> ;--------------------------------------------------------------------------
2644                            <1> ; SEEK
2645                            <1> ;     THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE TO THE NAMED
2646                            <1> ;     TRACK. IF THE DRIVE HAS NOT BEEN ACCESSED SINCE THE DRIVE
2647                            <1> ;     RESET COMMAND WAS ISSUED, THE DRIVE WILL BE RECALIBRATED.
2648                            <1> ;
2649                            <1> ; ON ENTRY:  DI = DRIVE #
2650                            <1> ;            CH = TRACK #
2651                            <1> ;
2652                            <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
2653                            <1> ;            AX,BX,CX DX DESTROYED
2654                            <1> ;--------------------------------------------------------------------------
2655                            <1> SEEK:
2656 00003FE4 89FB             <1>       MOV    eBX,eDI                  ; BX = DRIVE #
2657 00003FE6 B001             <1>       MOV    AL,1               ; ESTABLISH MASK FOR RECALIBRATE TEST
2658 00003FE8 86CB             <1>       XCHG   CL,BL              ; SET DRIVE VALULE INTO CL
2659 00003FEA D2C0             <1>       ROL    AL,CL              ; SHIFT MASK BY THE DRIVE VALUE
2660 00003FEC 86CB             <1>       XCHG   CL,BL              ; RECOVER TRACK VALUE
2661 00003FEE 8405[85520100]   <1>       TEST   AL,[SEEK_STATUS]   ; TEST FOR RECALIBRATE REQUIRED
2662 00003FF4 7526             <1>       JNZ    short J28A         ; JUMP IF RECALIBRATE NOT REQUIRED
2663                            <1>
2664 00003FF6 0805[85520100]   <1>       OR     [SEEK_STATUS],AL   ; TURN ON THE NO RECALIBRATE BIT IN FLAG
2665 00003FFC E862000000       <1>       CALL   RECAL              ; RECALIBRATE DRIVE
2666 00004001 730E             <1>       JNC    short AFT_RECAL          ; RECALIBRATE DONE
2667                            <1>
```

```
2668                                    <1> ;-----      ISSUE RECALIBRATE FOR 80 TRACK DISKETTES
2669                                    <1>
2670 00004003 C605[88520100]00          <1>       MOV   byte [DSKETTE_STATUS],0   ; CLEAR OUT INVALID STATUS
2671 0000400A E854000000                <1>       CALL  RECAL               ; RECALIBRATE DRIVE
2672 0000400F 7251                      <1>       JC    short RB            ; IF RECALIBRATE FAILS TWICE THEN ERROR
2673                                    <1>
2674                                    <1> AFT_RECAL:
2675 00004011 C687[99520100]00          <1>         MOV    byte [DSK_TRK+eDI],0    ; SAVE NEW CYLINDER AS PRESENT POSITION
2676 00004018 08ED                      <1>       OR    CH,CH               ; CHECK FOR SEEK TO TRACK 0
2677 0000401A 743F                      <1>       JZ    short DO_WAIT       ; HEAD SETTLE, CY = 0 IF JUMP
2678                                    <1>
2679                                    <1> ;-----      DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK
2680                                    <1>
2681 0000401C F687[95520100]20          <1> J28A: TEST  byte [DSK_STATE+eDI],DBL_STEP ; CHECK FOR DOUBLE STEP REQUIRED
2682 00004023 7402                      <1>       JZ    short _R7           ; SINGLE STEP REQUIRED BYPASS DOUBLE
2683 00004025 D0E5                      <1>       SHL   CH,1                ; DOUBLE NUMBER OF STEP TO TAKE
2684                                    <1>
2685 00004027 3AAF[99520100]            <1> _R7:  CMP   CH, [DSK_TRK+eDI]   ; SEE IF ALREADY AT THE DESIRED TRACK
2686 0000402D 7433                      <1>       JE    short RB            ; IF YES, DO NOT NEED TO SEEK
2687                                    <1>
2688 0000402F BA[62400000]              <1>       MOV   eDX, NEC_ERR        ; LOAD RETURN ADDRESS
2689 00004034 52                        <1>       PUSH  eDX ; (*)           ; ON STACK FOR NEC OUTPUT ERROR
2690 00004035 88AF[99520100]            <1>       MOV   [DSK_TRK+eDI],CH    ; SAVE NEW CYLINDER AS PRESENT POSITION
2691 0000403B B40F                      <1>       MOV   AH,0FH              ; SEEK COMMAND TO NEC
2692 0000403D E868FFFFFF                <1>       CALL  NEC_OUTPUT
2693 00004042 89FB                      <1>       MOV   eBX,eDI                   ; BX = DRIVE #
2694 00004044 88DC                      <1>       MOV   AH,BL               ; OUTPUT DRIVE NUMBER
2695 00004046 E85FFFFFFF                <1>       CALL  NEC_OUTPUT
2696 0000404B 8AA7[99520100]            <1>       MOV   AH, [DSK_TRK+eDI]   ; GET CYLINDER NUMBER
2697 00004051 E854FFFFFF                <1>       CALL  NEC_OUTPUT
2698 00004056 E829000000                <1>       CALL  CHK_STAT_2          ; ENDING INTERRUPT AND SENSE STATUS
2699                                    <1>
2700                                    <1> ;-----      WAIT FOR HEAD SETTLE
2701                                    <1>
2702                                    <1> DO_WAIT:
2703 0000405B 9C                        <1>       PUSHF                     ; SAVE STATUS
2704 0000405C E816FFFFFF                <1>       CALL  HD_WAIT                   ; WAIT FOR HEAD SETTLE TIME
2705 00004061 9D                        <1>       POPF                      ; RESTORE STATUS
2706                                    <1> RB:
2707                                    <1> NEC_ERR:
2708                                    <1>       ; 08/02/2015 (code trick here from original IBM PC/AT DISKETTE.ASM)
2709                                    <1>       ; (*) nec_err -> retn (push edx -> pop edx) -> nec_err -> retn
2710 00004062 C3                        <1>       RETn                      ; RETURN TO CALLER
2711                                    <1>
2712                                    <1> ;--------------------------------------------------------------------------------
2713                                    <1> ; RECAL
2714                                    <1> ;     RECALIBRATE DRIVE
2715                                    <1> ;
2716                                    <1> ; ON ENTRY:  DI = DRIVE #
2717                                    <1> ;
2718                                    <1> ; ON EXIT:   CY REFLECTS STATUS OF OPERATION.
2719                                    <1> ;--------------------------------------------------------------------------------
2720                                    <1> RECAL:
2721 00004063 6651                      <1>       PUSH  CX
2722 00004065 B8[81400000]              <1>       MOV   eAX, RC_BACK        ; LOAD NEC_OUTPUT ERROR
2723 0000406A 50                        <1>       PUSH  eAX
2724 0000406B B407                      <1>       MOV   AH,07H              ; RECALIBRATE COMMAND
2725 0000406D E838FFFFFF                <1>       CALL  NEC_OUTPUT
2726 00004072 89FB                      <1>       MOV   eBX,eDI                   ; BX = DRIVE #
2727 00004074 88DC                      <1>       MOV   AH,BL
2728 00004076 E82FFFFFFF                <1>       CALL  NEC_OUTPUT          ; OUTPUT THE DRIVE NUMBER
2729 0000407B E804000000                <1>       CALL  CHK_STAT_2          ; GET THE INTERRUPT AND SENSE INT STATUS
2730 00004080 58                        <1>       POP   eAX                 ; THROW AWAY ERROR
2731                                    <1> RC_BACK:
2732 00004081 6659                      <1>       POP   CX
2733 00004083 C3                        <1>       RETn
2734                                    <1>
2735                                    <1> ;--------------------------------------------------------------------------------
2736                                    <1> ; CHK_STAT_2
2737                                    <1> ;     THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER RECALIBRATE,
2738                                    <1> ;     OR SEEK TO THE ADAPTER. THE INTERRUPT IS WAITED FOR, THE
2739                                    <1> ;     INTERRUPT STATUS SENSED, AND THE RESULT RETURNED TO THE CALLER.
2740                                    <1> ;
2741                                    <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
2742                                    <1> ;--------------------------------------------------------------------------------
2743                                    <1> CHK_STAT_2:
2744 00004084 B8[AC400000]              <1>         MOV    eAX, CS_BACK            ; LOAD NEC_OUTPUT ERROR ADDRESS
2745 00004089 50                        <1>       PUSH  eAX
2746 0000408A E828000000                <1>       CALL  WAIT_INT            ; WAIT FOR THE INTERRUPT
2747 0000408F 721A                      <1>       JC    short J34           ; IF ERROR, RETURN IT
2748 00004091 B408                      <1>       MOV   AH,08H              ; SENSE INTERRUPT STATUS COMMAND
2749 00004093 E812FFFFFF                <1>       CALL  NEC_OUTPUT
2750 00004098 E84A000000                <1>       CALL  RESULTS                   ; READ IN THE RESULTS
2751 0000409D 720C                      <1>       JC    short J34
2752 0000409F A0[89520100]              <1>       MOV   AL,[NEC_STATUS]           ; GET THE FIRST STATUS BYTE
2753 000040A4 2460                      <1>       AND   AL,01100000B        ; ISOLATE THE BITS
2754 000040A6 3C60                      <1>       CMP   AL,01100000B        ; TEST FOR CORRECT VALUE
2755 000040A8 7403                      <1>       JZ    short J35           ; IF ERROR, GO MARK IT
2756 000040AA F8                        <1>       CLC                       ; GOOD RETURN
2757                                    <1> J34:
2758 000040AB 58                        <1>       POP   eAX                 ; THROW AWAY ERROR RETURN
2759                                    <1> CS_BACK:
2760 000040AC C3                        <1>       RETn
2761                                    <1> J35:
2762 000040AD 800D[88520100]40          <1>       OR    byte [DSKETTE_STATUS], BAD_SEEK
2763 000040B4 F9                        <1>       STC                       ; ERROR RETURN CODE
2764 000040B5 EBF4                      <1>       JMP   SHORT J34
2765                                    <1>
2766                                    <1> ;--------------------------------------------------------------------------------
2767                                    <1> ; WAIT_INT
2768                                    <1> ;     THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR A TIME OUT ROUTINE
2769                                    <1> ;     TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE RETURNED
```

```
2770                            <1> ;     IF THE DRIVE IS NOT READY.
2771                            <1> ;
2772                            <1> ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
2773                            <1> ;--------------------------------------------------------------------------------
2774                            <1>
2775                            <1> ; 17/12/2014
2776                            <1> ; 2.5 seconds waiting !
2777                            <1> ;(AWARD BIOS - 1999, WAIT_FDU_INT_LOW, WAIT_FDU_INT_HI)
2778                            <1> ; amount of time to wait for completion interrupt from NEC.
2779                            <1>
2780                            <1>
2781                            <1> WAIT_INT:
2782 000040B7 FB               <1>       STI                     ; TURN ON INTERRUPTS, JUST IN CASE
2783 000040B8 F8               <1>       CLC                     ; CLEAR TIMEOUT INDICATOR
2784                            <1>        ;MOV  BL,10            ; CLEAR THE COUNTERS
2785                            <1>        ;XOR  CX,CX            ; FOR 2 SECOND WAIT
2786                            <1>
2787                            <1>       ; Modification from AWARD BIOS - 1999 (ATORGS.ASM, WAIT
2788                            <1>       ;
2789                            <1>       ;WAIT_FOR_MEM:
2790                            <1>       ;     Waits for a bit at a specified memory location pointed
2791                            <1>       ;     to by ES:[DI] to become set.
2792                            <1>       ;INPUT:
2793                            <1>       ;     AH=Mask to test with.
2794                            <1>       ;     ES:[DI] = memory location to watch.
2795                            <1>       ;     BH:CX=Number of memory refresh periods to delay.
2796                            <1>       ;          (normally 30 microseconds per period.)
2797                            <1>
2798                            <1>       ; waiting for (max.) 2.5 secs in 30 micro units.
2799                            <1> ;     mov    cx, WAIT_FDU_INT_LO       ; 017798
2800                            <1> ;;    mov    bl, WAIT_FDU_INT_HI
2801                            <1> ;     mov    bl, WAIT_FDU_INT_HI + 1
2802                            <1>       ; 27/02/2015
2803 000040B9 B986450100       <1>       mov    ecx, WAIT_FDU_INT_LH      ; 83334 (2.5 seconds)
2804                            <1> WFMS_CHECK_MEM:
2805 000040BE F605[85520100]80 <1>       test   byte [SEEK_STATUS],INT_FLAG ; TEST FOR INTERRUPT OCCURRING
2806 000040C5 7516             <1>         jnz     short J37
2807                            <1> WFMS_HI:
2808 000040C7 E461             <1>       IN     AL,PORT_B  ; 061h   ; SYS1, wait for lo to hi
2809 000040C9 A810             <1>       TEST   AL,010H                   ; transition on memory
2810 000040CB 75FA             <1>       JNZ    SHORT WFMS_HI       ; refresh.
2811                            <1> WFMS_LO:
2812 000040CD E461             <1>       IN     AL,PORT_B           ;SYS1
2813 000040CF A810             <1>       TEST   AL,010H
2814 000040D1 74FA             <1>       JZ     SHORT WFMS_LO
2815 000040D3 E2E9             <1>         LOOP    WFMS_CHECK_MEM
2816                            <1> ;WFMS_OUTER_LP:
2817                            <1> ;;    or     bl, bl              ; check outer counter
2818                            <1> ;;    jz     short J36A          ; WFMS_TIMEOUT
2819                            <1> ;     dec    bl
2820                            <1> ;     jz     short J36A
2821                            <1> ;     jmp    short WFMS_CHECK_MEM
2822                            <1>
2823                            <1>       ;17/12/2014
2824                            <1>       ;16/12/2014
2825                            <1> ;        mov     byte [wait_count], 0    ; Reset (INT 08H) counter
2826                            <1> ;J36:
2827                            <1> ;     TEST   byte [SEEK_STATUS],INT_FLAG ; TEST FOR INTERRUPT OCCURRING
2828                            <1> ;     JNZ    short J37
2829                            <1>       ;16/12/2014
2830                            <1>       ;LOOP  J36                 ; COUNT DOWN WHILE WAITING
2831                            <1>       ;DEC   BL                  ; SECOND LEVEL COUNTER
2832                            <1>       ;JNZ   short J36
2833                            <1> ;       cmp     byte [wait_count], 46   ; (46/18.2 seconds)
2834                            <1> ;     jb     short J36
2835                            <1>
2836                            <1> ;WFMS_TIMEOUT:
2837                            <1> ;J36A:
2838 000040D5 800D[88520100]80 <1>       OR     byte [DSKETTE_STATUS], TIME_OUT ; NOTHING HAPPENED
2839 000040DC F9               <1>       STC                        ; ERROR RETURN
2840                            <1> J37:
2841 000040DD 9C               <1>       PUSHF                      ; SAVE CURRENT CARRY
2842 000040DE 8025[85520100]7F <1>       AND    byte [SEEK_STATUS], ~INT_FLAG ; TURN OFF INTERRUPT FLAG
2843 000040E5 9D               <1>       POPF                       ; RECOVER CARRY
2844 000040E6 C3               <1>       RETn                       ; GOOD RETURN CODE
2845                            <1>
2846                            <1> ;--------------------------------------------------------------------------------
2847                            <1> ; RESULTS
2848                            <1> ;     THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER RETURNS
2849                            <1> ;     FOLLOWING AN INTERRUPT.
2850                            <1> ;
2851                            <1> ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
2852                            <1> ;           AX,BX,CX,DX DESTROYED
2853                            <1> ;--------------------------------------------------------------------------------
2854                            <1> RESULTS:
2855 000040E7 57               <1>       PUSH   eDI
2856 000040E8 BF[89520100]     <1>       MOV    eDI, NEC_STATUS           ; POINTER TO DATA AREA
2857 000040ED B307             <1>       MOV    BL,7                ; MAX STATUS BYTES
2858 000040EF 66BAF403         <1>       MOV    DX,03F4H            ; STATUS PORT
2859                            <1>
2860                            <1> ;-----      WAIT FOR REQUEST FOR MASTER
2861                            <1>
2862                            <1> _R10:
2863                            <1>       ; 16/12/2014
2864                            <1>       ; wait for (max) 0.5 seconds
2865                            <1>       ;MOV   BH,2                ; HIGH ORDER COUNTER
2866                            <1>       ;XOR   CX,CX               ; COUNTER
2867                            <1>
2868                            <1>       ;Time to wait while waiting for each byte of NEC results = .5
2869                            <1>       ;seconds.  .5 seconds = 500,000 micros.  500,000/30 = 16,667.
2870                            <1>       ; 27/02/2015
2871 000040F3 B91B410000       <1>       mov    ecx, WAIT_FDU_RESULTS_LH ; 16667
```

```
2872                                    <1>       ;mov   cx, WAIT_FDU_RESULTS_LO  ; 16667
2873                                    <1>       ;mov   bh, WAIT_FDU_RESULTS_HI+1 ; 0+1
2874                                    <1>
2875                                    <1> WFPSR_OUTER_LP:
2876                                    <1>       ;
2877                                    <1> WFPSR_CHECK_PORT:
2878                                    <1> J39:                            ; WAIT FOR MASTER
2879 000040F8 EC                       <1>       IN     AL,DX              ; GET STATUS
2880 000040F9 24C0                     <1>       AND    AL,11000000B       ; KEEP ONLY STATUS AND DIRECTION
2881 000040FB 3CC0                     <1>       CMP    AL,11000000B       ; STATUS 1 AND DIRECTION 1 ?
2882 000040FD 7418                     <1>       JZ     short J42          ; STATUS AND DIRECTION OK
2883                                    <1> WFPSR_HI:
2884 000040FF E461                     <1>       IN     AL, PORT_B   ;061h  ; SYS1 ; wait for hi to lo
2885 00004101 A810                     <1>       TEST   AL,010H                   ; transition on memory
2886 00004103 75FA                     <1>       JNZ    SHORT WFPSR_HI            ; refresh.
2887                                    <1> WFPSR_LO:
2888 00004105 E461                     <1>       IN     AL, PORT_B          ; SYS1
2889 00004107 A810                     <1>       TEST   AL,010H
2890 00004109 74FA                     <1>       JZ     SHORT WFPSR_LO
2891 0000410B E2EB                     <1>         LOOP    WFPSR_CHECK_PORT
2892                                    <1>       ;; 27/02/2015
2893                                    <1>       ;;dec  bh
2894                                    <1>       ;;jnz  short WFPSR_OUTER_LP
2895                                    <1>       ;jmp   short WFPSR_TIMEOUT ; fail
2896                                    <1>
2897                                    <1>       ;;mov  byte [wait_count], 0
2898                                    <1> ;J39:                           ; WAIT FOR MASTER
2899                                    <1> ;     IN     AL,DX              ; GET STATUS
2900                                    <1> ;     AND    AL,11000000B       ; KEEP ONLY STATUS AND DIRECTION
2901                                    <1> ;     CMP    AL,11000000B       ; STATUS 1 AND DIRECTION 1 ?
2902                                    <1> ;     JZ     short J42          ; STATUS AND DIRECTION OK
2903                                    <1>       ;LOOP  J39                ; LOOP TILL TIMEOUT
2904                                    <1>       ;DEC   BH                 ; DECREMENT HIGH ORDER COUNTER
2905                                    <1>       ;JNZ   short J39          ; REPEAT TILL DELAY DONE
2906                                    <1>       ;
2907                                    <1>       ;;cmp  byte [wait_count], 10  ; (10/18.2 seconds)
2908                                    <1>       ;;jb   short J39
2909                                    <1>
2910                                    <1> ;WFPSR_TIMEOUT:
2911 0000410D 800D[88520100]80         <1>       OR     byte [DSKETTE_STATUS],TIME_OUT
2912 00004114 F9                       <1>       STC                        ; SET ERROR RETURN
2913 00004115 EB29                     <1>       JMP    SHORT POPRES        ; POP REGISTERS AND RETURN
2914                                    <1>
2915                                    <1> ;-----      READ IN THE STATUS
2916                                    <1>
2917                                    <1> J42:
2918 00004117 EB00                     <1>       JMP    $+2                 ; I/O DELAY
2919 00004119 6642                     <1>       INC    DX                  ; POINT AT DATA PORT
2920 0000411B EC                       <1>       IN     AL,DX               ; GET THE DATA
2921                                    <1>       ; 16/12/2014
2922                                    <1>       NEWIODELAY
2922 0000411C E6EB                     <2>  out 0ebh,al
2923 0000411E 8807                     <1>         MOV     [eDI],AL                ; STORE THE BYTE
2924 00004120 47                       <1>       INC    eDI                 ; INCREMENT THE POINTER
2925                                    <1>       ; 16/12/2014
2926                                    <1> ;     push   cx
2927                                    <1> ;     mov    cx, 30
2928                                    <1> ;wdw2:
2929                                    <1> ;     NEWIODELAY
2930                                    <1> ;     loop   wdw2
2931                                    <1> ;     pop    cx
2932                                    <1>
2933 00004121 B903000000               <1>       MOV    eCX,3               ; MINIMUM 24 MICROSECONDS FOR NEC
2934 00004126 E8C2DCFFFF               <1>       CALL   WAITF               ; WAIT 30 TO 45 MICROSECONDS
2935 0000412B 664A                     <1>       DEC    DX                  ; POINT AT STATUS PORT
2936 0000412D EC                       <1>       IN     AL,DX               ; GET STATUS
2937                                    <1>       ; 16/12/2014
2938                                    <1>       NEWIODELAY
2938 0000412E E6EB                     <2>  out 0ebh,al
2939                                    <1>       ;
2940 00004130 A810                     <1>       TEST   AL,00010000B        ; TEST FOR NEC STILL BUSY
2941 00004132 740C                     <1>       JZ     short POPRES        ; RESULTS DONE ?
2942                                    <1>
2943 00004134 FECB                     <1>       DEC    BL                  ; DECREMENT THE STATUS COUNTER
2944 00004136 75BB                     <1>         JNZ     short _R10              ; GO BACK FOR MORE
2945 00004138 800D[88520100]20         <1>       OR     byte [DSKETTE_STATUS],BAD_NEC ; TOO MANY STATUS BYTES
2946 0000413F F9                       <1>       STC                        ; SET ERROR FLAG
2947                                    <1>
2948                                    <1> ;-----      RESULT OPERATION IS DONE
2949                                    <1> POPRES:
2950 00004140 5F                       <1>       POP    eDI
2951 00004141 C3                       <1>       RETn                       ; RETURN WITH CARRY SET
2952                                    <1>
2953                                    <1> ;--------------------------------------------------------------------------
2954                                    <1> ; READ_DSKCHNG
2955                                    <1> ;     READS THE STATE OF THE DISK CHANGE LINE.
2956                                    <1> ;
2957                                    <1> ; ON ENTRY:  DI = DRIVE #
2958                                    <1> ;
2959                                    <1> ; ON EXIT:   DI = DRIVE #
2960                                    <1> ;            ZF = 0 : DISK CHANGE LINE INACTIVE
2961                                    <1> ;            ZF = 1 : DISK CHANGE LINE ACTIVE
2962                                    <1> ;            AX,CX,DX DESTROYED
2963                                    <1> ;--------------------------------------------------------------------------
2964                                    <1> READ_DSKCHNG:
2965 00004142 E8A2FDFFFF               <1>       CALL   MOTOR_ON            ; TURN ON THE MOTOR IF OFF
2966 00004147 66BAF703                 <1>       MOV    DX,03F7H            ; ADDRESS DIGITAL INPUT REGISTER
2967 0000414B EC                       <1>       IN     AL,DX               ; INPUT DIGITAL INPUT REGISTER
2968 0000414C A880                     <1>       TEST   AL,DSK_CHG          ; CHECK FOR DISK CHANGE LINE ACTIVE
2969 0000414E C3                       <1>       RETn                       ; RETURN TO CALLER WITH ZERO FLAG SET
2970                                    <1>
2971                                    <1> ;--------------------------------------------------------------------------
```

```
2972                            <1> ; DRIVE_DET
2973                            <1> ;     DETERMINES WHETHER DRIVE IS 80 OR 40 TRACKS AND
2974                            <1> ;     UPDATES STATE INFORMATION ACCORDINGLY.
2975                            <1> ; ON ENTRY:  DI = DRIVE #
2976                            <1> ;--------------------------------------------------------------------------------
2977                            <1> DRIVE_DET:
2978 0000414F E895FDFFFF          <1>       CALL   MOTOR_ON            ; TURN ON MOTOR IF NOT ALREADY ON
2979 00004154 E80AFFFFFF          <1>       CALL   RECAL               ; RECALIBRATE DRIVE
2980 00004159 7251                <1>       JC     short DD_BAC        ; ASSUME NO DRIVE PRESENT
2981 0000415B B530                <1>       MOV    CH,TRK_SLAP         ; SEEK TO TRACK 48
2982 0000415D E882FEFFFF          <1>       CALL   SEEK
2983 00004162 7248                <1>       JC     short DD_BAC        ; ERROR NO DRIVE
2984 00004164 B50B                <1>       MOV    CH,QUIET_SEEK+1           ; SEEK TO TRACK 10
2985                            <1> SK_GIN:
2986 00004166 FECD                <1>       DEC    CH                  ; DECREMENT TO NEXT TRACK
2987 00004168 6651                <1>       PUSH   CX                  ; SAVE TRACK
2988 0000416A E875FEFFFF          <1>       CALL   SEEK
2989 0000416F 723C                <1>       JC     short POP_BAC       ; POP AND RETURN
2990 00004171 B8[AD410000]        <1>       MOV    eAX, POP_BAC        ; LOAD NEC OUTPUT ERROR ADDRESS
2991 00004176 50                  <1>       PUSH   eAX
2992 00004177 B404                <1>       MOV    AH,SENSE_DRV_ST           ; SENSE DRIVE STATUS COMMAND BYTE
2993 00004179 E82CFEFFFF          <1>       CALL   NEC_OUTPUT          ; OUTPUT TO NEC
2994 0000417E 6689F8              <1>       MOV    AX,DI               ; AL = DRIVE
2995 00004181 88C4                <1>       MOV    AH,AL               ; AH = DRIVE
2996 00004183 E822FEFFFF          <1>       CALL   NEC_OUTPUT          ; OUTPUT TO NEC
2997 00004188 E85AFFFFFF          <1>       CALL   RESULTS                   ; GO GET STATUS
2998 0000418D 58                  <1>       POP    eAX                 ; THROW AWAY ERROR ADDRESS
2999 0000418E 6659                <1>       POP    CX                  ; RESTORE TRACK
3000 00004190 F605[89520100]10    <1>       TEST   byte [NEC_STATUS], HOME   ; TRACK 0 ?
3001 00004197 74CD                <1>       JZ     short SK_GIN        ; GO TILL TRACK 0
3002 00004199 08ED                <1>       OR     CH,CH               ; IS HOME AT TRACK 0
3003 0000419B 7408                <1>       JZ     short IS_80         ; MUST BE 80 TRACK DRIVE
3004                            <1>
3005                            <1> ;     DRIVE IS A 360; SET DRIVE TO DETERMINED;
3006                            <1> ;     SET MEDIA TO DETERMINED AT RATE 250.
3007                            <1>
3008 0000419D 808F[95520100]94    <1>       OR     byte [DSK_STATE+eDI], DRV_DET+MED_DET+RATE_250
3009 000041A4 C3                  <1>       RETn                       ; ALL INFORMATION SET
3010                            <1> IS_80:
3011 000041A5 808F[95520100]01    <1>       OR     byte [DSK_STATE+eDI], TRK_CAPA ; SETUP 80 TRACK CAPABILITY
3012                            <1> DD_BAC:
3013 000041AC C3                  <1>       RETn
3014                            <1> POP_BAC:
3015 000041AD 6659                <1>       POP    CX                  ; THROW AWAY
3016 000041AF C3                  <1>       RETn
3017                            <1>
3018                            <1> fdc_int:
3019                            <1>         ; 30/07/2015
3020                            <1>         ; 16/02/2015
3021                            <1> ;int_0Eh: ; 11/12/2014
3022                            <1>
3023                            <1> ;--- HARDWARE INT 0EH -- ( IRQ LEVEL  6 ) -------------------------------------
3024                            <1> ; DISK_INT
3025                            <1> ;     THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.
3026                            <1> ;
3027                            <1> ; ON EXIT:  THE INTERRUPT FLAG IS SET IN @SEEK_STATUS.
3028                            <1> ;--------------------------------------------------------------------------------
3029                            <1> DISK_INT_1:
3030                            <1>
3031 000041B0 6650                <1>       PUSH   AX                  ; SAVE WORK REGISTER
3032 000041B2 1E                  <1>       push   ds
3033 000041B3 66B81000            <1>       mov    ax, KDATA
3034 000041B7 8ED8                <1>       mov    ds, ax
3035 000041B9 800D[85520100]80    <1>         OR      byte [SEEK_STATUS], INT_FLAG ; TURN ON INTERRUPT OCCURRED
3036 000041C0 B020                <1>       MOV     AL,EOI                   ; END OF INTERRUPT MARKER
3037 000041C2 E620                <1>       OUT    INTA00,AL           ; INTERRUPT CONTROL PORT
3038 000041C4 1F                  <1>       pop    ds
3039 000041C5 6658                <1>       POP    AX                  ; RECOVER REGISTER
3040 000041C7 CF                  <1>       IRETd                      ; RETURN FROM INTERRUPT
3041                            <1>
3042                            <1> ;--------------------------------------------------------------------------------
3043                            <1> ; DSKETTE_SETUP
3044                            <1> ;     THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE OF
3045                            <1> ;     DISKETTE DRIVES ARE ATTACH TO THE SYSTEM.
3046                            <1> ;--------------------------------------------------------------------------------
3047                            <1>
3048                            <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
3049                            <1>
3050                            <1> DSKETTE_SETUP:
3051                            <1>       ;PUSH  AX                  ; SAVE REGISTERS
3052                            <1>       ;PUSH  BX
3053                            <1>       ;PUSH  CX
3054 000041C8 52                  <1>       PUSH   eDX
3055                            <1>       ;PUSH  DI
3056                            <1>       ;;PUSH DS
3057                            <1>       ; 14/12/2014
3058                            <1>       ;mov   word [DISK_POINTER], MD_TBL6
3059                            <1>       ;mov   [DISK_POINTER+2], cs
3060                            <1>       ;
3061                            <1>       ;OR    byte [RTC_WAIT_FLAG], 1   ; NO RTC WAIT, FORCE USE OF LOOP
3062 000041C9 31FF                <1>       XOR    eDI,eDI                   ; INITIALIZE DRIVE POINTER
3063 000041CB 66C705[95520100]00- <1>       MOV    WORD [DSK_STATE],0  ; INITIALIZE STATES
3063 000041D3 00                  <1>
3064 000041D4 8025[90520100]33    <1>       AND    byte [LASTRATE],~(STRT_MSK+SEND_MSK) ; CLEAR START & SEND
3065 000041DB 800D[90520100]C0    <1>       OR     byte [LASTRATE],SEND_MSK ; INITIALIZE SENT TO IMPOSSIBLE
3066 000041E2 C605[85520100]00    <1>       MOV    byte [SEEK_STATUS],0      ; INDICATE RECALIBRATE NEEDED
3067 000041E9 C605[87520100]00    <1>       MOV    byte [MOTOR_COUNT],0      ; INITIALIZE MOTOR COUNT
3068 000041F0 C605[86520100]00    <1>       MOV    byte [MOTOR_STATUS],0     ; INITIALIZE DRIVES TO OFF STATE
3069 000041F7 C605[88520100]00    <1>       MOV    byte [DSKETTE_STATUS],0   ; NO ERRORS
3070                            <1>       ;
3071                            <1>       ; 28/02/2015
3072                            <1>       ;mov   word [cfd], 100h
```

```
3073 000041FE E848F2FFFF          <1>       call   DSK_RESET
3074 00004203 5A                  <1>       pop    edx
3075 00004204 F8                  <1>       clc    ; 29/05/2016
3076 00004205 C3                  <1>       retn
3077                              <1>
3078                              <1> ;SUP0:
3079                              <1> ;     CALL   DRIVE_DET           ; DETERMINE DRIVE
3080                              <1> ;     CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
3081                              <1> ;     ; 02/01/2015
3082                              <1> ;     ;INC   DI                  ; POINT TO NEXT DRIVE
3083                              <1> ;     ;CMP   DI,MAX_DRV          ; SEE IF DONE
3084                              <1> ;     ;JNZ   short SUP0          ; REPEAT FOR EACH ORIVE
3085                              <1> ;       cmp     byte [fd1_type], 0
3086                              <1> ;     jna    short sup1
3087                              <1> ;     or     di, di
3088                              <1> ;     jnz    short sup1
3089                              <1> ;     inc    di
3090                              <1> ;       jmp     short SUP0
3091                              <1> ;sup1:
3092                              <1> ;     MOV    byte [SEEK_STATUS],0       ; FORCE RECALIBRATE
3093                              <1> ;     ;AND   byte [RTC_WAIT_FLAG],0FEH ; ALLOW FOR RTC WAIT
3094                              <1> ;     CALL   SETUP_END           ; VARIOUS CLEANUPS
3095                              <1> ;     ;;POP  DS                  ; RESTORE CALLERS REGISTERS
3096                              <1> ;     ;POP   DI
3097                              <1> ;     POP    eDX
3098                              <1> ;     ;POP   CX
3099                              <1> ;     ;POP   BX
3100                              <1> ;     ;POP   AX
3101                              <1> ;     RETn
3102                              <1>
3103                              <1> ;////////////////////////////////////////////////////
3104                              <1> ;; END OF DISKETTE I/O ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
3105                              <1> ;
3106                              <1>
3107                              <1> int13h: ; 21/02/2015
3108 00004206 9C                  <1>       pushfd
3109 00004207 0E                  <1>       push   cs
3110 00004208 E843010000          <1>       call   DISK_IO
3111 0000420D C3                  <1>       retn
3112                              <1>
3113                              <1> ;;;;;; DISK I/O ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; 21/02/2015 ;;;
3114                              <1> ;////////////////////////////////////////////////////////////////////
3115                              <1>
3116                              <1> ; DISK I/O - Erdogan Tan (Retro UNIX 386 v1 project)
3117                              <1> ; 18/02/2016
3118                              <1> ; 17/02/2016
3119                              <1> ; 23/02/2015
3120                              <1> ; 21/02/2015 (unix386.s)
3121                              <1> ; 22/12/2014 - 14/02/2015 (dsectrm2.s)
3122                              <1> ;
3123                              <1> ; Original Source Code:
3124                              <1> ; DISK ----- 09/25/85 FIXED DISK BIOS
3125                              <1> ; (IBM PC XT Model 286 System BIOS Source Code, 04-21-86)
3126                              <1> ;
3127                              <1> ; Modifications: by reference of AWARD BIOS 1999 (D1A0622)
3128                              <1> ;             Source Code - ATORGS.ASM, AHDSK.ASM
3129                              <1> ;
3130                              <1>
3131                              <1>
3132                              <1> ;The wait for controller to be not busy is 10 seconds.
3133                              <1> ;10,000,000 / 30 = 333,333.  333,333 decimal = 051615h
3134                              <1> ;;WAIT_HDU_CTLR_BUSY_LO   equ    1615h
3135                              <1> ;;WAIT_HDU_CTLR_BUSY_HI   equ      05h
3136                              <1> WAIT_HDU_CTRL_BUSY_LH     equ    51615h  ;21/02/2015
3137                              <1>
3138                              <1> ;The wait for controller to issue completion interrupt is 10 seconds.
3139                              <1> ;10,000,000 / 30 = 333,333.  333,333 decimal = 051615h
3140                              <1> ;;WAIT_HDU_INT_LO  equ    1615h
3141                              <1> ;;WAIT_HDU_INT_HI  equ      05h
3142                              <1> WAIT_HDU_INT_LH           equ    51615h ; 21/02/2015
3143                              <1>
3144                              <1> ;The wait for Data request on read and write longs is
3145                              <1> ;2000 us. (?)
3146                              <1> ;;WAIT_HDU_DRQ_LO  equ    1000   ; 03E8h
3147                              <1> ;;WAIT_HDU_DRQ_HI  equ    0
3148                              <1> WAIT_HDU_DRQ_LH           equ    1000   ; 21/02/2015
3149                              <1>
3150                              <1> ; Port 61h (PORT_B)
3151                              <1> SYS1        equ    61h    ; PORT_B  (diskette.inc)
3152                              <1>
3153                              <1> ; 23/12/2014
3154                              <1> %define CMD_BLOCK       eBP-8  ; 21/02/2015
3155                              <1>
3156                              <1>
3157                              <1> ;--- INT 13H ----------------------------------------------------------------------
3158                              <1> ;                                                                              :
3159                              <1> ; FIXED DISK I/O INTERFACE                                                       :
3160                              <1> ;                                                                              :
3161                              <1> ;     THIS INTERFACE PROVIDES ACCESS TO 5 1/4" FIXED DISKS THROUGH              :
3162                              <1> ;     THE IBM FIXED DISK CONTROLLER.                                           :
3163                              <1> ;                                                                              :
3164                              <1> ;     THE  BIOS  ROUTINES  ARE  MEANT  TO  BE  ACCESSED  THROUGH             :
3165                              <1> ;     SOFTWARE  INTERRUPTS  ONLY.    ANY  ADDRESSES  PRESENT     IN           :
3166                              <1> ;     THESE  LISTINGS  ARE  INCLUDED    ONLY  FOR  COMPLETENESS,             :
3167                              <1> ;     NOT  FOR  REFERENCE.  APPLICATIONS   WHICH  REFERENCE  ANY             :
3168                              <1> ;     ABSOLUTE  ADDRESSES  WITHIN  THE  CODE SEGMENTS  OF  BIOS              :
3169                              <1> ;     VIOLATE  THE  STRUCTURE  AND  DESIGN  OF  BIOS.                        :
3170                              <1> ;                                                                              :
3171                              <1> ;------------------------------------------------------------------------------:
3172                              <1> ;                                                                              :
3173                              <1> ; INPUT  (AH)= HEX COMMAND VALUE                                               :
3174                              <1> ;                                                                              :
```

```
3175                            <1> ;     (AH)= 00H  RESET DISK (DL = 80H,81H) / DISKETTE                    :
3176                            <1> ;     (AH)= 01H  READ THE STATUS OF THE LAST DISK OPERATION INTO (AL)        :
3177                            <1> ;                NOTE: DL < 80H - DISKETTE                                 :
3178                            <1> ;                    DL > 80H - DISK                                    :
3179                            <1> ;     (AH)= 02H  READ THE DESIRED SECTORS INTO MEMORY                     :
3180                            <1> ;     (AH)= 03H  WRITE THE DESIRED SECTORS FROM MEMORY                    :
3181                            <1> ;     (AH)= 04H  VERIFY THE DESIRED SECTORS                               :
3182                            <1> ;     (AH)= 05H  FORMAT THE DESIRED TRACK                                 :
3183                            <1> ;     (AH)= 06H  UNUSED                                                   :
3184                            <1> ;     (AH)= 07H  UNUSED                                                   :
3185                            <1> ;     (AH)= 08H  RETURN THE CURRENT DRIVE PARAMETERS                           :
3186                            <1> ;     (AH)= 09H  INITIALIZE DRIVE PAIR CHARACTERISTICS                    :
3187                            <1> ;                INTERRUPT 41 POINTS TO DATA BLOCK FOR DRIVE 0          :
3188                            <1> ;                INTERRUPT 46 POINTS TO DATA BLOCK FOR DRIVE 1          :
3189                            <1> ;     (AH)= 0AH  READ LONG                                                :
3190                            <1> ;     (AH)= 0BH  WRITE LONG  (READ & WRITE LONG ENCOMPASS 512 + 4 BYTES ECC) :
3191                            <1> ;     (AH)= 0CH  SEEK                                                     :
3192                            <1> ;     (AH)= 0DH  ALTERNATE DISK RESET (SEE DL)                            :
3193                            <1> ;     (AH)= 0EH  UNUSED                                                   :
3194                            <1> ;     (AH)= 0FH  UNUSED                                                   :
3195                            <1> ;     (AH)= 10H  TEST DRIVE READY                                         :
3196                            <1> ;     (AH)= 11H  RECALIBRATE                                              :
3197                            <1> ;     (AH)= 12H  UNUSED                                                   :
3198                            <1> ;     (AH)= 13H  UNUSED                                                   :
3199                            <1> ;     (AH)= 14H  CONTROLLER INTERNAL DIAGNOSTIC                           :
3200                            <1> ;     (AH)= 15H  READ DASD TYPE                                           :
3201                            <1> ;                                                                         :
3202                            <1> ;--------------------------------------------------------------------------
3203                            <1> ;                                                                         :
3204                            <1> ;     REGISTERS USED FOR FIXED DISK OPERATIONS                            :
3205                            <1> ;                                                                         :
3206                            <1> ;           (DL)   -  DRIVE NUMBER     (80H-81H FOR DISK. VALUE CHECKED)  :
3207                            <1> ;           (DH)   -  HEAD NUMBER         (0-15 ALLOWED, NOT VALUE CHECKED)  :
3208                            <1> ;           (CH)   -  CYLINDER NUMBER  (0-1023, NOT VALUE CHECKED)(SEE CL):
3209                            <1> ;           (CL)   -  SECTOR NUMBER    (1-17, NOT VALUE CHECKED)        :
3210                            <1> ;                                                                       :
3211                            <1> ;                     NOTE: HIGH 2 BITS OF CYLINDER NUMBER ARE PLACED     :
3212                            <1> ;                         IN THE HIGH 2 BITS OF THE CL REGISTER         :
3213                            <1> ;                         (10 BITS TOTAL)                               :
3214                            <1> ;                                                                       :
3215                            <1> ;           (AL)   -  NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H,    :
3216                            <1> ;                                    FOR READ/WRITE LONG 1-79H)       :
3217                            <1> ;                                                                       :
3218                            <1> ;           (ES:BX) -  ADDRESS OF BUFFER FOR READS AND WRITES,          :
3219                            <1> ;                      (NOT REQUIRED FOR VERIFY)                        :
3220                            <1> ;                                                                       :
3221                            <1> ;           FORMAT (AH=5) ES:BX POINTS TO A 512 BYTE BUFFER. THE FIRST     :
3222                            <1> ;                      2*(SECTORS/TRACK) BYTES CONTAIN F,N FOR EACH SECTOR.:
3223                            <1> ;                      F = 00H FOR A GOOD SECTOR                        :
3224                            <1> ;                          80H FOR A BAD SECTOR                         :
3225                            <1> ;                      N = SECTOR NUMBER                                :
3226                            <1> ;                      FOR AN INTERLEAVE OF 2 AND 17 SECTORS/TRACK             :
3227                            <1> ;                      THE TABLE SHOULD BE:                             :
3228                            <1> ;                                                                       :
3229                            <1> ;              DB      00H,01H,00H,0AH,00H,02H,00H,0BH,00H,03H,00H,0CH     :
3230                            <1> ;              DB      00H,04H,00H,0DH,00H,05H,00H,0EH,00H,06H,00H,0FH     :
3231                            <1> ;              DB      00H,07H,00H,10H,00H,08H,00H,11H,00H,09H          :
3232                            <1> ;                                                                       :
3233                            <1> ;--------------------------------------------------------------------------
3234                            <1>
3235                            <1> ;--------------------------------------------------------------------------
3236                            <1> ; OUTPUT                                                                  :
3237                            <1> ;     AH = STATUS OF CURRENT OPERATION                                    :
3238                            <1> ;          STATUS BITS ARE DEFINED IN THE EQUATES BELOW                       :
3239                            <1> ;     CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN)                        :
3240                            <1> ;     CY = 1 FAILED OPERATION (AH HAS ERROR REASON)                       :
3241                            <1> ;                                                                         :
3242                            <1> ;     NOTE:  ERROR 11H  INDICATES THAT THE DATA READ HAD A RECOVERABLE      :
3243                            <1> ;            ERROR WHICH WAS CORRECTED BY THE ECC ALGORITHM.  THE DATA      :
3244                            <1> ;            IS PROBABLY GOOD,   HOWEVER THE BIOS ROUTINE INDICATES AN      :
3245                            <1> ;            ERROR TO ALLOW THE CONTROLLING PROGRAM A CHANCE TO DECIDE       :
3246                            <1> ;             FOR ITSELF.  THE  ERROR  MAY  NOT  RECUR  IF  THE DATA IS     :
3247                            <1> ;             REWRITTEN.                                                  :
3248                            <1> ;                                                                         :
3249                            <1> ;     IF DRIVE PARAMETERS WERE REQUESTED (DL >= 80H),                        :
3250                            <1> ;        INPUT:                                                           :
3251                            <1> ;          (DL) = DRIVE NUMBER                                            :
3252                            <1> ;          ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)                                  :

3253                            <1> ;           EBX = Buffer address for fixed disk parameters table (32 bytes)   :
3254                            <1> ;        OUTPUT:                                                          :
3255                            <1> ;          (DL) = NUMBER OF CONSECUTIVE ACKNOWLEDGING DRIVES ATTACHED (1-2)  :
3256                            <1> ;               (CONTROLLER CARD ZERO TALLY ONLY)                         :
3257                            <1> ;          (DH) = MAXIMUM USEABLE VALUE FOR HEAD NUMBER                   :
3258                            <1> ;          (CH) = MAXIMUM USEABLE VALUE FOR CYLINDER NUMBER                    :
3259                            <1> ;          (CL) = MAXIMUM USEABLE VALUE FOR SECTOR NUMBER                 :
3260                            <1> ;               AND CYLINDER NUMBER HIGH BITS                             :
3261                            <1> ;                                                                         :
3262                            <1> ;     IF READ DASD TYPE WAS REQUESTED,                                    :
3263                            <1> ;                                                                         :
3264                            <1> ;     AH = 0 - NOT PRESENT                                                :
3265                            <1> ;          1 - DISKETTE - NO CHANGE LINE AVAILABLE                        :
3266                            <1> ;          2 - DISKETTE - CHANGE LINE AVAILABLE                           :
3267                            <1> ;          3 - FIXED DISK                                                 :
3268                            <1> ;                                                                         :
3269                            <1> ;     CX,DX = NUMBER OF 512 BYTE BLOCKS WHEN AH = 3                       :
3270                            <1> ;                                                                         :
3271                            <1> ;     REGISTERS WILL BE PRESERVED EXCEPT WHEN THEY ARE USED TO RETURN        :
3272                            <1> ;     INFORMATION.                                                        :
3273                            <1> ;                                                                         :
3274                            <1> ;     NOTE: IF AN ERROR IS REPORTED BY THE DISK CODE, THE APPROPRIATE        :
3275                            <1> ;           ACTION IS TO RESET THE DISK, THEN RETRY THE OPERATION.            :
```

```
3276                            <1> ;                                                                      :
3277                            <1> ;---------------------------------------------------------------------------
3278                            <1>
3279                            <1> SENSE_FAIL   EQU    0FFH          ; NOT IMPLEMENTED
3280                            <1> NO_ERR               EQU    0E0H          ; STATUS ERROR/ERROR REGISTER=0
3281                            <1> WRITE_FAULT  EQU    0CCH          ; WRITE FAULT ON SELECTED DRIVE
3282                            <1> UNDEF_ERR    EQU    0BBH          ; UNDEFINED ERROR OCCURRED
3283                            <1> NOT_RDY      EQU    0AAH          ; DRIVE NOT READY
3284                            <1> TIME_OUT     EQU    80H           ; ATTACHMENT FAILED TO RESPOND
3285                            <1> BAD_SEEK     EQU    40H           ; SEEK OPERATION FAILED
3286                            <1> BAD_CNTLR    EQU    20H           ; CONTROLLER HAS FAILED
3287                            <1> DATA_CORRECTED       EQU    11H           ; ECC CORRECTED DATA ERROR
3288                            <1> BAD_ECC      EQU    10H           ; BAD ECC ON DISK READ
3289                            <1> BAD_TRACK    EQU    0BH           ; NOT IMPLEMENTED
3290                            <1> BAD_SECTOR   EQU    0AH           ; BAD SECTOR FLAG DETECTED
3291                            <1> ;DMA_BOUNDARY        EQU    09H           ; DATA EXTENDS TOO FAR
3292                            <1> INIT_FAIL    EQU    07H           ; DRIVE PARAMETER ACTIVITY FAILED
3293                            <1> BAD_RESET    EQU    05H           ; RESET FAILED
3294                            <1> ;RECORD_NOT_FND      EQU    04H           ; REQUESTED SECTOR NOT FOUND
3295                            <1> ;BAD_ADDR_MARK       EQU    02H           ; ADDRESS MARK NOT FOUND
3296                            <1> ;BAD_CMD     EQU    01H           ; BAD COMMAND PASSED TO DISK I/O
3297                            <1>
3298                            <1> ;---------------------------------------------------
3299                            <1> ;                                          :
3300                            <1> ; FIXED DISK PARAMETER TABLE                    :
3301                            <1> ;  -  THE TABLE IS COMPOSED OF A BLOCK DEFINED AS:  :
3302                            <1> ;                                          :
3303                            <1> ;  +0 (1 WORD) - MAXIMUM NUMBER OF CYLINDERS        :
3304                            <1> ;  +2 (1 BYTE) - MAXIMUM NUMBER OF HEADS            :
3305                            <1> ;  +3 (1 WORD) - NOT USED/SEE PC-XT                 :
3306                            <1> ;  +5 (1 WORD) - STARTING WRITE PRECOMPENSATION CYL :
3307                            <1> ;  +7 (1 BYTE) - MAXIMUM ECC DATA BURST LENGTH      :
3308                            <1> ;  +8 (1 BYTE) - CONTROL BYTE                 :
3309                            <1> ;                BIT   7 DISABLE RETRIES -OR-   :
3310                            <1> ;                BIT   6 DISABLE RETRIES        :
3311                            <1> ;                BIT   3 MORE THAN 8 HEADS            :
3312                            <1> ;  +9 (3 BYTES)- NOT USED/SEE PC-XT                 :
3313                            <1> ; +12 (1 WORD) - LANDING ZONE                 :
3314                            <1> ; +14 (1 BYTE) - NUMBER OF SECTORS/TRACK            :
3315                            <1> ; +15 (1 BYTE) - RESERVED FOR FUTURE USE            :
3316                            <1> ;                                          :
3317                            <1> ;      - TO DYNAMICALLY DEFINE A SET OF PARAMETERS  :
3318                            <1> ;        BUILD A TABLE FOR UP TO 15 TYPES AND PLACE :
3319                            <1> ;        THE CORRESPONDING VECTOR INTO INTERRUPT 41 :
3320                            <1> ;        FOR DRIVE 0 AND INTERRUPT 46 FOR DRIVE 1.  :
3321                            <1> ;                                          :
3322                            <1> ;---------------------------------------------------
3323                            <1>
3324                            <1> ;---------------------------------------------------
3325                            <1> ;                                          :
3326                            <1> ; HARDWARE SPECIFIC VALUES                       :
3327                            <1> ;                                          :
3328                            <1> ;  -  CONTROLLER I/O PORT                    :
3329                            <1> ;                                          :
3330                            <1> ;     > WHEN READ FROM:                      :
3331                            <1> ;     HF_PORT+0 - READ DATA (FROM CONTROLLER TO CPU)      :
3332                            <1> ;     HF_PORT+1 - GET ERROR REGISTER               :
3333                            <1> ;     HF_PORT+2 - GET SECTOR COUNT                 :
3334                            <1> ;     HF_PORT+3 - GET SECTOR NUMBER                :
3335                            <1> ;     HF_PORT+4 - GET CYLINDER LOW                 :
3336                            <1> ;     HF_PORT+5 - GET CYLINDER HIGH (2 BITS)       :
3337                            <1> ;     HF_PORT+6 - GET SIZE/DRIVE/HEAD        :
3338                            <1> ;     HF_PORT+7 - GET STATUS REGISTER        :
3339                            <1> ;                                          :
3340                            <1> ;     > WHEN WRITTEN TO:                     :
3341                            <1> ;     HF_PORT+0 - WRITE DATA (FROM CPU TO CONTROLLER) :
3342                            <1> ;     HF_PORT+1 - SET PRECOMPENSATION CYLINDER      :
3343                            <1> ;     HF_PORT+2 - SET SECTOR COUNT                 :
3344                            <1> ;     HF_PORT+3 - SET SECTOR NUMBER                :
3345                            <1> ;     HF_PORT+4 - SET CYLINDER LOW                 :
3346                            <1> ;     HF_PORT+5 - SET CYLINDER HIGH (2 BITS)       :
3347                            <1> ;     HF_PORT+6 - SET SIZE/DRIVE/HEAD        :
3348                            <1> ;     HF_PORT+7 - SET COMMAND REGISTER       :
3349                            <1> ;                                          :
3350                            <1> ;---------------------------------------------------
3351                            <1>
3352                            <1> ;HF_PORT     EQU    01F0H  ; DISK PORT
3353                            <1> ;HF1_PORT    equ    0170h
3354                            <1> ;HF_REG_PORT EQU    03F6H
3355                            <1> ;HF1_REG_PORT        equ    0376h
3356                            <1>
3357                            <1> HDC1_BASEPORT        equ    1F0h
3358                            <1> HDC2_BASEPORT        equ    170h
3359                            <1>
3360                            <1> align 2
3361                            <1>
3362                            <1> ;-----               STATUS REGISTER
3363                            <1>
3364                            <1> ST_ERROR     EQU    00000001B     ;
3365                            <1> ST_INDEX     EQU    00000010B     ;
3366                            <1> ST_CORRCTD   EQU    00000100B     ; ECC CORRECTION SUCCESSFUL
3367                            <1> ST_DRQ               EQU    00001000B     ;
3368                            <1> ST_SEEK_COMPL        EQU    00010000B     ; SEEK COMPLETE
3369                            <1> ST_WRT_FLT   EQU    00100000B     ; WRITE FAULT
3370                            <1> ST_READY     EQU    01000000B     ;
3371                            <1> ST_BUSY      EQU    10000000B     ;
3372                            <1>
3373                            <1> ;-----               ERROR REGISTER
3374                            <1>
3375                            <1> ERR_DAM      EQU    00000001B     ; DATA ADDRESS MARK NOT FOUND
3376                            <1> ERR_TRK_0    EQU    00000010B     ; TRACK 0 NOT FOUND ON RECAL
3377                            <1> ERR_ABORT    EQU    00000100B     ; ABORTED COMMAND
```

```
                                <1> ;           EQU     00001000B       ; NOT USED
                                <1> ERR_ID              EQU     00010000B       ; ID NOT FOUND
                                <1> ;           EQU     00100000B       ; NOT USED
                                <1> ERR_DATA_ECC EQU    01000000B
                                <1> ERR_BAD_BLOCK       EQU     10000000B
                                <1>
                                <1>
                                <1> RECAL_CMD   EQU     00010000B       ; DRIVE RECAL(10H)
                                <1> READ_CMD    EQU     00100000B       ;      READ   (20H)
                                <1> WRITE_CMD   EQU     00110000B       ;      WRITE  (30H)
                                <1> VERIFY_CMD  EQU     01000000B       ;      VERIFY (40H)
                                <1> FMTTRK_CMD  EQU     01010000B       ; FORMAT TRACK      (50H)
                                <1> INIT_CMD    EQU     01100000B       ;   INITIALIZE      (60H)
                                <1> SEEK_CMD    EQU     01110000B       ;      SEEK   (70H)
                                <1> DIAG_CMD    EQU     10010000B       ; DIAGNOSTIC (90H)
                                <1> SET_PARM_CMD EQU    10010001B       ; DRIVE PARMS(91H)
                                <1> NO_RETRIES  EQU     00000001B       ; CHD MODIFIER      (01H)
                                <1> ECC_MODE    EQU     00000010B       ; CMD MODIFIER      (02H)
                                <1> BUFFER_MODE EQU     00001000B       ; CMD MODIFIER      (08H)
                                <1>
                                <1> ;MAX_FILE   EQU     2
                                <1> ;S_MAX_FILE EQU     2
                                <1> MAX_FILE    equ     4               ; 22/12/2014
                                <1> S_MAX_FILE  equ     4               ; 22/12/2014
                                <1>
                                <1> DELAY_1     EQU     25H             ; DELAY FOR OPERATION COMPLETE
                                <1> DELAY_2     EQU     0600H           ; DELAY FOR READY
                                <1> DELAY_3     EQU     0100H           ; DELAY FOR DATA REQUEST
                                <1>
                                <1> HF_FAIL     EQU     08H             ; CMOS FLAG IN BYTE 0EH
                                <1>
                                <1> ;-----              COMMAND BLOCK REFERENCE
                                <1>
                                <1> ;CMD_BLOCK      EQU     BP-8            ; @CMD_BLOCK REFERENCES BLOCK HEAD IN SS
                                <1>                                 ;  (BP) POINTS TO COMMAND BLOCK TAIL
                                <1>                                 ;     AS DEFINED BY THE "ENTER" PARMS
                                <1> ; 19/12/2014
                                <1> ORG_VECTOR  equ     4*13h           ; INT 13h vector
                                <1> DISK_VECTOR equ     4*40h           ; INT 40h vector (for floppy disks)
                                <1> ;HDISK_INT  equ     4*76h           ; Primary HDC - Hardware interrupt (IRQ14)
                                <1> ;HDISK_INT1 equ     4*76h           ; Primary HDC - Hardware interrupt (IRQ14)
                                <1> ;HDISK_INT2 equ     4*77h           ; Secondary HDC - Hardware interrupt (IRQ15)
                                <1> ;HF_TBL_VEC equ     4*41h           ; Pointer to 1st fixed disk parameter table
                                <1> ;HF1_TBL_VEC equ    4*46h           ; Pointer to 2nd fixed disk parameter table
                                <1>
                                <1> align 2
                                <1>
                                <1> ;--------------------------------------------------------------
                                <1> ; FIXED DISK I/O SETUP                                          :
                                <1> ;                                               :
                                <1> ;  -  ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK             :
                                <1> ;  -  PERFORM POWER ON DIAGNOSTICS                              :
                                <1> ;     SHOULD AN ERROR OCCUR A "1701" MESSAGE IS DISPLAYED        :
                                <1> ;                                               :
                                <1> ;--------------------------------------------------------------
                                <1>
                                <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>
                                <1> DISK_SETUP:
                                <1>       ;CLI
                                <1>       ;;MOV  AX,ABS0                    ; GET ABSOLUTE SEGMENT
                                <1>       ;xor   ax,ax
                                <1>       ;MOV   DS,AX                      ; SET SEGMENT REGISTER
                                <1>       ;MOV   AX, [ORG_VECTOR]           ; GET DISKETTE VECTOR
                                <1>       ;MOV   [DISK_VECTOR],AX           ;  INTO INT 40H
                                <1>       ;MOV   AX, [ORG_VECTOR+2]
                                <1>       ;MOV   [DISK_VECTOR+2],AX
                                <1>       ;MOV   word [ORG_VECTOR],DISK_IO  ; FIXED DISK HANDLER
                                <1>       ;MOV   [ORG_VECTOR+2],CS
                                <1>       ; 1st controller (primary master, slave)   - IRQ 14
                                <1>       ;;MOV  word [HDISK_INT],HD_INT          ; FIXED DISK INTERRUPT
                                <1>       ;mov   word [HDISK_INT1],HD_INT  ;
                                <1>       ;;MOV  [HDISK_INT+2],CS
                                <1>       ;mov   [HDISK_INT1+2],CS
                                <1>       ; 2nd controller (secondary master, slave) - IRQ 15
                                <1>       ;mov   word [HDISK_INT2],HD1_INT ;
                                <1>       ;mov   [HDISK_INT2+2],CS
                                <1>       ;
                                <1>       ;;MOV  word [HF_TBL_VEC],HD0_DPT  ; PARM TABLE DRIVE 80
                                <1>       ;;MOV  word [HF_TBL_VEC+2],DPT_SEGM
                                <1>       ;;MOV  word [HF1_TBL_VEC],HD1_DPT; PARM TABLE DRIVE 81
                                <1>       ;;MOV  word [HF1_TBL_VEC+2],DPT_SEGM
                                <1>       ;push  cs
                                <1>       ;pop   ds
                                <1>       ;mov   word [HDPM_TBL_VEC],HD0_DPT      ; PARM TABLE DRIVE 80h
                                <1>       ;mov   word [HDPM_TBL_VEC+2],DPT_SEGM
0000420E C705[A0520100]0000-    <1>       mov    dword [HDPM_TBL_VEC], (DPT_SEGM*16)+HD0_DPT
00004216 0900                   <1>
                                <1>       ;mov   word [HDPS_TBL_VEC],HD1_DPT      ; PARM TABLE DRIVE 81h
                                <1>       ;mov   word [HDPS_TBL_VEC+2],DPT_SEGM
00004218 C705[A4520100]2000-    <1>       mov    dword [HDPS_TBL_VEC], (DPT_SEGM*16)+HD1_DPT
00004220 0900                   <1>
                                <1>       ;mov   word [HDSM_TBL_VEC],HD2_DPT      ; PARM TABLE DRIVE 82h
                                <1>       ;mov   word [HDSM_TBL_VEC+2],DPT_SEGM
00004222 C705[A8520100]4000-    <1>       mov    dword [HDSM_TBL_VEC], (DPT_SEGM*16)+HD2_DPT
0000422A 0900                   <1>
                                <1>       ;mov   word [HDSS_TBL_VEC],HD3_DPT      ; PARM TABLE DRIVE 83h
                                <1>       ;mov   word [HDSS_TBL_VEC+2],DPT_SEGM
0000422C C705[AC520100]6000-    <1>       mov    dword [HDSS_TBL_VEC], (DPT_SEGM*16)+HD3_DPT
00004234 0900                   <1>
                                <1>       ;
                                <1>       ;;IN   AL,INTB01          ; TURN ON SECOND INTERRUPT CHIP
```

```
                        <1>       ;;;AND AL,0BFH
                        <1>       ;;and  al, 3Fh                  ; enable IRQ 14 and IRQ 15
                        <1>       ;;;JMP $+2
                        <1>       ;;IODELAY
                        <1>       ;;OUT  INTB01,AL
                        <1>       ;;IODELAY
                        <1>       ;;IN   AL,INTA01          ; LET INTERRUPTS PASS THRU TO
                        <1>       ;;AND  AL,0FBH            ;  SECOND CHIP
                        <1>       ;;;JMP $+2
                        <1>       ;;IODELAY
                        <1>       ;;OUT  INTA01,AL
                        <1>       ;
                        <1>       ;STI
                        <1>       ;;PUSH DS                 ; MOVE ABS0 POINTER TO
                        <1>       ;;POP  ES                 ; EXTRA SEGMENT POINTER
                        <1>       ;;;CALL      DDS                ; ESTABLISH DATA SEGMENT
                        <1>       ;;MOV  byte [DISK_STATUS1],0     ; RESET THE STATUS INDICATOR
                        <1>       ;;MOV  byte [HF_NUM],0           ; ZERO NUMBER OF FIXED DISKS
                        <1>       ;;MOV  byte [CONTROL_BYTE],0
                        <1>       ;;MOV  byte [PORT_OFF],0   ; ZERO CARD OFFSET
                        <1>       ; 20/12/2014 - private code by Erdogan Tan
                        <1>                    ; (out of original PC-AT, PC-XT BIOS code)
                        <1>       ;mov   si, hd0_type
0000423236 BE[F85C0000] <1>       mov    esi, hd0_type
                        <1>       ;mov   cx, 4
0000423B B904000000     <1>       mov    ecx, 4
                        <1> hde_l:
00004240 AC             <1>       lodsb
00004241 3C80           <1>       cmp    al, 80h                  ; 8?h = existing
00004243 7206           <1>       jb     short _L4
00004245 FE05[9C520100] <1>       inc    byte [HF_NUM]       ; + 1 hard (fixed) disk drives
                        <1> _L4: ; 26/02/2015
0000424B E2F3           <1>       loop   hde_l
                        <1> ;_L4:                           ; 0 <= [HF_NUM] =< 4
                        <1> ;L4:
                        <1>       ;
                        <1>       ;; 31/12/2014 - cancel controller diagnostics here
                        <1>       ;;;mov       cx, 3  ; 26/12/2014 (Award BIOS 1999)
                        <1>       ;;mov  cl, 3
                        <1>       ;;
                        <1>       ;;MOV  DL,80H             ; CHECK THE CONTROLLER
                        <1> ;;hdc_dl:
                        <1>       ;;MOV  AH,14H             ; USE CONTROLLER DIAGNOSTIC COMMAND
                        <1>       ;;INT  13H                ; CALL BIOS WITH DIAGNOSTIC COMMAND
                        <1>       ;;;JC  short CTL_ERRX            ; DISPLAY ERROR MESSAGE IF BAD RETURN
                        <1>       ;;;jc  short POD_DONE ;22/12/2014
                        <1>       ;;jnc  short hdc_reset0
                        <1>       ;;loop hdc_dl
                        <1>       ;;; 27/12/2014
                        <1>       ;;stc
                        <1>       ;;retn
                        <1>       ;
                        <1> ;;hdc_reset0:
                        <1>       ; 18/01/2015
0000424D 8A0D[9C520100] <1>       mov    cl, [HF_NUM]
00004253 20C9           <1>       and    cl, cl
00004255 740E           <1>       jz     short POD_DONE
                        <1>       ;
00004257 B27F           <1>       mov    dl, 7Fh
                        <1> hdc_reset1:
00004259 FEC2           <1>       inc    dl
                        <1>       ;; 31/12/2015
                        <1>       ;;push dx
                        <1>       ;;push cx
                        <1>       ;;push ds
                        <1>       ;;sub  ax, ax
                        <1>       ;;mov  ds, ax
                        <1>       ;;MOV  AX, [TIMER_LOW]           ; GET START TIMER COUNTS
                        <1>       ;;pop  ds
                        <1>       ;;MOV  BX,AX
                        <1>       ;;ADD  AX,6*182           ; 60 SECONDS* 18.2
                        <1>       ;;MOV  CX,AX
                        <1>       ;;mov  word [wait_count], 0      ; 22/12/2014 (reset wait counter)
                        <1>       ;;
                        <1>       ;; 31/12/2014 - cancel HD_RESET_1
                        <1>       ;;CALL HD_RESET_1         ; SET UP DRIVE 0, (1,2,3)
                        <1>       ;;pop  cx
                        <1>       ;;pop  dx
                        <1>       ;;
                        <1>       ; 18/01/2015
0000425B B40D           <1>       mov    ah, 0Dh ; ALTERNATE RESET
                        <1>       ;int   13h
0000425D E8A4FFFFFF     <1>       call   int13h
00004262 E2F5           <1>       loop   hdc_reset1
00004264 F8             <1>       clc    ; 29/05/2016
                        <1> POD_DONE:
00004265 C3             <1>       RETn
                        <1>
                        <1> ;;-----     POD_ERROR
                        <1>
                        <1> ;;CTL_ERRX:
                        <1> ;     ;MOV   SI,OFFSET F1782     ; CONTROLLER ERROR
                        <1> ;     ;CALL  SET_FAIL            ; DO NOT IPL FROM DISK
                        <1> ;     ;CALL  E_MSG               ; DISPLAY ERROR AND SET (BP) ERROR FLAG
                        <1> ;     ;JMP   short POD_DONE
                        <1>
                        <1> ;;HD_RESET_1:
                        <1> ;;    ;PUSH  BX                  ; SAVE TIMER LIMITS
                        <1> ;;    ;PUSH  CX
                        <1> ;;RES_1: MOV AH,09H              ; SET DRIVE PARAMETERS
                        <1> ;;    INT    13H
                        <1> ;;    JC     short RES_2
```

```
3578                    <1> ;;    MOV   AH,11H              ; RECALIBRATE DRIVE
3579                    <1> ;;    INT   13H
3580                    <1> ;;    JNC   short RES_CK        ; DRIVE OK
3581                    <1> ;;RES_2: ;CALL     POD_TCHK           ; CHECK TIME OUT
3582                    <1> ;;    cmp   word [wait_count], 6*182 ; waiting time (in timer ticks)
3583                    <1> ;;                               ; (30 seconds)
3584                    <1> ;;    ;cmc
3585                    <1> ;;    ;JNC   short RES_1
3586                    <1> ;;    jb     short RES_1
3587                    <1> ;;;RES_FL: ;MOV    SI,OFFSET F1781     ; INDICATE DISK 1 FAILURE;
3588                    <1> ;;    ;TEST  DL,1
3589                    <1> ;;    ;JNZ   RES_E1
3590                    <1> ;;    ;MOV   SI,OFFSET F1780     ; INDICATE DISK 0 FAILURE
3591                    <1> ;;    ;CALL  SET_FAIL            ; DO NOT TRY TO IPL DISK 0
3592                    <1> ;;    ;JMP   SHORT RES_E1
3593                    <1> ;;RES_ER: ; 22/12/2014
3594                    <1> ;;RES_OK:
3595                    <1> ;;    ;POP   CX                  ; RESTORE TIMER LIMITS
3596                    <1> ;;    ;POP   BX
3597                    <1> ;;    RETn
3598                    <1> ;;
3599                    <1> ;;RES_RS: MOV      AH,00H              ; RESET THE DRIVE
3600                    <1> ;;    INT   13H
3601                    <1> ;;RES_CK: MOV      AH,08H              ; GET MAX CYLINDER,HEAD,SECTOR
3602                    <1> ;;    MOV   BL,DL                ; SAVE DRIVE CODE
3603                    <1> ;;    INT   13H
3604                    <1> ;;    JC    short RES_ER
3605                    <1> ;;    MOV   [NEC_STATUS],CX      ; SAVE MAX CYLINDER, SECTOR
3606                    <1> ;;    MOV   DL,BL                ; RESTORE DRIVE CODE
3607                    <1> ;;RES_3: MOV AX,0401H            ; VERIFY THE LAST SECTOR
3608                    <1> ;;    INT   13H
3609                    <1> ;;    JNC   short RES_OK         ; VERIFY OK
3610                    <1> ;;    CMP   AH,BAD_SECTOR        ; OK ALSO IF JUST ID READ
3611                    <1> ;;    JE    short RES_OK
3612                    <1> ;;    CMP   AH,DATA_CORRECTED
3613                    <1> ;;    JE    short RES_OK
3614                    <1> ;;    CMP   AH,BAD_ECC
3615                    <1> ;;    JE    short RES_OK
3616                    <1> ;;    ;CALL  POD_TCHK            ; CHECK FOR TIME OUT
3617                    <1> ;;    cmp   word [wait_count], 6*182 ; waiting time (in timer ticks)
3618                    <1> ;;                               ; (60 seconds)
3619                    <1> ;;    cmc
3620                    <1> ;;    JC    short RES_ER         ; FAILED
3621                    <1> ;;    MOV   CX,[NEC_STATUS]      ; GET SECTOR ADDRESS, AND CYLINDER
3622                    <1> ;;    MOV   AL,CL                ; SEPARATE OUT SECTOR NUMBER
3623                    <1> ;;    AND   AL,3FH
3624                    <1> ;;    DEC   AL                   ; TRY PREVIOUS ONE
3625                    <1> ;;    JZ    short RES_RS         ; WE'VE TRIED ALL SECTORS ON TRACK
3626                    <1> ;;    AND   CL,0C0H              ; KEEP CYLINDER BITS
3627                    <1> ;;    OR    CL,AL                ; MERGE SECTOR WITH CYLINDER BITS
3628                    <1> ;;    MOV   [NEC_STATUS],CX      ; SAVE CYLINDER, NEW SECTOR NUMBER
3629                    <1> ;;    JMP   short RES_3          ; TRY AGAIN
3630                    <1> ;;;RES_ER: MOV     SI,OFFSET F1791     ; INDICATE DISK 1 ERROR
3631                    <1> ;;    ;TEST  DL,1
3632                    <1> ;;    ;JNZ   short RES_E1
3633                    <1> ;;    ;MOV   SI,OFFSET F1790     ; INDICATE DISK 0 ERROR
3634                    <1> ;;;RES_E1:
3635                    <1> ;;    ;CALL  E_MSG               ; DISPLAY ERROR AND SET (BP) ERROR FLAG
3636                    <1> ;;;RES_OK:
3637                    <1> ;;    ;POP   CX                  ; RESTORE TIMER LIMITS
3638                    <1> ;;    ;POP   BX
3639                    <1> ;;    ;RETn
3640                    <1> ;
3641                    <1> ;;SET_FAIL:
3642                    <1> ;     ;MOV   AX,X*(CMOS_DIAG+NMI)      ; GET CMOS ERROR BYTE
3643                    <1> ;     ;CALL  CMOS_READ
3644                    <1> ;     ;OR    AL,HF_FAIL          ; SET DO NOT IPL FROM DISK FLAG
3645                    <1> ;     ;XCHG  AH,AL               ; SAVE IT
3646                    <1> ;     ;CALL  CMOS_WRITE          ; PUT IT OUT
3647                    <1> ;     ;RETn
3648                    <1> ;
3649                    <1> ;;POD_TCHK:                      ; CHECK FOR 30 SECOND TIME OUT
3650                    <1> ;     ;POP   AX                  ; SAVE RETURN
3651                    <1> ;     ;POP   CX                  ; GET TIME OUT LIMITS
3652                    <1> ;     ;POP   BX
3653                    <1> ;     ;PUSH  BX                  ; AND SAVE THEM AGAIN
3654                    <1> ;     ;PUSH  CX
3655                    <1> ;     ;PUSH  AX
3656                    <1> ;     ;push  ds
3657                    <1> ;     ;xor   ax, ax
3658                    <1> ;     ;mov   ds, ax              ; RESTORE RETURN
3659                    <1> ;     ;MOV   AX, [TIMER_LOW]           ; AX = CURRENT TIME
3660                    <1> ;     ;                          ; BX = START TIME
3661                    <1> ;     ;                          ; CX = END TIME
3662                    <1> ;     ;pop   ds
3663                    <1> ;     ;CMP   BX,CX
3664                    <1> ;     ;JB    short TCHK1         ; START < END
3665                    <1> ;     ;CMP   BX,AX
3666                    <1> ;     ;JB    short TCHKG         ; END < START < CURRENT
3667                    <1> ;     ;JMP   SHORT TCHK2         ; END, CURRENT < START
3668                    <1> ;;TCHK1: CMP AX,BX
3669                    <1> ;;    JB    short TCHKNG         ; CURRENT < START < END
3670                    <1> ;;TCHK2: CMP AX,CX
3671                    <1> ;;    JB    short TCHKG          ; START < CURRENT < END
3672                    <1> ;;                               ; OR CURRENT < END < START
3673                    <1> ;;TCHKNG: STC                          ; CARRY SET INDICATES TIME OUT
3674                    <1> ;;    RETn
3675                    <1> ;;TCHKG: CLC                     ; INDICATE STILL TIME
3676                    <1> ;;    RETn
3677                    <1> ;;
3678                    <1> ;;int_13h:
3679                    <1>
```

```
                                <1> ;----------------------------------------
                                <1> ;     FIXED DISK BIOS ENTRY POINT      :
                                <1> ;----------------------------------------
                                <1>
                                <1> ; 15/01/2017
                                <1> ; 14/01/2017
                                <1> ; 07/01/2017
                                <1> ; 02/01/2017
                                <1> ; 01/06/2016
                                <1> ; 16/05/2016, 27/05/2016, 28/05/2016, 29/05/2016
                                <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                                <1> int33h:  ; DISK I/O
                                <1>       ; 29/05/2016
00004266 80642408FE             <1>       and   byte [esp+8], 11111110b  ; clear carry bit of eflags register
                                <1>       ; 16/05/2016
0000426B 1E                     <1>       push  ds
0000426C 53                     <1>       push  ebx ; user's buffer address (virtual)
0000426D 66BB1000               <1>       mov   bx, KDATA ; System (Kernel's) data segment
00004271 8EDB                   <1>       mov   ds, bx
                                <1>
                                <1>       ;;15/01/2017
                                <1>       ; 14/01/2017
                                <1>       ; 02/01/2017
                                <1>       ;;mov  byte [intflg], 33h  ; disk io interrupt
                                <1>       ;pop  ebx
                                <1>       ;mov  [user_buffer], ebx
                                <1>
00004273 8F05[905F0100]         <1>       pop   dword [user_buffer] ; 01/06/2016
                                <1>
00004279 C605[C6580100]00       <1>       mov   byte [scount], 0 ; sector count for transfer
00004280 80FC03                 <1>       cmp   ah, 03h ; chs write
00004283 7744                   <1>       ja    short int33h_2
00004285 7407                   <1>       je    short int33h_0
00004287 80FC02                 <1>       cmp   ah, 02h ; chs read
0000428A 726A                   <1>       jb    short int33h_5
0000428C EB63                   <1>       jmp   short int33h_4
                                <1> int33h_0:
                                <1>       ; transfer user's buffer content to sector buffer
0000428E 51                     <1>       push  ecx
0000428F 0FB6C8                 <1>       movzx ecx, al
                                <1> int33h_1:
00004292 56                     <1>       push  esi
00004293 8B35[905F0100]         <1>       mov   esi, [user_buffer]
                                <1>       ; esi = user's buffer address (virtual, ebx)
00004299 57                     <1>       push  edi
0000429A 06                     <1>       push  es
0000429B 50                     <1>       push  eax
0000429C 66B81000               <1>       mov   ax, KDATA
000042A0 8EC0                   <1>       mov   es, ax
000042A2 BF00000700             <1>       mov   edi, Cluster_Buffer
000042A7 C1E109                 <1>       shl   ecx, 9 ; * 512
000042AA E821A60000             <1>       call  transfer_from_user_buffer
000042AF 58                     <1>       pop   eax
000042B0 07                     <1>       pop   es
000042B1 5F                     <1>       pop   edi
000042B2 5E                     <1>       pop   esi
000042B3 59                     <1>       pop   ecx
000042B4 7340                   <1>       jnc   short int33h_5
000042B6 8B1D[905F0100]         <1>       mov   ebx, [user_buffer] ; 01/06/2016
000042BC 1F                     <1>       pop   ds
                                <1>
                                <1>       ;;15/01/2017
                                <1>       ; 02/01/2017
                                <1>       ;cli
                                <1>       ;;mov  byte [ss:intflg], 0 ; 07/01/2017
                                <1>       ;
                                <1>       ; (*) 29/05/2016
                                <1>       ; (*) retf 4 ; skip eflags on stack
                                <1>
                                <1>       ; 29/05/2016 -set carry flag on stack-
                                <1>       ; [esp] = EIP
                                <1>       ; [esp+4] = CS
                                <1>       ; [esp+8] = E-FLAGS
000042BD 804C240801             <1>       or    byte [esp+8], 1  ; set carry bit of eflags register
                                <1>       ; [esp+12] = ESP (user)
                                <1>       ; [esp+16] = SS (User)
000042C2 B8FF000000             <1>       mov   eax, 0FFh ; Unknown error !?
                                <1>       ;iretd
000042C7 EB79                   <1>       jmp   short int33h_7  ; 07/01/2017
                                <1>
                                <1>       ; (*) 29/05/2016 - 'ref 4' intruction causes to stack fault
                                <1>       ; (OUTER-PRIVILEGE-LEVEL)
                                <1>       ; INTEL 80386 PROGRAMMER'S REFERENCE MANUAL 1986
                                <1>       ; // RETF instruction:
                                <1>       ;
                                <1>       ; IF OperandMode=32 THEN
                                <1>       ;    Load CS:EIP from stack;
                                <1>       ;    Set CS RPL to CPL;
                                <1>       ;    Increment eSP by 8 plus the immediate offset if it exists;
                                <1>       ;    Load SS:eSP from stack;
                                <1>       ; ELSE (* OperandMode=16 *)
                                <1>       ;    Load CS:IP from stack;
                                <1>       ;    Set CS RPL to CPL;
                                <1>       ;    Increment eSP by 4 plus the immediate offset if it exists;
                                <1>       ;    Load SS:eSP from stack;
                                <1>       ; FI;
                                <1>       ;
                                <1>       ; //
                                <1>
                                <1> int33h_2:
000042C9 80FC05                 <1>       cmp   ah, 05h ; format track
000042CC 770A                   <1>       ja    short int33h_3
```

```
3782 000042CE 7226                <1>       jb    short int33h_5
3783 000042D0 51                  <1>       push  ecx
3784 000042D1 B901000000          <1>       mov   ecx, 1
3785 000042D6 EBBA                <1>       jmp   short int33h_1
3786                              <1> int33h_3:
3787 000042D8 80FC1C              <1>       cmp   ah, 1Ch ; LBA write
3788 000042DB 7719                <1>       ja    short int33h_5
3789 000042DD 74AF                <1>       je    short int33h_0
3790 000042DF 80FC1B              <1>       cmp   ah, 1Bh ; LBA read
3791 000042E2 740D                <1>       je    short int33h_4
3792 000042E4 80FC08              <1>       cmp   ah, 08h ; get disk parameters
3793 000042E7 750D                <1>       jne   short int33h_5
3794                              <1>       ; 01/06/2016
3795 000042E9 8B1D[905F0100]      <1>       mov   ebx, [user_buffer] ; user's buffer address
3796 000042EF EB0A                <1>       jmp   short int33h_6
3797                              <1> int33h_4:
3798 000042F1 A2[C6580100]        <1>       mov   byte [scount], al ; <= 128 sectors
3799                              <1> int33h_5:
3800 000042F6 BB00000700          <1>       mov   ebx, Cluster_Buffer ; max. 65536 bytes
3801                              <1>                                 ; buf. addr: 70000h
3802                              <1>       ;mov  byte [ClusterBuffer_Valid], 0
3803                              <1> int33h_6:
3804 000042FB 1F                  <1>       pop   ds
3805 000042FC 9C                  <1>       pushfd
3806 000042FD 0E                  <1>       push  cs
3807 000042FE E84D000000          <1>       call  DISK_IO
3808 00004303 2E8B1D[905F0100]    <1>       mov   ebx, [CS:user_buffer] ; 01/06/2016
3809 0000430A 723D                <1>       jc    short int33h_9
3810                              <1>       ;
3811 0000430C 2E803D[C6580100]00  <1>       cmp   byte [CS:scount], 0
3812 00004314 762C                <1>       jna   short int33h_7
3813                              <1>       ; transfer sector buffer content to user's buffer
3814 00004316 06                  <1>       push  es
3815 00004317 1E                  <1>       push  ds
3816 00004318 50                  <1>       push  eax
3817 00004319 66B81000            <1>       mov   ax, KDATA
3818 0000431D 8ED8                <1>       mov   ds, ax
3819 0000431F 8EC0                <1>       mov   es, ax
3820 00004321 51                  <1>       push  ecx
3821 00004322 56                  <1>       push  esi
3822 00004323 57                  <1>       push  edi
3823 00004324 0FB60D[C6580100]    <1>       movzx ecx, byte [scount]
3824 0000432B C1E109              <1>       shl   ecx, 9 ; * 512 bytes
3825 0000432E 89DF                <1>       mov   edi, ebx ; user's buffer address
3826 00004330 BE00000700          <1>       mov   esi, Cluster_Buffer
3827 00004335 E84CA50000          <1>       call  transfer_to_user_buffer
3828 0000433A 5F                  <1>       pop   edi
3829 0000433B 5E                  <1>       pop   esi
3830 0000433C 59                  <1>       pop   ecx
3831 0000433D 58                  <1>       pop   eax
3832 0000433E 1F                  <1>       pop   ds
3833 0000433F 07                  <1>       pop   es
3834 00004340 7202                <1>       jc    short int33h_8
3835                              <1> int33h_7:
3836 00004342 FA                  <1>       cli
3837                              <1>       ;;15/01/2017
3838                              <1>       ;;mov  byte [ss:intflg], 0 ; 07/01/2017
3839                              <1>       ; cf = 0  ; use eflags which is in stack
3840 00004343 CF                  <1>       iretd
3841                              <1> int33h_8:
3842 00004344 B8FF000000          <1>       mov   eax, 0FFh ; Unknown error !?
3843                              <1> int33h_9:
3844                              <1>       ; cf = 1
3845                              <1>
3846                              <1>       ; (*) 29/05/2016
3847                              <1>       ; (*) retf 4 ; skip eflags on stack
3848                              <1>       ; Note: This 'retf 4' was wrong, -it was causing
3849                              <1>       ;       to stack errors in ring 3-
3850                              <1>       ;     POP sequence of 'retf 4' is as
3851                              <1>       ;       "eip, cs, eflags, esp, ss, +4 bytes"
3852                              <1>         ;       it is not as "eip, cs, +4 bytes, esp, ss" !
3853                              <1>
3854                              <1>       ; 29/05/2016 -set carry flag on stack-
3855 00004349 804C240801          <1>       or    byte [esp+8], 1  ; set carry bit of eflags register
3856                              <1>       ;iretd
3857 0000434E EBF2                <1>       jmp   short int33h_7 ; 07/01/2017
3858                              <1>
3859                              <1> ; 29/05/2016
3860                              <1> ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)
3861                              <1>
3862                              <1> DISK_IO:
3863 00004350 80FA80              <1>       CMP   DL,80H              ; TEST FOR FIXED DISK DRIVE
3864                              <1>       ;JAE  short A1            ; YES, HANDLE HERE
3865                              <1>       ;;;INT 40H                ; DISKETTE HANDLER
3866                              <1>       ;;call int40h
3867 00004353 0F8222F0FFFF        <1>       jb    DISKETTE_IO_1
3868                              <1> ;RET_2:
3869                              <1>       ;RETf 2                   ; BACK TO CALLER
3870                              <1> ;     retf  4
3871                              <1> A1:
3872 00004359 FB                  <1>       STI                       ; ENABLE INTERRUPTS
3873                              <1>       ;; 04/01/2015
3874                              <1>       ;OR   AH,AH
3875                              <1>       ;JNZ  short A2
3876                              <1>       ;INT  40H                 ; RESET NEC WHEN AH=0
3877                              <1>       ;SUB  AH,AH
3878 0000435A 80FA83              <1>       CMP   DL,(80H + S_MAX_FILE - 1)
3879                              <1>       ;JA   short RET_2
3880 0000435D 7616                <1>       jna   short _A0
3881                              <1>       ; 29/05/2016
3882 0000435F 1E                  <1>       push  ds
3883 00004360 6650                <1>       push  ax
```

```
3884 00004362 66B81000            <1>       mov    ax, KDATA
3885 00004366 8ED8                <1>       mov    ds, ax
3886 00004368 6658                <1>       pop    ax
3887 0000436A B4AA                <1>         mov     ah, 0AAh        ; Hard disk drive not ready !
3888                              <1>                                 ; (Programmer's guide to AMIBIOS, 1992)
3889 0000436C 8825[9B520100]      <1>       mov     byte [DISK_STATUS1], ah
3890 00004372 1F                  <1>       pop    ds
3891 00004373 EB38                <1>       jmp    short RET_2
3892                              <1> _A0:
3893                              <1>       ; 18/01/2015
3894 00004375 08E4                <1>       or     ah,ah
3895 00004377 743A                <1>       jz     short A4
3896 00004379 80FC0D              <1>       cmp    ah, 0Dh      ; Alternate reset
3897 0000437C 7504                <1>       jne    short A2
3898 0000437E 28E4                <1>       sub    ah,ah  ; Reset
3899 00004380 EB31                <1>       jmp    short A4
3900                              <1> A2:
3901 00004382 80FC08              <1>       CMP    AH,08H              ; GET PARAMETERS IS A SPECIAL CASE
3902                              <1>       ;JNZ   short A3
3903                              <1>         ;JMP    GET_PARM_N
3904 00004385 0F8431030000        <1>       je     GET_PARM_N
3905 0000438B 80FC15              <1> A3:   CMP    AH,15H              ; READ DASD TYPE IS ALSO
3906                              <1>       ;JNZ   short A4
3907                              <1>         ;JMP    READ_DASD_TYPE
3908 0000438E 0F84DA020000        <1>         je      READ_DASD_TYPE
3909                              <1>       ; 02/02/2015
3910 00004394 80FC1D              <1>       cmp    ah, 1Dh                    ;(Temporary for Retro UNIX 386 v1)
3911                              <1>       ; 12/01/2015
3912 00004397 F5                  <1>       cmc
3913 00004398 7319                <1>       jnc    short A4
3914                              <1> int33h_bad_cmd:
3915                              <1>       ; 16/05/2016
3916                              <1>       ; 30/01/2015
3917                              <1>       ; 29/05/2016
3918 0000439A 1E                  <1>       push   ds
3919 0000439B 6650                <1>       push   ax
3920 0000439D 66B81000            <1>       mov    ax, KDATA
3921 000043A1 8ED8                <1>       mov    ds, ax
3922 000043A3 6658                <1>       pop    ax
3923 000043A5 B401                <1>       mov    ah, BAD_CMD
3924 000043A7 8825[9B520100]      <1>       mov     [DISK_STATUS1], ah ; BAD_CMD  ; COMMAND ERROR
3925                              <1>         ;jmp short RET_2
3926                              <1> RET_2:
3927                              <1>       ; (*) 29/05/2016
3928                              <1>       ; (*) retf 4
3929 000043AD 804C240801          <1>       or     byte [esp+8], 1 ; set carry bit of eflags register
3930 000043B2 CF                  <1>       iretd
3931                              <1> A4:                               ; SAVE REGISTERS DURING OPERATION
3932 000043B3 C8080000            <1>       ENTER  8,0                  ; SAVE (BP) AND MAKE ROOM FOR @CMD_BLOCK
3933 000043B7 53                  <1>       PUSH   eBX                  ;  IN THE STACK, THE COMMAND BLOCK IS:
3934 000043B8 51                  <1>       PUSH   eCX                  ;   @CMD_BLOCK == BYTE PTR [BP]-8
3935 000043B9 52                  <1>       PUSH   eDX
3936 000043BA 1E                  <1>       PUSH   DS
3937 000043BB 06                  <1>       PUSH   ES
3938 000043BC 56                  <1>       PUSH   eSI
3939 000043BD 57                  <1>       PUSH   eDI
3940                              <1>       ;;04/01/2015
3941                              <1>       ;;OR   AH,AH                ; CHECK FOR RESET
3942                              <1>       ;;JNZ  short A5
3943                              <1>       ;;MOV  DL,80H               ; FORCE DRIVE 80 FOR RESET
3944                              <1> ;;A5:
3945                              <1>       ;push  cs
3946                              <1>       ;pop   ds
3947                              <1>       ; 21/02/2015
3948 000043BE 6650                <1>       push   ax
3949 000043C0 66B81000            <1>       mov    ax, KDATA
3950 000043C4 8ED8                <1>       mov    ds, ax
3951 000043C6 8EC0                <1>       mov    es, ax
3952 000043C8 6658                <1>       pop    ax
3953 000043CA E88D000000          <1>       CALL   DISK_IO_CONT         ; PERFORM THE OPERATION
3954                              <1>       ;;CALL DDS                  ; ESTABLISH SEGMENT
3955 000043CF 8A25[9B520100]      <1>       MOV    AH,[DISK_STATUS1]    ; GET STATUS FROM OPERATION
3956                              <1>       ;(*) CMP AH,1               ; SET THE CARRY FLAG TO INDICATE
3957                              <1>       ;(*) CMC                    ; SUCCESS OR FAILURE
3958 000043D5 5F                  <1>       POP    eDI                  ; RESTORE REGISTERS
3959 000043D6 5E                  <1>       POP    eSI
3960 000043D7 07                  <1>         POP     ES
3961 000043D8 1F                  <1>         POP     DS
3962 000043D9 5A                  <1>       POP    eDX
3963 000043DA 59                  <1>       POP    eCX
3964 000043DB 5B                  <1>       POP    eBX
3965 000043DC C9                  <1>       LEAVE                       ; ADJUST (SP) AND RESTORE (BP)
3966                              <1>       ;RETf  2                    ; THROW AWAY SAVED FLAGS
3967                              <1>       ; (*) 29/05/2016
3968                              <1>       ; (*) retf 4
3969 000043DD 80FC01              <1>       cmp    ah, 1
3970 000043E0 7205                <1>       jc     short _A5
3971 000043E2 804C240801          <1>       or     byte [esp+8], 1 ; set carry bit of eflags register
3972                              <1> _A5:
3973 000043E7 CF                  <1>       iretd
3974                              <1>
3975                              <1> ; 21/02/2015
3976                              <1> ;       dw --> dd
3977                              <1> D1:                               ; FUNCTION TRANSFER TABLE
3978 000043E8 [AA450000]          <1>       dd     DISK_RESET           ; 000H
3979 000043EC [21460000]          <1>       dd     RETURN_STATUS        ; 001H
3980 000043F0 [2E460000]          <1>       dd     DISK_READ            ; 002H
3981 000043F4 [37460000]          <1>       dd     DISK_WRITE           ; 003H
3982 000043F8 [40460000]          <1>       dd     DISK_VERF            ; 004H
3983 000043FC [58460000]          <1>       dd     FMT_TRK              ; 005H
3984 00004400 [A0450000]          <1>       dd     BAD_COMMAND          ; 006H FORMAT BAD SECTORS
3985 00004404 [A0450000]          <1>       dd     BAD_COMMAND          ; 007H FORMAT DRIVE
```

```
3986 00004408 [A0450000]          <1>       dd     BAD_COMMAND         ; 008H RETURN PARAMETERS
3987 0000440C [43470000]          <1>       dd     INIT_DRV            ; 009H
3988 00004410 [A2470000]          <1>       dd     RD_LONG             ; 00AH
3989 00004414 [AB470000]          <1>       dd     WR_LONG             ; 00BH
3990 00004418 [B4470000]          <1>       dd     DISK_SEEK           ; 00CH
3991 0000441C [AA450000]          <1>       dd     DISK_RESET          ; 00DH
3992 00004420 [A0450000]          <1>       dd     BAD_COMMAND         ; 00EH READ BUFFER
3993 00004424 [A0450000]          <1>       dd     BAD_COMMAND         ; 00FH WRITE BUFFER
3994 00004428 [DC470000]          <1>       dd     TST_RDY             ; 010H
3995 0000442C [00480000]          <1>       dd     HDISK_RECAL         ; 011H
3996 00004430 [A0450000]          <1>       dd     BAD_COMMAND         ; 012H MEMORY DIAGNOSTIC
3997 00004434 [A0450000]          <1>       dd     BAD_COMMAND         ; 013H DRIVE DIAGNOSTIC
3998 00004438 [36480000]          <1>       dd     CTLR_DIAGNOSTIC     ; 014H CONTROLLER DIAGNOSTIC
3999                              <1>       ; 02/02/2015 (Temporary - Retro UNIX 386 v1 - DISK I/O test)
4000 0000443C [A0450000]          <1>       dd     BAD_COMMAND         ; 015h
4001 00004440 [A0450000]          <1>       dd     BAD_COMMAND         ; 016h
4002 00004444 [A0450000]          <1>       dd     BAD_COMMAND         ; 017h
4003 00004448 [A0450000]          <1>       dd     BAD_COMMAND         ; 018h
4004 0000444C [A0450000]          <1>       dd     BAD_COMMAND         ; 019h
4005 00004450 [A0450000]          <1>       dd     BAD_COMMAND         ; 01Ah
4006 00004454 [2E460000]          <1>       dd     DISK_READ           ; 01Bh ; LBA read
4007 00004458 [37460000]          <1>       dd     DISK_WRITE          ; 01Ch ; LBA write
4008                              <1> D1L     EQU    $ - D1
4009                              <1>
4010                              <1> DISK_IO_CONT:
4011                              <1>       ;;CALL DDS                 ; ESTABLISH SEGMENT
4012 0000445C 80FC01              <1>       CMP    AH,01H              ; RETURN STATUS
4013                              <1>       ;;JNZ  short SU0
4014                              <1>         ;;JMP    RETURN_STATUS
4015 0000445F 0F84BC010000        <1>       je     RETURN_STATUS
4016                              <1> SU0:
4017 00004465 C605[9B520100]00    <1>       MOV    byte [DISK_STATUS1],0     ; RESET THE STATUS INDICATOR
4018                              <1>       ;;PUSH BX                  ; SAVE DATA ADDRESS
4019                              <1>       ;mov   si, bx ;; 14/02/2015
4020 0000446C 89DE                <1>       mov    esi, ebx ; 21/02/2015
4021 0000446E 8A1D[9C520100]      <1>       MOV    BL,[HF_NUM]         ; GET NUMBER OF DRIVES
4022                              <1>       ;; 04/01/2015
4023                              <1>       ;;PUSH AX
4024 00004474 80E27F              <1>       AND    DL,7FH              ; GET DRIVE AS 0 OR 1
4025                              <1>                                  ; (get drive number as 0 to 3)
4026 00004477 38D3                <1>       CMP    BL,DL
4027                              <1>         ;;JBE   BAD_COMMAND_POP         ; INVALID DRIVE
4028 00004479 0F8621010000        <1>         jbe     BAD_COMMAND ;; 14/02/2015
4029                              <1>         ;
4030                              <1>       ;;03/01/2015
4031 0000447F 29DB                <1>       sub    ebx, ebx
4032 00004481 88D3                <1>       mov    bl, dl
4033                              <1>       ;sub   bh, bh
4034 00004483 883D[B0520100]      <1>       mov    [LBAMode], bh       ; 0
4035                              <1>       ;;test byte [bx+hd0_type], 1     ; LBA ready ?
4036                              <1>       ;test  byte [ebx+hd0_type], 1
4037                              <1>       ;jz    short su1           ; no
4038                              <1>       ;inc   byte [LBAMode]
4039                              <1> ;su1:
4040                              <1>       ; 21/02/2015 (32 bit modification)
4041                              <1>       ;04/01/2015
4042 00004489 6650                <1>       push   ax ; ***
4043                              <1>       ;PUSH  ES ; **
4044 0000448B 6652                <1>       PUSH   DX ; *
4045 0000448D 6650                <1>       push   ax
4046 0000448F E888060000          <1>       CALL   GET_VEC             ; GET DISK PARAMETERS
4047                              <1>       ; 02/02/2015
4048                              <1>       ;mov   ax, [ES:BX+16] ; I/O port base address (1F0h, 170h)
4049 00004494 668B4310            <1>       mov    ax, [ebx+16]
4050 00004498 66A3[E85C0000]      <1>       mov    [HF_PORT], ax
4051                              <1>       ;mov   dx, [ES:BX+18] ; control port address (3F6h, 376h)
4052 0000449E 668B5312            <1>       mov    dx, [ebx+18]
4053 000044A2 668915[EA5C0000]    <1>       mov    [HF_REG_PORT], dx
4054                              <1>       ;mov   al, [ES:BX+20] ; head register upper nibble (A0h,B0h,E0h,F0h)
4055 000044A9 8A4314              <1>       mov    al, [ebx+20]
4056                              <1>       ; 23/02/2015
4057 000044AC A840                <1>       test   al, 40h       ; LBA bit (bit 6)
4058 000044AE 7406                <1>       jz     short su1
4059 000044B0 FE05[B0520100]      <1>       inc    byte [LBAMode] ; 1
4060                              <1> su1:
4061 000044B6 C0E804              <1>       shr    al, 4
4062 000044B9 2401                <1>       and    al, 1
4063 000044BB A2[EC5C0000]        <1>       mov    [hf_m_s], al
4064                              <1>       ;
4065                              <1>       ; 03/01/2015
4066                              <1>       ;MOV   AL,byte [ES:BX+8]   ; GET CONTROL BYTE MODIFIER
4067 000044C0 8A4308              <1>       mov    al, [ebx+8]
4068                              <1>       ;MOV   DX,[HF_REG_PORT]    ; Device Control register
4069 000044C3 EE                  <1>       OUT    DX,AL               ; SET EXTRA HEAD OPTION
4070                              <1>                                  ; Control Byte:  (= 08h, here)
4071                              <1>                                  ; bit 0 - 0
4072                              <1>                                  ; bit 1 - nIEN (1 = disable irq)
4073                              <1>                                  ; bit 2 - SRST (software RESET)
4074                              <1>                                  ; bit 3 - use extra heads (8 to 15)
4075                              <1>                                  ;         -always set to 1-
4076                              <1>                                  ; (bits 3 to 7 are reserved
4077                              <1>                                  ;          for ATA devices)
4078 000044C4 8A25[9D520100]      <1>       MOV    AH,[CONTROL_BYTE]   ; SET EXTRA HEAD OPTION IN
4079 000044CA 80E4C0              <1>       AND    AH,0C0H             ; CONTROL BYTE
4080 000044CD 08C4                <1>       OR     AH,AL
4081 000044CF 8825[9D520100]      <1>       MOV    [CONTROL_BYTE],AH
4082                              <1>       ; 04/01/2015
4083 000044D5 6658                <1>       pop    ax
4084 000044D7 665A                <1>       pop    dx ; * ;; 14/02/2015
4085 000044D9 20E4                <1>       and    ah, ah ; Reset function ?
4086 000044DB 7507                <1>       jnz    short su2
4087                              <1>       ;;pop  dx ; * ;; 14/02/2015
```

```
                                <1>       ;pop   es ; **
000044DD 6658                   <1>       pop    ax ; ***
                                <1>       ;;pop  bx
000044DF E9C6000000             <1>         jmp     DISK_RESET
                                <1> su2:
000044E4 803D[B0520100]00       <1>       cmp    byte [LBAMode], 0
000044EB 7661                   <1>       jna    short su3
                                <1>       ;
                                <1>       ; 02/02/2015 (LBA read/write function calls)
000044ED 80FC1B                 <1>       cmp    ah, 1Bh
000044F0 720B                   <1>       jb     short lbarw1
000044F2 80FC1C                 <1>       cmp    ah, 1Ch
000044F5 775C                   <1>       ja     short invldfnc
                                <1>       ;;pop  dx ; * ; 14/02/2015
                                <1>       ;mov   ax, cx ; Lower word of LBA address (bits 0-15)
000044F7 89C8                   <1>       mov    eax, ecx ; LBA address (21/02/2015)
                                <1>       ;; 14/02/2015
000044F9 88D1                   <1>       mov    cl, dl ; 14/02/2015
                                <1>       ;;mov  dx, bx
                                <1>       ;mov   dx, si ; higher word of LBA address (bits 16-23)
                                <1>       ;;mov  bx, di
                                <1>       ;mov   si, di ; Buffer offset
000044FB EB31                   <1>       jmp    short lbarw2
                                <1> lbarw1:
                                <1>       ; convert CHS to LBA
                                <1>       ;
                                <1>       ; LBA calculation - AWARD BIOS - 1999 - AHDSK.ASM
                                <1>       ; LBA = "# of Heads" * Sectors/Track * Cylinder + Head * Sectors/Track
                                <1>       ;      + Sector - 1
000044FD 6652                   <1>       push   dx ; * ;; 14/02/2015
                                <1>       ;xor   dh, dh
000044FF 31D2                   <1>       xor    edx, edx
00004501 8A530E                 <1>       ;mov   dl, [ES:BX+14]      ; sectors per track (logical)
                                <1>       mov    dl, [ebx+14]
00004504 31C0                   <1>       ;xor   ah, ah
                                <1>       xor    eax, eax
00004506 8A4302                 <1>       ;mov   al, [ES:BX+2]; heads (logical)
                                <1>       mov    al, [ebx+2]
00004509 FEC8                   <1>       dec    al
0000450B 6640                   <1>       inc    ax           ; 0 =  256
0000450D 66F7E2                 <1>       mul    dx
                                <1>              ; AX = # of Heads" * Sectors/Track
00004510 6689CA                 <1>       mov    dx, cx
                                <1>       ;and   cx, 3Fh       ; sector  (1 to 63)
00004513 83E13F                 <1>       and    ecx, 3fh
00004516 86D6                   <1>       xchg   dl, dh
00004518 C0EE06                 <1>       shr    dh, 6
                                <1>              ; DX = cylinder (0 to 1023)
                                <1>       ;mul   dx
                                <1>              ; DX:AX = # of Heads" * Sectors/Track * Cylinder
0000451B F7E2                   <1>       mul    edx
0000451D FEC9                   <1>       dec    cl  ; sector - 1
                                <1>       ;add   ax, cx
                                <1>       ;adc   dx, 0
                                <1>              ; DX:AX = # of Heads" * Sectors/Track * Cylinder + Sector -1
0000451F 01C8                   <1>       add    eax, ecx
00004521 6659                   <1>       pop    cx ; * ; ch = head, cl = drive number (zero based)
                                <1>       ;push  dx
                                <1>       ;push  ax
00004523 50                     <1>       push   eax
00004524 8A430E                 <1>       ;mov   al, [ES:BX+14]      ; sectors per track (logical)
                                <1>       mov    al, [ebx+14]
00004527 F6E5                   <1>       mul    ch
                                <1>              ;  AX = Head * Sectors/Track
00004529 6699                   <1>         cwd
                                <1>       ;pop   dx
0000452B 5A                     <1>       pop    edx
                                <1>       ;add   ax, dx
                                <1>       ;pop   dx
                                <1>       ;adc   dx, 0 ; add carry bit
0000452C 01D0                   <1>       add    eax, edx
                                <1> lbarw2:
0000452E 29D2                   <1>       sub    edx, edx ; 21/02/2015
00004530 88CA                   <1>       mov    dl, cl ; 21/02/2015
00004532 C645F800               <1>         mov     byte [CMD_BLOCK], 0 ; Features Register
                                <1>                             ; NOTE: Features register (1F1h, 171h)
                                <1>                             ; is not used for ATA device R/W functions.
                                <1>                             ; It is old/obsolete 'write precompensation'
                                <1>                             ; register and error register
                                <1>                             ; for old ATA/IDE devices.
                                <1>       ; 18/01/2014
                                <1>       ;mov   ch, [hf_m_s] ; Drive 0 (master) or 1 (slave)
00004536 8A0D[EC5C0000]         <1>       mov    cl, [hf_m_s]
                                <1>       ;shl   ch, 4        ; bit 4 (drive bit)
                                <1>       ;or    ch, 0E0h     ; bit 5 = 1
                                <1>                             ; bit 6 = 1 = LBA mode
                                <1>                             ; bit 7 = 1
0000453C 80C90E                 <1>       or     cl, 0Eh ; 1110b
                                <1>       ;and   dh, 0Fh             ; LBA byte 4 (bits 24 to 27)
0000453F 25FFFFFF0F             <1>       and    eax, 0FFFFFFFh
00004544 C1E11C                 <1>       shl    ecx, 28 ; 21/02/2015
                                <1>       ;or    dh, ch
00004547 09C8                   <1>       or     eax, ecx
                                <1>       ;;mov  [CMD_BLOCK+2], al ; LBA byte 1 (bits 0 to 7)
                                <1>                               ; (Sector Number Register)
                                <1>       ;;mov  [CMD_BLOCK+3], ah ; LBA byte 2 (bits 8 to 15)
                                <1>                               ; (Cylinder Low Register)
                                <1>       ;mov   [CMD_BLOCK+2], ax ; LBA byte 1, 2
                                <1>       ;mov   [CMD_BLOCK+4], dl ; LBA byte 3 (bits 16 to 23)
                                <1>                               ; (Cylinder High Register)
                                <1>       ;;mov  [CMD_BLOCK+5], dh ; LBA byte 4 (bits 24 to 27)
                                <1>                               ; (Drive/Head Register)
```

```
                                <1>
                                <1>       ;mov   [CMD_BLOCK+4], dx ; LBA byte 4, LBA & DEV select bits
00004549 8945FA                 <1>       mov    [CMD_BLOCK+2], eax ; 21/02/2015
                                <1>       ;14/02/2015
                                <1>       ;mov   dl, cl ; Drive number (INIT_DRV)
0000454C EB38                   <1>       jmp    short su4
                                <1> su3:
                                <1>       ; 02/02/2015
                                <1>       ; (Temporary functions 1Bh & 1Ch are not valid for CHS mode)
0000454E 80FC14                 <1>       cmp    ah, 14h
00004551 7604                   <1>       jna    short chsfnc
                                <1> invldfnc:
                                <1>         ; 14/02/2015
                                <1>       ;pop   es ; **
                                <1>         pop     ax ; ***
                                <1>         ;jmp     short BAD_COMMAND_POP
00004555 EB49                   <1>         jmp     short BAD_COMMAND
                                <1> chsfnc:
                                <1>       ;MOV   AX,[ES:BX+5]       ; GET WRITE PRE-COMPENSATION CYLINDER
00004557 668B4305               <1>       mov    ax, [ebx+5]
0000455B 66C1E802               <1>       SHR    AX,2
0000455F 8845F8                 <1>       MOV    [CMD_BLOCK],AL
                                <1>       ;;MOV  AL,[ES:BX+8]       ; GET CONTROL BYTE MODIFIER
                                <1>       ;;PUSH DX
                                <1>       ;;MOV  DX,[HF_REG_PORT]
                                <1>       ;;OUT  DX,AL              ; SET EXTRA HEAD OPTION
                                <1>       ;;POP  DX ; *
                                <1>       ;;POP  ES ; **
                                <1>       ;;MOV  AH,[CONTROL_BYTE]  ; SET EXTRA HEAD OPTION IN
                                <1>       ;;AND  AH,0C0H            ; CONTROL BYTE
                                <1>       ;;OR   AH,AL
                                <1>       ;;MOV  [CONTROL_BYTE],AH
                                <1>       ;
00004562 88C8                   <1>       MOV    AL,CL              ; GET SECTOR NUMBER
00004564 243F                   <1>       AND    AL,3FH
00004566 8845FA                 <1>       MOV    [CMD_BLOCK+2],AL
00004569 886DFB                 <1>       MOV    [CMD_BLOCK+3],CH   ; GET CYLINDER NUMBER
0000456C 88C8                   <1>       MOV    AL,CL
0000456E C0E806                 <1>       SHR    AL,6
00004571 8845FC                 <1>       MOV    [CMD_BLOCK+4],AL   ; CYLINDER HIGH ORDER 2 BITS
                                <1>       ;;05/01/2015
                                <1>       ;;MOV  AL,DL              ; DRIVE NUMBER
00004574 A0[EC5C0000]           <1>       mov    al, [hf_m_s]
00004579 C0E004                 <1>       SHL    AL,4
0000457C 80E60F                 <1>       AND    DH,0FH             ; HEAD NUMBER
0000457F 08F0                   <1>       OR     AL,DH
                                <1>       ;OR    AL,80H or 20H
00004581 0CA0                   <1>       OR     AL,80h+20h         ; ECC AND 512 BYTE SECTORS
00004583 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
00004586 6658                   <1>         pop     ax ; ***
00004588 8845F9                 <1>         mov     [CMD_BLOCK+1], al
0000458B 29DB                   <1>         sub  ebx, ebx
0000458D 88E3                   <1>       mov     bl, ah
                                <1>         ;xor     bh, bh
                                <1>         ;sal     bx, 1
0000458F 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
00004593 83FB74                 <1>       cmp    ebx, D1L
00004596 7308                   <1>       jnb    short BAD_COMMAND
                                <1>         ;xchg    bx, si
00004598 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]
0000459A FFA6[E8430000]         <1>       jmp    dword [esi+D1]
                                <1> ;;BAD_COMMAND_POP:
                                <1> ;;    POP    AX
                                <1> ;;    POP    BX
                                <1> BAD_COMMAND:
000045A0 C605[9B520100]01       <1>         MOV     byte [DISK_STATUS1],BAD_CMD  ; COMMAND ERROR
000045A7 B000                   <1>       MOV    AL,0
000045A9 C3                     <1>       RETn
                                <1>
                                <1> ;---------------------------------------
                                <1> ;     RESET THE DISK SYSTEM  (AH=00H) :
                                <1> ;---------------------------------------
                                <1>
                                <1> ; 18-1-2015 : one controller reset (not other one)
```

```
4292                                    <1>
4293                                    <1> DISK_RESET:
4294 000045AA FA                        <1>         CLI
4295 000045AB E4A1                      <1>         IN      AL,INTB01           ; GET THE MASK REGISTER
4296                                    <1>         ;JMP    $+2
4297                                    <1>         IODELAY
4297 000045AD EB00                      <2>  jmp short $+2
4297 000045AF EB00                      <2>  jmp short $+2
4298                                    <1>         ;AND    AL,0BFH             ; ENABLE FIXED DISK INTERRUPT
4299 000045B1 243F                      <1>         and     al,3Fh              ; 22/12/2014 (IRQ 14 & IRQ 15)
4300 000045B3 E6A1                      <1>         OUT     INTB01,AL
4301 000045B5 FB                        <1>         STI                         ; START INTERRUPTS
4302                                    <1>         ; 14/02/2015
4303 000045B6 6689D7                    <1>         mov     di, dx
4304                                    <1>         ; 04/01/2015
4305                                    <1>         ;xor    di,di
4306                                    <1> drst0:
4307 000045B9 B004                      <1>         MOV     AL,04H  ; bit 2 - SRST
4308                                    <1>         ;MOV    DX,HF_REG_PORT
4309 000045BB 668B15[EA5C0000]          <1>         MOV     DX,[HF_REG_PORT]
4310 000045C2 EE                        <1>         OUT     DX,AL               ; RESET
4311                                    <1> ;       MOV     CX,10               ; DELAY COUNT
4312                                    <1> ;DRD:   DEC     CX
4313                                    <1> ;       JNZ     short DRD           ; WAIT 4.8 MICRO-SEC
4314                                    <1>         ;mov    cx,2                ; wait for 30 micro seconds
4315 000045C3 B902000000                <1>           mov   ecx, 2 ; 21/02/2015
4316 000045C8 E820D8FFFF                <1>         call    WAITF                   ; (Award Bios 1999 - WAIT_REFRESH,
4317                                    <1>                                           ; 40 micro seconds)
4318 000045CD A0[9D520100]              <1>         mov     al,[CONTROL_BYTE]
4319 000045D2 240F                      <1>         AND     AL,0FH              ; SET HEAD OPTION
4320 000045D4 EE                        <1>         OUT     DX,AL               ; TURN RESET OFF
4321 000045D5 E838040000                <1>         CALL    NOT_BUSY
4322 000045DA 7515                      <1>         JNZ     short DRERR         ; TIME OUT ON RESET
4323 000045DC 668B15[E85C0000]          <1>         MOV     DX,[HF_PORT]
4324 000045E3 FEC2                      <1>         inc     dl  ; HF_PORT+1
4325                                    <1>         ; 02/01/2015 - Award BIOS 1999 - AHDSK.ASM
4326                                    <1>           ;mov     cl, 10
4327 000045E5 B90A000000                <1>           mov     ecx, 10 ; 21/02/2015
4328                                    <1> drst1:
4329 000045EA EC                        <1>         IN      AL,DX               ; GET RESET STATUS
4330 000045EB 3C01                      <1>         CMP     AL,1
4331                                    <1>         ; 04/01/2015
4332 000045ED 740A                      <1>         jz      short drst2
4333                                    <1>         ;JNZ    short DRERR         ; BAD RESET STATUS
4334                                    <1>                 ; Drive/Head Register - bit 4
4335 000045EF E2F9                      <1>         loop    drst1
4336                                    <1> DRERR:
4337 000045F1 C605[9B520100]05          <1>         MOV     byte [DISK_STATUS1],BAD_RESET ; CARD FAILED
4338 000045F8 C3                        <1>         RETn
4339                                    <1> drst2:
4340                                    <1>         ; 14/02/2015
4341 000045F9 6689FA                    <1>         mov     dx,di
4342                                    <1> ;drst3:
4343                                    <1> ;       ; 05/01/2015
4344                                    <1> ;       shl     di,1
4345                                    <1> ;       ; 04/01/2015
4346                                    <1> ;       mov     ax,[di+hd_cports]
4347                                    <1> ;       cmp     ax,[HF_REG_PORT]
4348                                    <1> ;       je      short drst4
4349                                    <1> ;       mov     [HF_REG_PORT], ax
4350                                    <1> ;       ; 03/01/2015
4351                                    <1> ;       mov     ax,[di+hd_ports]
4352                                    <1> ;         mov     [HF_PORT], ax
4353                                    <1> ;       ; 05/01/2014
4354                                    <1> ;       shr     di,1
4355                                    <1> ;       ; 04/01/2015
4356                                    <1> ;       jmp     short drst0  ; reset other controller
4357                                    <1> ;drst4:
4358                                    <1> ;       ; 05/01/2015
4359                                    <1> ;       shr     di,1
4360                                    <1> ;       mov     al,[di+hd_dregs]
4361                                    <1> ;       and     al,10h ; bit 4 only
4362                                    <1> ;       shr     al,4 ; bit 4  -> bit 0
4363                                    <1> ;       mov     [hf_m_s], al ; (0 = master, 1 = slave)
4364                                    <1>         ;
4365 000045FC A0[EC5C0000]              <1>         mov     al, [hf_m_s] ; 18/01/2015
4366 00004601 A801                      <1>         test    al,1
4367                                    <1> ;       jnz     short drst6
4368 00004603 7516                      <1>           jnz     short drst4
4369 00004605 8065FDEF                  <1>         AND     byte [CMD_BLOCK+5],0EFH ; SET TO DRIVE 0
4370                                    <1> ;drst5:
4371                                    <1> drst3:
4372 00004609 E835010000                <1>         CALL    INIT_DRV            ; SET MAX HEADS
4373                                    <1>         ;mov    dx,di
4374 0000460E E8ED010000                <1>         CALL    HDISK_RECAL         ; RECAL TO RESET SEEK SPEED
4375                                    <1>         ; 04/01/2014
4376                                    <1> ;       inc     di
4377                                    <1> ;       mov     dx,di
4378                                    <1> ;       cmp     dl,[HF_NUM]
4379                                    <1> ;       jb      short drst3
4380                                    <1> ;DRE:
4381 00004613 C605[9B520100]00          <1>         MOV     byte [DISK_STATUS1],0     ; IGNORE ANY SET UP ERRORS
4382 0000461A C3                        <1>         RETn
4383                                    <1> ;drst6:
4384                                    <1> drst4:              ; Drive/Head Register - bit 4
4385 0000461B 804DFD10                  <1>         OR      byte [CMD_BLOCK+5],010H ; SET TO DRIVE 1
4386                                    <1>           ;jmp    short drst5
4387 0000461F EBE8                      <1>           jmp     short drst3
4388                                    <1>
4389                                    <1> ;----------------------------------------
4390                                    <1> ;       DISK STATUS ROUTINE  (AH = 01H) :
4391                                    <1> ;----------------------------------------
```

```
                                <1>
                                <1> RETURN_STATUS:
00004621 A0[9B520100]           <1>       MOV    AL,[DISK_STATUS1]   ; OBTAIN PREVIOUS STATUS
00004626 C605[9B520100]00       <1>         MOV     byte [DISK_STATUS1],0   ; RESET STATUS
0000462D C3                     <1>       RETn
                                <1>
                                <1> ;---------------------------------------
                                <1> ;     DISK READ ROUTINE    (AH = 02H) :
                                <1> ;---------------------------------------
                                <1>
                                <1> DISK_READ:
0000462E C645FE20               <1>       MOV    byte [CMD_BLOCK+6],READ_CMD
00004632 E954020000             <1>         JMP     COMMANDI
                                <1>
                                <1> ;---------------------------------------
                                <1> ;     DISK WRITE ROUTINE   (AH = 03H) :
                                <1> ;---------------------------------------
                                <1>
                                <1> DISK_WRITE:
00004637 C645FE30               <1>       MOV    byte [CMD_BLOCK+6],WRITE_CMD
0000463B E9A6020000             <1>         JMP     COMMANDO
                                <1>
                                <1> ;---------------------------------------
                                <1> ;     DISK VERIFY       (AH = 04H) :
                                <1> ;---------------------------------------
                                <1>
                                <1> DISK_VERF:
00004640 C645FE40               <1>       MOV    byte [CMD_BLOCK+6],VERIFY_CMD
00004644 E814030000             <1>       CALL   COMMAND
00004649 750C                   <1>       JNZ    short VERF_EXIT           ; CONTROLLER STILL BUSY
0000464B E886030000             <1>       CALL   _WAIT               ; (Original: CALL WAIT)
00004650 7505                   <1>       JNZ    short VERF_EXIT           ; TIME OUT
00004652 E813040000             <1>       CALL   CHECK_STATUS
                                <1> VERF_EXIT:
00004657 C3                     <1>       RETn
                                <1>
                                <1> ;---------------------------------------
                                <1> ;     FORMATTING        (AH = 05H) :
                                <1> ;---------------------------------------
                                <1>
                                <1> FMT_TRK:                          ; FORMAT TRACK      (AH = 005H)
00004658 C645FE50               <1>       MOV    byte [CMD_BLOCK+6],FMTTRK_CMD
                                <1>       ;PUSH  ES
                                <1>       ;PUSH  BX
0000465C 53                     <1>       push   ebx
0000465D E8BA040000             <1>       CALL   GET_VEC             ; GET DISK PARAMETERS ADDRESS
                                <1>       ;MOV   AL,[ES:BX+14]       ; GET SECTORS/TRACK
00004662 8A430E                 <1>       mov    al, [ebx+14]
00004665 8845F9                 <1>       MOV    [CMD_BLOCK+1],AL    ; SET SECTOR COUNT IN COMMAND
00004668 5B                     <1>       pop    ebx
                                <1>       ;POP   BX
                                <1>       ;POP   ES
00004669 E97F020000             <1>         JMP     CMD_OF                  ; GO EXECUTE THE COMMAND
                                <1>
                                <1> ;---------------------------------------
                                <1> ;     READ DASD TYPE           (AH = 15H) :
                                <1> ;---------------------------------------
                                <1>
                                <1> READ_DASD_TYPE:
                                <1> READ_D_T:                         ; GET DRIVE PARAMETERS
0000466E 1E                     <1>       PUSH   DS                   ; SAVE REGISTERS
                                <1>       ;PUSH  ES
0000466F 53                     <1>       PUSH   eBX
                                <1>       ;CALL  DDS                  ; ESTABLISH ADDRESSING
                                <1>       ;push  cs
                                <1>       ;pop   ds
00004670 66BB1000               <1>         mov  bx, KDATA
00004674 8EDB                   <1>       mov    ds, bx
                                <1>       ;mov   es, bx
00004676 C605[9B520100]00       <1>       MOV     byte [DISK_STATUS1],0
0000467D 8A1D[9C520100]         <1>       MOV    BL,[HF_NUM]          ; GET NUMBER OF DRIVES
00004683 80E27F                 <1>       AND    DL,7FH               ; GET DRIVE NUMBER
00004686 38D3                   <1>       CMP    BL,DL
00004688 7627                   <1>       JBE    short RDT_NOT_PRESENT     ; RETURN DRIVE NOT PRESENT
0000468A E88D040000             <1>       CALL   GET_VEC              ; GET DISK PARAMETER ADDRESS
                                <1>       ;MOV   AL,[ES:BX+2]         ; HEADS
0000468F 8A4302                 <1>       mov    al, [ebx+2]
                                <1>       ;MOV   CL,[ES:BX+14]
00004692 8A4B0E                 <1>       mov    cl, [ebx+14]
00004695 F6E9                   <1>       IMUL   CL                   ; * NUMBER OF SECTORS
                                <1>       ;MOV   CX,[ES:BX]           ; MAX NUMBER OF CYLINDERS
00004697 668B0B                 <1>       mov    cx ,[ebx]
                                <1>       ;
                                <1>       ; 02/01/2015
                                <1>       ; ** leave the last cylinder as reserved for diagnostics **
                                <1>       ; (Also in Award BIOS - 1999, AHDSK.ASM, FUN15 -> sub ax, 1)
0000469A 6649                   <1>       DEC    CX                   ; LEAVE ONE FOR DIAGNOSTICS
                                <1>       ;
0000469C 66F7E9                 <1>       IMUL   CX                   ; NUMBER OF SECTORS
0000469F 6689D1                 <1>       MOV    CX,DX                ; HIGH ORDER HALF
000046A2 6689C2                 <1>       MOV    DX,AX                ; LOW ORDER HALF
                                <1>       ;SUB   AX,AX
000046A5 28C0                   <1>       sub    al, al
000046A7 B403                   <1>       MOV    AH,03H               ; INDICATE FIXED DISK
000046A9 5B                     <1> RDT2: POP    eBX                  ; RESTORE REGISTERS
                                <1>       ;POP   ES
000046AA 1F                     <1>       POP    DS
                                <1>       ; (*) CLC                   ; CLEAR CARRY
                                <1>       ;RETf  2
                                <1>       ; (*) 29/05/2016
                                <1>       ; (*) retf 4
000046AB 80642408FE             <1>       and    byte [esp+8], 0FEh ; clear carry bit of eflags register
```

```
4494 000046B0 CF                <1>       iretd
4495                            <1>
4496                            <1> RDT_NOT_PRESENT:
4497 000046B1 6629C0            <1>       SUB    AX,AX               ; DRIVE NOT PRESENT RETURN
4498 000046B4 6689C1            <1>       MOV    CX,AX               ; ZERO BLOCK COUNT
4499 000046B7 6689C2            <1>       MOV    DX,AX
4500 000046BA EBED              <1>       JMP    short RDT2
4501                            <1>
4502                            <1> ; 28/05/2016
4503                            <1> ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)
4504                            <1>
4505                            <1> ;----------------------------------------
4506                            <1> ;     GET PARAMETERS          (AH = 08H) :
4507                            <1> ;----------------------------------------
4508                            <1>
4509                            <1> GET_PARM_N:
4510                            <1>       ; ebx = user's buffer address for parameters table
4511                            <1> ;GET_PARM:                      ; GET DRIVE PARAMETERS
4512 000046BC 1E                <1>       PUSH   DS                  ; SAVE REGISTERS
4513 000046BD 06                <1>       PUSH   ES
4514 000046BE 53                <1>       PUSH   eBX
4515                            <1>       ;MOV   AX,ABS0             ; ESTABLISH ADDRESSING
4516                            <1>       ;MOV   DS,AX
4517                            <1>       ;TEST  DL,1                ; CHECK FOR DRIVE 1
4518                            <1>       ;JZ    short G0
4519                            <1>       ;LES   BX,@HF1_TBL_VEC
4520                            <1>       ;JMP   SHORT G1
4521                            <1> ;G0:  LES    BX,@HF_TBL_VEC
4522                            <1> ;G1:
4523                            <1>       ;CALL  DDS                 ; ESTABLISH SEGMENT
4524                            <1>       ; 22/12/2014
4525                            <1>       ;push  cs
4526                            <1>       ;pop   ds
4527 000046BF 66BB1000          <1>       mov    bx, KDATA
4528 000046C3 8EDB              <1>       mov    ds, bx
4529 000046C5 8EC3              <1>       mov    es, bx ; 27/05/2016
4530                            <1>       ;
4531 000046C7 80EA80            <1>       SUB    DL,80H
4532 000046CA 80FA04            <1>       CMP    DL,MAX_FILE         ; TEST WITHIN RANGE
4533 000046CD 7361              <1>       JAE    short G4
4534                            <1>       ;
4535 000046CF 31DB              <1>       xor    ebx, ebx ; 21/02/2015
4536                            <1>       ; 22/12/2014
4537 000046D1 88D3              <1>       mov    bl, dl
4538                            <1>       ;xor   bh, bh
4539 000046D3 C0E302            <1>       shl    bl, 2               ; convert index to offset
4540                            <1>       ;add   bx, HF_TBL_VEC
4541 000046D6 81C3[A0520100]    <1>       add    ebx, HF_TBL_VEC
4542                            <1>       ;mov   ax, [bx+2]
4543                            <1>       ;mov   es, ax              ; dpt segment
4544                            <1>       ;mov   bx, [bx]            ; dpt offset
4545 000046DC 8B1B              <1>       mov    ebx, [ebx] ; 32 bit offset
4546                            <1>
4547 000046DE C605[9B520100]00  <1>       MOV    byte [DISK_STATUS1],0
4548                            <1>         ;MOV     AX,[ES:BX]             ; MAX NUMBER OF CYLINDERS
4549 000046E5 668B03            <1>       mov    ax, [ebx]
4550                            <1>       ;;SUB  AX,2                ; ADJUST FOR 0-N
4551 000046E8 6648              <1>       dec    ax                  ; max. cylinder number
4552 000046EA 88C5              <1>       MOV    CH,AL
4553 000046EC 66250003          <1>       AND    AX,0300H            ; HIGH TWO BITS OF CYLINDER
4554 000046F0 66D1E8            <1>       SHR    AX,1
4555 000046F3 66D1E8            <1>       SHR    AX,1
4556                            <1>       ;OR    AL,[ES:BX+14]       ; SECTORS
4557 000046F6 0A430E            <1>       or     al, [ebx+14]
4558 000046F9 88C1              <1>       MOV    CL,AL
4559                            <1>       ;MOV   DH,[ES:BX+2]        ; HEADS
4560 000046FB 8A7302            <1>       mov    dh, [ebx+2]
4561 000046FE FECE              <1>       DEC    DH                  ; 0-N RANGE
4562 00004700 8A15[9C520100]    <1>       MOV    DL,[HF_NUM]         ; DRIVE COUNT
4563 00004706 6629C0            <1>       SUB    AX,AX
4564                            <1>         ;27/12/2014
4565                            <1>       ;mov   di, bx              ; HDPT offset
4566                            <1>
4567                            <1>       ; 27/05/2016
4568                            <1>       ; return fixed disk parameters table to user
4569                            <1>       ; in user's buffer, which is pointed by EBX
4570                            <1>       ;
4571 00004709 873C24            <1>       xchg   edi, [esp]          ; ebx (input)-> edi, edi -> [esp]
4572 0000470C 56                <1>       push   esi
4573 0000470D 89DE              <1>       mov    esi, ebx            ; hard disk parameter table (32 bytes)
4574 0000470F 89FB              <1>       mov    ebx, edi            ; ebx = user's buffer address
4575 00004711 51                <1>       push   ecx
4576 00004712 50                <1>       push   eax
4577 00004713 B920000000        <1>       mov    ecx, 32 ; 32 bytes
4578 00004718 E869A10000        <1>       call   transfer_to_user_buffer ; trdosk6.s (16/05/2016)
4579 0000471D 58                <1>       pop    eax
4580 0000471E 59                <1>       pop    ecx
4581 0000471F 5E                <1>       pop    esi
4582 00004720 5F                <1>       pop    edi
4583 00004721 730A              <1>       jnc    short G5
4584                            <1>       ; 29/05/2016 (*)
4585 00004723 B8FF000000        <1>       mov    eax, 0FFh ; unknown error !
4586                            <1> _G6:
4587 00004728 804C241001        <1>       or     byte [esp+16], 1 ; set carry bit of eflags register
4588                            <1> G5:
4589                            <1>       ; 27/05/2016
4590                            <1>       ;POP   eBX                 ; RESTORE REGISTERS
4591 0000472D 07                <1>       POP    ES
4592 0000472E 1F                <1>       POP    DS
4593                            <1>       ;RETf  2
4594                            <1>       ; (*) 29/05/2016
4595                            <1>       ; (*) retf 4
```

```
4596                                    <1>       ; (*) or byte [esp+8], 1 ; set carry bit of eflags register
4597 0000472F CF                        <1>       iretd
4598                                    <1> G4:
4599 00004730 C605[9B520100]07          <1>       MOV     byte [DISK_STATUS1],INIT_FAIL ; OPERATION FAILED
4600 00004737 B407                      <1>       MOV    AH,INIT_FAIL
4601 00004739 28C0                      <1>       SUB    AL,AL
4602 0000473B 6629D2                    <1>       SUB    DX,DX
4603 0000473E 6629C9                    <1>       SUB    CX,CX
4604                                    <1>       ; 29/05/2016 (*)
4605                                    <1>       ;STC                    ; SET ERROR FLAG
4606                                    <1>       ;JMP   short G5
4607 00004741 EBE5                      <1>       jmp    short _G6
4608                                    <1>
4609                                    <1> ;--------------------------------------
4610                                    <1> ;     INITIALIZE DRIVE     (AH = 09H) :
4611                                    <1> ;--------------------------------------
4612                                    <1>       ; 03/01/2015
4613                                    <1>       ; According to ATA-ATAPI specification v2.0 to v5.0
4614                                    <1>       ; logical sector per logical track
4615                                    <1>       ; and logical heads - 1 would be set but
4616                                    <1>       ; it is seen as it will be good
4617                                    <1>       ; if physical parameters will be set here
4618                                    <1>       ; because, number of heads <= 16.
4619                                    <1>       ; (logical heads usually more than 16)
4620                                    <1>       ; NOTE: ATA logical parameters (software C, H, S)
4621                                    <1>       ;     == INT 13h physical parameters
4622                                    <1>
4623                                    <1> ;INIT_DRV:
4624                                    <1> ;     MOV    byte [CMD_BLOCK+6],SET_PARM_CMD
4625                                    <1> ;     CALL   GET_VEC             ; ES:BX -> PARAMETER BLOCK
4626                                    <1> ;     MOV    AL,[ES:BX+2]        ; GET NUMBER OF HEADS
4627                                    <1> ;     DEC    AL                  ; CONVERT TO 0-INDEX
4628                                    <1> ;     MOV    AH,[CMD_BLOCK+5]    ; GET SDH REGISTER
4629                                    <1> ;     AND    AH,0F0H             ; CHANGE HEAD NUMBER
4630                                    <1> ;     OR     AH,AL               ; TO MAX HEAD
4631                                    <1> ;     MOV    [CMD_BLOCK+5],AH
4632                                    <1> ;     MOV    AL,[ES:BX+14]       ; MAX SECTOR NUMBER
4633                                    <1> ;     MOV    [CMD_BLOCK+1],AL
4634                                    <1> ;     SUB    AX,AX
4635                                    <1> ;     MOV    [CMD_BLOCK+3],AL    ; ZERO FLAGS
4636                                    <1> ;     CALL   COMMAND             ; TELL CONTROLLER
4637                                    <1> ;     JNZ    short INIT_EXIT           ; CONTROLLER BUSY ERROR
4638                                    <1> ;     CALL   NOT_BUSY            ; WAIT FOR IT TO BE DONE
4639                                    <1> ;     JNZ    short INIT_EXIT           ; TIME OUT
4640                                    <1> ;     CALL   CHECK_STATUS
4641                                    <1> ;INIT_EXIT:
4642                                    <1> ;     RETn
4643                                    <1>
4644                                    <1> ; 04/01/2015
4645                                    <1> ; 02/01/2015 - Derived from from AWARD BIOS 1999
4646                                    <1> ;                          AHDSK.ASM - INIT_DRIVE
4647                                    <1> INIT_DRV:
4648                                    <1>       ;xor   ah,ah
4649 00004743 31C0                      <1>       xor    eax, eax ; 21/02/2015
4650 00004745 B00B                      <1>       mov    al,11 ; Physical heads from translated HDPT
4651 00004747 3825[B0520100]            <1>         cmp     [LBAMode], ah   ; 0
4652 0000474D 7702                      <1>       ja     short idrv0
4653 0000474F B002                      <1>       mov    al,2  ; Physical heads from standard HDPT
4654                                    <1> idrv0:
4655                                    <1>       ; DL = drive number (0 based)
4656 00004751 E8C6030000                <1>       call   GET_VEC
4657                                    <1>       ;push  bx
4658 00004756 53                        <1>       push   ebx ; 21/02/2015
4659                                    <1>       ;add   bx,ax
4660 00004757 01C3                      <1>       add    ebx, eax
4661                                    <1>       ;; 05/01/2015
4662 00004759 8A25[EC5C0000]            <1>       mov    ah, [hf_m_s] ; drive number (0= master, 1= slave)
4663                                    <1>       ;;and  ah,1
4664 0000475F C0E404                    <1>       shl    ah,4
4665 00004762 80CCA0                    <1>       or     ah,0A0h  ; Drive/Head register - 10100000b (A0h)
4666                                    <1>       ;mov   al,[es:bx]
4667 00004765 8A03                      <1>       mov    al, [ebx] ; 21/02/2015
4668 00004767 FEC8                      <1>       dec    al      ; last head number
4669                                    <1>       ;and   al,0Fh
4670 00004769 08E0                      <1>       or     al,ah   ; lower 4 bits for head number
4671                                    <1>       ;
4672 0000476B C645FE91                  <1>       mov    byte [CMD_BLOCK+6],SET_PARM_CMD
4673 0000476F 8845FD                    <1>       mov    [CMD_BLOCK+5],al
4674                                    <1>       ;pop   bx
4675 00004772 5B                        <1>       pop    ebx
4676 00004773 29C0                      <1>       sub    eax, eax ; 21/02/2015
4677 00004775 B004                      <1>       mov    al,4 ; Physical sec per track from translated HDPT
4678 00004777 803D[B0520100]00          <1>       cmp    byte [LBAMode], 0
4679 0000477E 7702                      <1>       ja     short idrv1
4680 00004780 B00E                      <1>       mov    al,14 ; Physical sec per track from standard HDPT
4681                                    <1> idrv1:
4682                                    <1>       ;xor   ah,ah
4683                                    <1>       ;add   bx,ax
4684 00004782 01C3                      <1>       add    ebx, eax ; 21/02/2015
4685                                    <1>       ;mov   al,[es:bx]
4686                                    <1>                      ; sector number
4687 00004784 8A03                      <1>       mov    al, [ebx]
4688 00004786 8845F9                    <1>       mov    [CMD_BLOCK+1],al
4689 00004789 28C0                      <1>       sub    al,al
4690 0000478B 8845FB                    <1>       mov    [CMD_BLOCK+3],al  ; ZERO FLAGS
4691 0000478E E8CA010000                <1>       call   COMMAND        ; TELL CONTROLLER
4692 00004793 750C                      <1>       jnz    short INIT_EXIT       ; CONTROLLER BUSY ERROR
4693 00004795 E878020000                <1>       call   NOT_BUSY       ; WAIT FOR IT TO BE DONE
4694 0000479A 7505                      <1>       jnz    short INIT_EXIT       ; TIME OUT
4695 0000479C E8C9020000                <1>       call   CHECK_STATUS
4696                                    <1> INIT_EXIT:
4697 000047A1 C3                        <1>       RETn
```

```
4698                                    <1>
4699                                    <1> ;----------------------------------------
4700                                    <1> ;     READ LONG         (AH = 0AH) :
4701                                    <1> ;----------------------------------------
4702                                    <1>
4703                                    <1> RD_LONG:
4704                                    <1>       ;MOV   @CMD_BLOCK+6,READ_CMD OR ECC_MODE
4705 000047A2 C645FE22                 <1>         mov     byte [CMD_BLOCK+6],READ_CMD + ECC_MODE
4706 000047A6 E9E0000000               <1>         JMP     COMMANDI
4707                                    <1>
4708                                    <1> ;----------------------------------------
4709                                    <1> ;     WRITE LONG        (AH = 0BH) :
4710                                    <1> ;----------------------------------------
4711                                    <1>
4712                                    <1> WR_LONG:
4713                                    <1>       ;MOV   @CMD_BLOCK+6,WRITE_CMD OR ECC_MODE
4714 000047AB C645FE32                 <1>         MOV     byte [CMD_BLOCK+6],WRITE_CMD + ECC_MODE
4715 000047AF E932010000               <1>         JMP     COMMANDO
4716                                    <1>
4717                                    <1> ;----------------------------------------
4718                                    <1> ;     SEEK              (AH = 0CH) :
4719                                    <1> ;----------------------------------------
4720                                    <1>
4721                                    <1> DISK_SEEK:
4722 000047B4 C645FE70                 <1>         MOV     byte [CMD_BLOCK+6],SEEK_CMD
4723 000047B8 E8A0010000               <1>       CALL   COMMAND
4724 000047BD 751C                     <1>       JNZ    short DS_EXIT             ; CONTROLLER BUSY ERROR
4725 000047BF E812020000               <1>       CALL   _WAIT
4726 000047C4 7515                     <1>         JNZ     DS_EXIT                  ; TIME OUT ON SEEK
4727 000047C6 E89F020000               <1>       CALL   CHECK_STATUS
4728 000047CB 803D[9B520100]40         <1>         CMP     byte [DISK_STATUS1],BAD_SEEK
4729 000047D2 7507                     <1>       JNE    short DS_EXIT
4730 000047D4 C605[9B520100]00         <1>         MOV     byte [DISK_STATUS1],0
4731                                    <1> DS_EXIT:
4732 000047DB C3                       <1>       RETn
4733                                    <1>
4734                                    <1> ;----------------------------------------
4735                                    <1> ;     TEST DISK READY      (AH = 10H) :
4736                                    <1> ;----------------------------------------
4737                                    <1>
4738                                    <1> TST_RDY:                          ; WAIT FOR CONTROLLER
4739 000047DC E831020000               <1>       CALL   NOT_BUSY
4740 000047E1 751C                     <1>       JNZ    short TR_EX
4741 000047E3 8A45FD                   <1>       MOV    AL,[CMD_BLOCK+5]    ; SELECT DRIVE
4742 000047E6 668B15[E85C0000]         <1>       MOV    DX,[HF_PORT]
4743 000047ED 80C206                   <1>       add    dl,6
4744 000047F0 EE                       <1>       OUT    DX,AL
4745 000047F1 E88C020000               <1>       CALL   CHECK_ST           ; CHECK STATUS ONLY
4746 000047F6 7507                     <1>       JNZ    short TR_EX
4747 000047F8 C605[9B520100]00         <1>       MOV    byte [DISK_STATUS1],0     ; WIPE OUT DATA CORRECTED ERROR
4748                                    <1> TR_EX:
4749 000047FF C3                       <1>       RETn
4750                                    <1>
4751                                    <1> ;----------------------------------------
4752                                    <1> ;     RECALIBRATE       (AH = 11H) :
4753                                    <1> ;----------------------------------------
4754                                    <1>
4755                                    <1> HDISK_RECAL:
4756 00004800 C645FE10                 <1>         MOV     byte [CMD_BLOCK+6],RECAL_CMD ; 10h, 16
4757 00004804 E854010000               <1>       CALL   COMMAND             ; START THE OPERATION
4758 00004809 7523                     <1>       JNZ    short RECAL_EXIT    ; ERROR
4759 0000480B E8C6010000               <1>       CALL   _WAIT               ; WAIT FOR COMPLETION
4760 00004810 7407                     <1>       JZ     short RECAL_X              ; TIME OUT ONE OK ?
4761 00004812 E8BF010000               <1>       CALL   _WAIT               ; WAIT FOR COMPLETION LONGER
4762 00004817 7515                     <1>       JNZ    short RECAL_EXIT    ; TIME OUT TWO TIMES IS ERROR
4763                                    <1> RECAL_X:
4764 00004819 E84C020000               <1>       CALL   CHECK_STATUS
4765 0000481E 803D[9B520100]40         <1>       CMP    byte [DISK_STATUS1],BAD_SEEK ; SEEK NOT COMPLETE
4766 00004825 7507                     <1>       JNE    short RECAL_EXIT    ; IS OK
4767 00004827 C605[9B520100]00         <1>       MOV    byte [DISK_STATUS1],0
4768                                    <1> RECAL_EXIT:
4769 0000482E 803D[9B520100]00         <1>         CMP     byte [DISK_STATUS1],0
4770 00004835 C3                       <1>       RETn
4771                                    <1>
4772                                    <1> ;----------------------------------------
4773                                    <1> ;      CONTROLLER DIAGNOSTIC (AH = 14H) :
4774                                    <1> ;----------------------------------------
4775                                    <1>
4776                                    <1> CTLR_DIAGNOSTIC:
4777 00004836 FA                       <1>         CLI                             ; DISABLE INTERRUPTS WHILE CHANGING MASK
4778 00004837 E4A1                     <1>       IN     AL,INTB01          ; TURN ON SECOND INTERRUPT CHIP
4779                                    <1>       ;AND   AL,0BFH
4780 00004839 243F                     <1>       and    al, 3Fh                  ; enable IRQ 14 & IRQ 15
4781                                    <1>       ;JMP   $+2
4782                                    <1>       IODELAY
4782 0000483B EB00                     <2>  jmp short $+2
4782 0000483D EB00                     <2>  jmp short $+2
4783 0000483F E6A1                     <1>       OUT    INTB01,AL
4784                                    <1>       IODELAY
4784 00004841 EB00                     <2>  jmp short $+2
4784 00004843 EB00                     <2>  jmp short $+2
4785 00004845 E421                     <1>       IN     AL,INTA01          ; LET INTERRUPTS PASS THRU TO
4786 00004847 24FB                     <1>       AND    AL,0FBH            ;  SECOND CHIP
4787                                    <1>       ;JMP   $+2
4788                                    <1>       IODELAY
4788 00004849 EB00                     <2>  jmp short $+2
4788 0000484B EB00                     <2>  jmp short $+2
4789 0000484D E621                     <1>       OUT    INTA01,AL
4790 0000484F FB                       <1>       STI
4791 00004850 E8BD010000               <1>       CALL   NOT_BUSY           ; WAIT FOR CARD
4792 00004855 752B                     <1>       JNZ    short CD_ERR       ; BAD CARD
4793                                    <1>       ;MOV   DX, HF_PORT+7
```

```
4794 00004857 668B15[E85C0000]    <1>       mov    dx, [HF_PORT]
4795 0000485E 80C207              <1>       add    dl, 7
4796 00004861 B090                <1>       MOV    AL,DIAG_CMD         ; START DIAGNOSE
4797 00004863 EE                  <1>       OUT    DX,AL
4798 00004864 E8A9010000          <1>       CALL   NOT_BUSY            ; WAIT FOR IT TO COMPLETE
4799 00004869 B480                <1>       MOV    AH,TIME_OUT
4800 0000486B 7517                <1>       JNZ    short CD_EXIT             ; TIME OUT ON DIAGNOSTIC
4801                              <1>       ;MOV   DX,HF_PORT+1        ; GET ERROR REGISTER
4802 0000486D 668B15[E85C0000]    <1>       mov    dx, [HF_PORT]
4803 00004874 FEC2                <1>       inc    dl
4804 00004876 EC                  <1>       IN     AL,DX
4805 00004877 A2[92520100]        <1>       MOV    [HF_ERROR],AL       ; SAVE IT
4806 0000487C B400                <1>       MOV    AH,0
4807 0000487E 3C01                <1>       CMP    AL,1                ; CHECK FOR ALL OK
4808 00004880 7402                <1>       JE     SHORT CD_EXIT
4809 00004882 B420                <1> CD_ERR: MOV  AH,BAD_CNTLR
4810                              <1> CD_EXIT:
4811 00004884 8825[9B520100]      <1>       MOV    [DISK_STATUS1],AH
4812 0000488A C3                  <1>       RETn
4813                              <1>
4814                              <1> ;---------------------------------------
4815                              <1> ; COMMANDI                        :
4816                              <1> ;     REPEATEDLY INPUTS DATA TILL      :
4817                              <1> ;     NSECTOR RETURNS ZERO             :
4818                              <1> ;---------------------------------------
4819                              <1> COMMANDI:
4820 0000488B E862020000          <1>       CALL   CHECK_DMA           ; CHECK 64K BOUNDARY ERROR
4821 00004890 7253                <1>       JC     short CMD_ABORT
4822                              <1>       ;MOV   DI,BX
4823 00004892 89DF                <1>       mov    edi, ebx ; 21/02/2015
4824 00004894 E8C4000000          <1>       CALL   COMMAND             ; OUTPUT COMMAND
4825 00004899 754A                <1>       JNZ    short CMD_ABORT
4826                              <1> CMD_I1:
4827 0000489B E836010000          <1>       CALL   _WAIT               ; WAIT FOR DATA REQUEST INTERRUPT
4828 000048A0 7543                <1>       JNZ    short TM_OUT        ; TIME OUT
4829                              <1> cmd_i1x: ; 18/02/2016
4830                              <1>       ;MOV   CX,256              ; SECTOR SIZE IN WORDS
4831 000048A2 B900010000          <1>       mov    ecx, 256 ; 21/02/2015
4832                              <1>       ;MOV   DX,HF_PORT
4833 000048A7 668B15[E85C0000]    <1>       mov    dx,[HF_PORT]
4834 000048AE FA                  <1>       CLI
4835 000048AF FC                  <1>       CLD
4836 000048B0 F3666D              <1>       REP    INSW                ; GET THE SECTOR
4837 000048B3 FB                  <1>       STI
4838 000048B4 F645FE02            <1>       TEST   byte [CMD_BLOCK+6],ECC_MODE ; CHECK FOR NORMAL INPUT
4839 000048B8 7419                <1>       JZ     short CMD_I3
4840 000048BA E880010000          <1>       CALL   WAIT_DRQ            ; WAIT FOR DATA REQUEST
4841 000048BF 7224                <1>       JC     short TM_OUT
4842                              <1>       ;MOV   DX,HF_PORT
4843 000048C1 668B15[E85C0000]    <1>       mov    dx,[HF_PORT]
4844                              <1>       ;MOV   CX,4                ; GET ECC BYTES
4845 000048C8 B904000000          <1>       mov    ecx, 4 ; mov cx, 4
4846 000048CD EC                  <1> CMD_I2: IN   AL,DX
4847                              <1>       ;MOV   [ES:DI],AL          ; GO SLOW FOR BOARD
4848 000048CE 8807                <1>       mov    [edi], al ; 21/02/2015
4849 000048D0 47                  <1>       INC    eDI
4850 000048D1 E2FA                <1>       LOOP   CMD_I2
4851                              <1> CMD_I3:
4852                              <1>       ; wait for 400 ns
4853 000048D3 80C207              <1>       add    dl, 7
4854 000048D6 EC                  <1>       in     al, dx
4855 000048D7 EC                  <1>       in     al, dx
4856 000048D8 EC                  <1>       in     al, dx
4857                              <1>       ;
4858 000048D9 E88C010000          <1>       CALL   CHECK_STATUS
4859 000048DE 7505                <1>       JNZ    short CMD_ABORT           ; ERROR RETURNED
4860 000048E0 FE4DF9              <1>       DEC    byte [CMD_BLOCK+1]  ; CHECK FOR MORE
4861                              <1>       ;JNZ   SHORT CMD_I1
4862 000048E3 75BD                <1>       jnz    short cmd_i1x ; 18/02/2016
4863                              <1> CMD_ABORT:
4864 000048E5 C3                  <1> TM_OUT: RETn
4865                              <1>
4866                              <1> ;---------------------------------------
4867                              <1> ; COMMANDO                        :
4868                              <1> ;     REPEATEDLY OUTPUTS DATA TILL     :
4869                              <1> ;     NSECTOR RETURNS ZERO             :
4870                              <1> ;---------------------------------------
4871                              <1> COMMANDO:
4872 000048E6 E807020000          <1>       CALL   CHECK_DMA           ; CHECK 64K BOUNDARY ERROR
4873 000048EB 72F8                <1>       JC     short CMD_ABORT
4874 000048ED 89DE                <1> CMD_OF: MOV  eSI,eBX ; 21/02/2015
4875 000048EF E869000000          <1>       CALL   COMMAND             ; OUTPUT COMMAND
4876 000048F4 75EF                <1>       JNZ    short CMD_ABORT
4877 000048F6 E844010000          <1>       CALL   WAIT_DRQ            ; WAIT FOR DATA REQUEST
4878 000048FB 72E8                <1>       JC     short TM_OUT              ; TOO LONG
4879                              <1> CMD_O1: ;PUSH      DS
4880                              <1>       ;PUSH  ES                  ; MOVE ES TO DS
4881                              <1>       ;POP   DS
4882                              <1>       ;MOV   CX,256              ; PUT THE DATA OUT TO THE CARD
4883                              <1>       ;MOV   DX,HF_PORT
4884                              <1>       ; 01/02/2015
4885 000048FD 668B15[E85C0000]    <1>       mov    dx, [HF_PORT]
4886                              <1>       ;push  es
4887                              <1>       ;pop   ds
4888                              <1>       ;mov   cx, 256
4889 00004904 B900010000          <1>       mov    ecx, 256 ; 21/02/2015
4890 00004909 FA                  <1>       CLI
4891 0000490A FC                  <1>       CLD
4892 0000490B F3666F              <1>       REP    OUTSW
4893 0000490E FB                  <1>       STI
4894                              <1>       ;POP   DS                  ; RESTORE DS
4895 0000490F F645FE02            <1>       TEST   byte [CMD_BLOCK+6],ECC_MODE ; CHECK FOR NORMAL OUTPUT
```

```
4896 00004913 7419                <1>       JZ     short CMD_O3
4897 00004915 E825010000          <1>       CALL   WAIT_DRQ            ; WAIT FOR DATA REQUEST
4898 0000491A 72C9                <1>       JC     short TM_OUT
4899                              <1>       ;MOV   DX,HF_PORT
4900 0000491C 668B15[E85C0000]    <1>       mov    dx, [HF_PORT]
4901                              <1>       ;MOV   CX,4                ; OUTPUT THE ECC BYTES
4902 00004923 B904000000          <1>       mov    ecx, 4  ; mov cx, 4
4903                              <1> CMD_O2: ;MOV AL,[ES:SI]
4904 00004928 8A06                <1>       mov    al, [esi]
4905 0000492A EE                  <1>       OUT    DX,AL
4906 0000492B 46                  <1>       INC    eSI
4907 0000492C E2FA                <1>       LOOP   CMD_O2
4908                              <1> CMD_O3:
4909 0000492E E8A3000000          <1>       CALL   _WAIT               ; WAIT FOR SECTOR COMPLETE INTERRUPT
4910 00004933 75B0                <1>       JNZ    short TM_OUT        ; ERROR RETURNED
4911 00004935 E830010000          <1>       CALL   CHECK_STATUS
4912 0000493A 75A9                <1>       JNZ    short CMD_ABORT
4913 0000493C F605[91520100]08    <1>       TEST   byte [HF_STATUS],ST_DRQ   ; CHECK FOR MORE
4914 00004943 75B8                <1>       JNZ    SHORT CMD_O1
4915                              <1>       ;MOV   DX,HF_PORT+2        ; CHECK RESIDUAL SECTOR COUNT
4916 00004945 668B15[E85C0000]    <1>       mov    dx, [HF_PORT]
4917                              <1>       ;add   dl, 2
4918 0000494C FEC2                <1>       inc    dl
4919 0000494E FEC2                <1>       inc    dl
4920 00004950 EC                  <1>       IN     AL,DX               ;
4921 00004951 A8FF                <1>       TEST   AL,0FFH             ;
4922 00004953 7407                <1>       JZ     short CMD_O4              ; COUNT = 0  OK
4923 00004955 C605[9B520100]BB    <1>       MOV    byte [DISK_STATUS1],UNDEF_ERR
4924                              <1>                                  ; OPERATION ABORTED - PARTIAL TRANSFER
4925                              <1> CMD_O4:
4926 0000495C C3                  <1>       RETn
4927                              <1>
4928                              <1> ;--------------------------------------------------
4929                              <1> ; COMMAND                                       :
4930                              <1> ;     THIS ROUTINE OUTPUTS THE COMMAND BLOCK       :
4931                              <1> ; OUTPUT                                        :
4932                              <1> ;     BL = STATUS                               :
4933                              <1> ;     BH = ERROR REGISTER                       :
4934                              <1> ;--------------------------------------------------
4935                              <1>
4936                              <1> COMMAND:
4937 0000495D 53                  <1>       PUSH   eBX                 ; WAIT FOR SEEK COMPLETE AND READY
4938                              <1>       ;;MOV  CX,DELAY_2          ; SET INITIAL DELAY BEFORE TEST
4939                              <1> COMMAND1:
4940                              <1>       ;;PUSH CX                  ; SAVE LOOP COUNT
4941 0000495E E879FEFFFF          <1>       CALL   TST_RDY             ; CHECK DRIVE READY
4942                              <1>       ;;POP  CX
4943 00004963 7419                <1>       JZ     short COMMAND2             ; DRIVE IS READY
4944 00004965 803D[9B520100]80    <1>         CMP     byte [DISK_STATUS1],TIME_OUT ; TST_RDY TIMED OUT--GIVE UP
4945                              <1>       ;JZ    short CMD_TIMEOUT
4946                              <1>       ;;LOOP COMMAND1            ; KEEP TRYING FOR A WHILE
4947                              <1>       ;JMP   SHORT COMMAND4             ; ITS NOT GOING TO GET READY
4948 0000496C 7507                <1>       jne    short COMMAND4
4949                              <1> CMD_TIMEOUT:
4950 0000496E C605[9B520100]20    <1>       MOV    byte [DISK_STATUS1],BAD_CNTLR
4951                              <1> COMMAND4:
4952 00004975 5B                  <1>       POP    eBX
4953 00004976 803D[9B520100]00    <1>         CMP     byte [DISK_STATUS1],0   ; SET CONDITION CODE FOR CALLER
4954 0000497D C3                  <1>       RETn
4955                              <1> COMMAND2:
4956 0000497E 5B                  <1>       POP    eBX
4957 0000497F 57                  <1>       PUSH   eDI
4958 00004980 C605[93520100]00    <1>       MOV    byte [HF_INT_FLAG],0       ; RESET INTERRUPT FLAG
4959 00004987 FA                  <1>       CLI                        ; INHIBIT INTERRUPTS WHILE CHANGING MASK
4960 00004988 E4A1                <1>       IN     AL,INTB01           ; TURN ON SECOND INTERRUPT CHIP
4961                              <1>       ;AND   AL,0BFH
4962 0000498A 243F                <1>       and    al, 3Fh                    ; Enable IRQ 14 & 15
4963                              <1>       ;JMP   $+2
4964                              <1>       IODELAY
4964 0000498C EB00                <2>  jmp short $+2
4964 0000498E EB00                <2>  jmp short $+2
4965 00004990 E6A1                <1>       OUT    INTB01,AL
4966 00004992 E421                <1>       IN     AL,INTA01           ; LET INTERRUPTS PASS THRU TO
4967 00004994 24FB                <1>       AND    AL,0FBH             ;  SECOND CHIP
4968                              <1>       ;JMP   $+2
4969                              <1>       IODELAY
4969 00004996 EB00                <2>  jmp short $+2
4969 00004998 EB00                <2>  jmp short $+2
4970 0000499A E621                <1>       OUT    INTA01,AL
4971 0000499C FB                  <1>       STI
4972 0000499D 31FF                <1>       XOR    eDI,eDI                    ; INDEX THE COMMAND TABLE
4973                              <1>       ;MOV   DX,HF_PORT+1        ; DISK ADDRESS
4974 0000499F 668B15[E85C0000]    <1>       mov    dx, [HF_PORT]
4975 000049A6 FEC2                <1>       inc    dl
4976 000049A8 F605[9D520100]C0    <1>       TEST   byte [CONTROL_BYTE],0C0H ; CHECK FOR RETRY SUPPRESSION
4977 000049AF 7411                <1>       JZ     short COMMAND3
4978 000049B1 8A45FE              <1>       MOV    AL, [CMD_BLOCK+6]   ; YES-GET OPERATION CODE
4979 000049B4 24F0                <1>       AND    AL,0F0H             ; GET RID OF MODIFIERS
4980 000049B6 3C20                <1>       CMP    AL,20H              ; 20H-40H IS READ, WRITE, VERIFY
4981 000049B8 7208                <1>       JB     short COMMAND3
4982 000049BA 3C40                <1>       CMP    AL,40H
4983 000049BC 7704                <1>       JA     short COMMAND3
4984 000049BE 804DFE01            <1>       OR     byte [CMD_BLOCK+6],NO_RETRIES
4985                              <1>                                  ; VALID OPERATION FOR RETRY SUPPRESS
4986                              <1> COMMAND3:
4987 000049C2 8A443DF8            <1>       MOV    AL,[CMD_BLOCK+eDI]  ; GET THE COMMAND STRING BYTE
4988 000049C6 EE                  <1>       OUT    DX,AL               ; GIVE IT TO CONTROLLER
4989                              <1>       IODELAY
4989 000049C7 EB00                <2>  jmp short $+2
4989 000049C9 EB00                <2>  jmp short $+2
4990 000049CB 47                  <1>       INC    eDI                 ; NEXT BYTE IN COMMAND BLOCK
4991 000049CC 6642                <1>       INC    DX                  ; NEXT DISK ADAPTER REGISTER
```

```
4992 000049CE 6683FF07          <1>       cmp    di, 7  ; 1/1/2015   ; ALL DONE?
4993 000049D2 75EE              <1>       JNZ    short COMMAND3            ; NO--GO DO NEXT ONE
4994 000049D4 5F                <1>       POP    eDI
4995 000049D5 C3                <1>       RETn                       ; ZERO FLAG IS SET
4996                            <1>
4997                            <1> ;CMD_TIMEOUT:
4998                            <1> ;     MOV    byte [DISK_STATUS1],BAD_CNTLR
4999                            <1> ;COMMAND4:
5000                            <1> ;     POP    BX
5001                            <1> ;     CMP    [DISK_STATUS1],0    ; SET CONDITION CODE FOR CALLER
5002                            <1> ;     RETn
5003                            <1>
5004                            <1> ;---------------------------------------
5005                            <1> ;     WAIT FOR INTERRUPT        :
5006                            <1> ;---------------------------------------
5007                            <1> ;WAIT:
5008                            <1> _WAIT:
5009 000049D6 FB                <1>       STI                        ; MAKE SURE INTERRUPTS ARE ON
5010                            <1>       ;SUB   CX,CX               ; SET INITIAL DELAY BEFORE TEST
5011                            <1>       ;CLC
5012                            <1>       ;MOV   AX,9000H            ; DEVICE WAIT INTERRUPT
5013                            <1>       ;INT   15H
5014                            <1>       ;JC    WT2                 ; DEVICE TIMED OUT
5015                            <1>       ;MOV   BL,DELAY_1          ; SET DELAY COUNT
5016                            <1>
5017                            <1>       ;mov   bl, WAIT_HDU_INT_HI
5018                            <1>       ;; 21/02/2015
5019                            <1>       ;;mov  bl, WAIT_HDU_INT_HI + 1
5020                            <1>       ;;mov  cx, WAIT_HDU_INT_LO
5021 000049D7 B915160500        <1>       mov    ecx, WAIT_HDU_INT_LH
5022                            <1>                                  ; (AWARD BIOS -> WAIT_FOR_MEM)
5023                            <1> ;-----      WAIT LOOP
5024                            <1>
5025                            <1> WT1:
5026                            <1>       ;TEST  byte [HF_INT_FLAG],80H    ; TEST FOR INTERRUPT
5027 000049DC F605[93520100]C0  <1>       test   byte [HF_INT_FLAG],0C0h
5028                            <1>       ;LOOPZ WT1
5029 000049E3 7517              <1>       JNZ    short WT3           ; INTERRUPT--LETS GO
5030                            <1>       ;DEC   BL
5031                            <1>       ;JNZ   short WT1           ; KEEP TRYING FOR A WHILE
5032                            <1>
5033                            <1> WT1_hi:
5034 000049E5 E461              <1>       in     al, SYS1 ; 61h (PORT_B)   ; wait for lo to hi
5035 000049E7 A810              <1>       test   al, 10h                   ; transition on memory
5036 000049E9 75FA              <1>       jnz    short WT1_hi        ; refresh.
5037                            <1> WT1_lo:
5038 000049EB E461              <1>       in     al, SYS1            ; 061h (PORT_B)
5039 000049ED A810              <1>       test   al, 10h
5040 000049EF 74FA              <1>       jz     short WT1_lo
5041 000049F1 E2E9              <1>       loop   WT1
5042                            <1>       ;;or   bl, bl
5043                            <1>       ;;jz   short WT2
5044                            <1>       ;;dec  bl
5045                            <1>       ;;jmp  short WT1
5046                            <1>       ;dec   bl
5047                            <1>       ;jnz   short WT1
5048                            <1>
5049 000049F3 C605[9B520100]80  <1> WT2:  MOV    byte [DISK_STATUS1],TIME_OUT ; REPORT TIME OUT ERROR
5050 000049FA EB0E              <1>       JMP    SHORT WT4
5051 000049FC C605[9B520100]00  <1> WT3:  MOV    byte [DISK_STATUS1],0
5052 00004A03 C605[93520100]00  <1>       MOV    byte [HF_INT_FLAG],0
5053 00004A0A 803D[9B520100]00  <1> WT4:  CMP    byte [DISK_STATUS1],0     ; SET CONDITION CODE FOR CALLER
5054 00004A11 C3                <1>       RETn
5055                            <1>
5056                            <1> ;---------------------------------------
5057                            <1> ;     WAIT FOR CONTROLLER NOT BUSY     :
5058                            <1> ;---------------------------------------
5059                            <1> NOT_BUSY:
5060 00004A12 FB                <1>       STI                        ; MAKE SURE INTERRUPTS ARE ON
5061                            <1>       ;PUSH  eBX
5062                            <1>       ;SUB   CX,CX               ; SET INITIAL DELAY BEFORE TEST
5063 00004A13 668B15[E85C0000]  <1>       mov    DX, [HF_PORT]
5064 00004A1A 80C207            <1>       add    dl, 7               ; Status port (HF_PORT+7)
5065                            <1>       ;MOV   BL,DELAY_1
5066                            <1>                                  ; wait for 10 seconds
5067                            <1>       ;mov   cx, WAIT_HDU_INT_LO ; 1615h
5068                            <1>       ;;mov  bl, WAIT_HDU_INT_HI ;   05h
5069                            <1>       ;mov   bl, WAIT_HDU_INT_HI + 1
5070 00004A1D B915160500        <1>       mov    ecx, WAIT_HDU_INT_LH  ; 21/02/2015
5071                            <1>       ;
5072                            <1> ;;      mov     byte [wait_count], 0    ; Reset wait counter
5073                            <1> NB1:
5074 00004A22 EC                <1>       IN     AL,DX               ; CHECK STATUS
5075                            <1>       ;TEST  AL,ST_BUSY
5076 00004A23 2480              <1>       and    al, ST_BUSY
5077                            <1>       ;LOOPNZ      NB1
5078 00004A25 7410              <1>       JZ     short NB2           ; NOT BUSY--LETS GO
5079                            <1>       ;DEC   BL
5080                            <1>       ;JNZ   short NB1           ; KEEP TRYING FOR A WHILE
5081                            <1>
5082 00004A27 E461              <1> NB1_hi: IN   AL,SYS1                   ; wait for hi to lo
5083 00004A29 A810              <1>       TEST   AL,010H                   ; transition on memory
5084 00004A2B 75FA              <1>       JNZ    SHORT NB1_hi        ; refresh.
5085 00004A2D E461              <1> NB1_lo: IN   AL,SYS1
5086 00004A2F A810              <1>       TEST   AL,010H
5087 00004A31 74FA              <1>       JZ     short NB1_lo
5088 00004A33 E2ED              <1>       LOOP   NB1
5089                            <1>       ;dec   bl
5090                            <1>       ;jnz   short NB1
5091                            <1>       ;
5092                            <1> ;;      cmp     byte [wait_count], 182  ; 10 seconds (182 timer ticks)
5093                            <1> ;;    jb     short NB1
```

```
5094                                <1>       ;
5095                                <1>       ;MOV   [DISK_STATUS1],TIME_OUT   ; REPORT TIME OUT ERROR
5096                                <1>       ;JMP   SHORT NB3
5097 00004A35 B080                  <1>       mov    al, TIME_OUT
5098                                <1> NB2:
5099                                <1>       ;MOV   byte [DISK_STATUS1],0
5100                                <1> ;NB3:
5101                                <1>       ;POP   eBX
5102 00004A37 A2[9B520100]          <1>       mov    [DISK_STATUS1], al  ;;; will be set after return
5103                                <1>       ;CMP   byte [DISK_STATUS1],0     ; SET CONDITION CODE FOR CALLER
5104 00004A3C 08C0                  <1>       or     al, al              ; (zf = 0 --> timeout)
5105 00004A3E C3                    <1>       RETn
5106                                <1>
5107                                <1> ;----------------------------------------
5108                                <1> ;     WAIT FOR DATA REQUEST            :
5109                                <1> ;----------------------------------------
5110                                <1> WAIT_DRQ:
5111                                <1>       ;MOV   CX,DELAY_3
5112                                <1>       ;MOV   DX,HF_PORT+7
5113 00004A3F 668B15[E85C0000]      <1>       mov    dx, [HF_PORT]
5114 00004A46 80C207                <1>       add    dl, 7
5115                                <1>       ;;MOV  bl, WAIT_HDU_DRQ_HI ; 0
5116                                <1>       ;MOV   cx, WAIT_HDU_DRQ_LO ; 1000 (30 milli seconds)
5117                                <1>                                  ; (but it is written as 2000
5118                                <1>                                  ; micro seconds in ATORGS.ASM file
5119                                <1>                                  ; of Award Bios - 1999, D1A0622)
5120 00004A49 B9E8030000            <1>       mov    ecx, WAIT_HDU_DRQ_LH ; 21/02/2015
5121 00004A4E EC                    <1> WQ_1: IN     AL,DX               ; GET STATUS
5122 00004A4F A808                  <1>       TEST   AL,ST_DRQ           ; WAIT FOR DRQ
5123 00004A51 7516                  <1>       JNZ    short WQ_OK
5124                                <1>       ;LOOP  WQ_1                ; KEEP TRYING FOR A SHORT WHILE
5125                                <1> WQ_hi:
5126 00004A53 E461                  <1>       IN     AL,SYS1                   ; wait for hi to lo
5127 00004A55 A810                  <1>       TEST   AL,010H                   ; transition on memory
5128 00004A57 75FA                  <1>       JNZ    SHORT WQ_hi         ; refresh.
5129 00004A59 E461                  <1> WQ_lo:  IN      AL,SYS1
5130 00004A5B A810                  <1>       TEST   AL,010H
5131 00004A5D 74FA                  <1>       JZ     SHORT WQ_lo
5132 00004A5F E2ED                  <1>       LOOP   WQ_1
5133                                <1>
5134 00004A61 C605[9B520100]80      <1>         MOV     byte [DISK_STATUS1],TIME_OUT ; ERROR
5135 00004A68 F9                    <1>       STC
5136                                <1> WQ_OK:
5137 00004A69 C3                    <1>       RETn
5138                                <1> ;WQ_OK:      ;CLC
5139                                <1> ;     RETn
5140                                <1>
5141                                <1> ;----------------------------------------
5142                                <1> ;     CHECK FIXED DISK STATUS   :
5143                                <1> ;----------------------------------------
5144                                <1> CHECK_STATUS:
5145 00004A6A E813000000            <1>       CALL   CHECK_ST            ; CHECK THE STATUS BYTE
5146 00004A6F 7509                  <1>       JNZ    short CHECK_S1             ; AN ERROR WAS FOUND
5147 00004A71 A801                  <1>       TEST   AL,ST_ERROR         ; WERE THERE ANY OTHER ERRORS
5148 00004A73 7405                  <1>       JZ     short CHECK_S1             ; NO ERROR REPORTED
5149 00004A75 E849000000            <1>       CALL   CHECK_ER            ; ERROR REPORTED
5150                                <1> CHECK_S1:
5151 00004A7A 803D[9B520100]00      <1>       CMP    byte [DISK_STATUS1],0     ; SET STATUS FOR CALLER
5152 00004A81 C3                    <1>       RETn
5153                                <1>
5154                                <1> ;----------------------------------------
5155                                <1> ;     CHECK FIXED DISK STATUS BYTE     :
5156                                <1> ;----------------------------------------
5157                                <1> CHECK_ST:
5158                                <1>       ;MOV   DX,HF_PORT+7        ; GET THE STATUS
5159 00004A82 668B15[E85C0000]      <1>       mov    dx, [HF_PORT]
5160 00004A89 80C207                <1>       add    dl, 7
5161                                <1>
5162                                <1>       ; 17/02/2016
5163                                <1>       ;(http://wiki.osdev.org/ATA_PIO_Mode)
5164                                <1>       ;"delay 400ns to allow drive to set new values of BSY and DRQ"
5165 00004A8C EC                    <1>       IN     AL,DX
5166                                <1>       ;in    al, dx ; 100ns
5167                                <1>       ;in    al, dx ; 100ns
5168                                <1>       ;in    al, dx ; 100ns
5169                                <1>       NEWIODELAY ; 18/02/2016 (AWARD BIOS - 1999, 'CKST' in AHSDK.ASM)
5169 00004A8D E6EB                  <2>  out 0ebh,al
5170                                <1>       ;
5171 00004A8F A2[91520100]          <1>       MOV    [HF_STATUS],AL
5172 00004A94 B400                  <1>       MOV    AH,0
5173 00004A96 A880                  <1>       TEST   AL,ST_BUSY          ; IF STILL BUSY
5174 00004A98 751A                  <1>       JNZ    short CKST_EXIT           ;  REPORT OK
5175 00004A9A B4CC                  <1>       MOV    AH,WRITE_FAULT
5176 00004A9C A820                  <1>       TEST   AL,ST_WRT_FLT       ; CHECK FOR WRITE FAULT
5177 00004A9E 7514                  <1>       JNZ    short CKST_EXIT
5178 00004AA0 B4AA                  <1>       MOV    AH,NOT_RDY
5179 00004AA2 A840                  <1>       TEST   AL,ST_READY         ; CHECK FOR NOT READY
5180 00004AA4 740E                  <1>       JZ     short CKST_EXIT
5181 00004AA6 B440                  <1>       MOV    AH,BAD_SEEK
5182 00004AA8 A810                  <1>       TEST   AL,ST_SEEK_COMPL    ; CHECK FOR SEEK NOT COMPLETE
5183 00004AAA 7408                  <1>       JZ     short CKST_EXIT
5184 00004AAC B411                  <1>       MOV    AH,DATA_CORRECTED
5185 00004AAE A804                  <1>       TEST   AL,ST_CORRCTD       ; CHECK FOR CORRECTED ECC
5186 00004AB0 7502                  <1>       JNZ    short CKST_EXIT
5187 00004AB2 B400                  <1>       MOV    AH,0
5188                                <1> CKST_EXIT:
5189 00004AB4 8825[9B520100]        <1>       MOV    [DISK_STATUS1],AH   ; SET ERROR FLAG
5190 00004ABA 80FC11                <1>       CMP    AH,DATA_CORRECTED   ; KEEP GOING WITH DATA CORRECTED
5191 00004ABD 7403                  <1>       JZ     short CKST_EX1
5192 00004ABF 80FC00                <1>       CMP    AH,0
5193                                <1> CKST_EX1:
5194 00004AC2 C3                    <1>       RETn
```

```
                                <1>
                                <1> ;----------------------------------------
                                <1> ;     CHECK FIXED DISK ERROR REGISTER :
                                <1> ;----------------------------------------
                                <1> CHECK_ER:
                                <1>       ;MOV   DX, HF_PORT+1       ; GET THE ERROR REGISTER
00004AC3 668B15[E85C0000]       <1>       mov    dx, [HF_PORT]       ;
00004ACA FEC2                   <1>       inc    dl
00004ACC EC                     <1>       IN     AL,DX
00004ACD A2[92520100]           <1>       MOV    [HF_ERROR],AL
00004AD2 53                     <1>       PUSH   eBX ; 21/02/2015
00004AD3 B908000000             <1>       MOV    eCX,8               ; TEST ALL 8 BITS
00004AD8 D0E0                   <1> CK1:  SHL    AL,1                ; MOVE NEXT ERROR BIT TO CARRY
00004ADA 7202                   <1>       JC     short CK2           ; FOUND THE ERROR
00004ADC E2FA                   <1>       LOOP   CK1                 ; KEEP TRYING
00004ADE BB[DC5C0000]           <1> CK2:  MOV    eBX, ERR_TBL        ; COMPUTE ADDRESS OF
00004AE3 01CB                   <1>       ADD    eBX,eCX                  ; ERROR CODE
                                <1>       ;;MOV  AH,BYTE [CS:BX]          ; GET ERROR CODE
                                <1>       ;mov   ah, [bx]
00004AE5 8A23                   <1>       mov    ah, [ebx] ; 21/02/2015
00004AE7 8825[9B520100]         <1> CKEX: MOV    [DISK_STATUS1],AH   ; SAVE ERROR CODE
00004AED 5B                     <1>       POP    eBX
00004AEE 80FC00                 <1>       CMP    AH,0
00004AF1 C3                     <1>       RETn
                                <1>
                                <1> ;--------------------------------------------------------
                                <1> ; CHECK_DMA                                 :
                                <1> ;  -CHECK ES:BX AND # SECTORS TO MAKE SURE THAT IT WILL :
                                <1> ;   FIT WITHOUT SEGMENT OVERFLOW.                       :
                                <1> ;  -ES:BX HAS BEEN REVISED TO THE FORMAT SSSS:000X  :
                                <1> ;  -OK IF # SECTORS < 80H (7FH IF LONG READ OR WRITE)     :
                                <1> ;  -OK IF # SECTORS = 80H (7FH) AND BX <= 00H (04H) :
                                <1> ;  -ERROR OTHERWISE                                     :
                                <1> ;--------------------------------------------------------
                                <1> CHECK_DMA:
00004AF2 6650                   <1>       PUSH   AX                  ; SAVE REGISTERS
00004AF4 66B80080               <1>       MOV    AX,8000H            ; AH = MAX # SECTORS AL = MAX OFFSET
00004AF8 F645FE02               <1>         TEST    byte [CMD_BLOCK+6],ECC_MODE
00004AFC 7404                   <1>       JZ     short CKD1
00004AFE 66B8047F               <1>       MOV    AX,7F04H            ; ECC IS 4 MORE BYTES
00004B02 3A65F9                 <1> CKD1: CMP    AH, [CMD_BLOCK+1]   ; NUMBER OF SECTORS
00004B05 7706                   <1>       JA     short CKDOK         ; IT WILL FIT
00004B07 7208                   <1>       JB     short CKDERR        ; TOO MANY
00004B09 38D8                   <1>       CMP    AL,BL               ; CHECK OFFSET ON MAX SECTORS
00004B0B 7204                   <1>       JB     short CKDERR        ; ERROR
00004B0D F8                     <1> CKDOK:       CLC                       ; CLEAR CARRY
00004B0E 6658                   <1>       POP    AX
00004B10 C3                     <1>       RETn                       ; NORMAL RETURN
00004B11 F9                     <1> CKDERR: STC                      ; INDICATE ERROR
00004B12 C605[9B520100]09       <1>         MOV     byte [DISK_STATUS1],DMA_BOUNDARY
00004B19 6658                   <1>       POP    AX
00004B1B C3                     <1>       RETn
                                <1>
                                <1> ;----------------------------------------
                                <1> ;     SET UP ES:BX-> DISK PARMS :
                                <1> ;----------------------------------------
                                <1>
                                <1> ; INPUT -> DL = 0 based drive number
                                <1> ; OUTPUT -> ES:BX = disk parameter table address
                                <1>
                                <1> GET_VEC:
                                <1>       ;SUB   AX,AX               ; GET DISK PARAMETER ADDRESS
                                <1>       ;MOV   ES,AX
                                <1>       ;TEST  DL,1
                                <1>       ;JZ    short GV_0
                                <1> ;     LES    BX,[HF1_TBL_VEC]    ; ES:BX -> DRIVE PARAMETERS
                                <1> ;     JMP    SHORT GV_EXIT
                                <1> ;GV_0:
                                <1> ;     LES    BX,[HF_TBL_VEC]           ; ES:BX -> DRIVE PARAMETERS
                                <1> ;
                                <1>       ;xor   bh, bh
00004B1C 31DB                   <1>       xor    ebx, ebx
00004B1E 88D3                   <1>       mov    bl, dl
                                <1>       ;;02/01/2015
                                <1>       ;;shl  bl, 1               ; port address offset
                                <1>       ;;mov  ax, [bx+hd_ports]   ; Base port address (1F0h, 170h)
                                <1>       ;;shl  bl, 1               ; dpt pointer offset
00004B20 C0E302                 <1>       shl    bl, 2  ;;
                                <1>       ;add   bx, HF_TBL_VEC            ; Disk parameter table pointer
00004B23 81C3[A0520100]         <1>       add    ebx, HF_TBL_VEC ; 21/02/2015
                                <1>       ;push  word [bx+2]         ; dpt segment
                                <1>       ;pop   es
                                <1>       ;mov   bx, [bx]            ; dpt offset
00004B29 8B1B                   <1>       mov    ebx, [ebx]
                                <1> ;GV_EXIT:
00004B2B C3                     <1>       RETn
                                <1>
                                <1> hdc1_int: ; 21/02/2015
                                <1> ;--- HARDWARE INT 76H -- ( IRQ LEVEL  14 ) ----------------------
                                <1> ;                                                       :
                                <1> ;     FIXED DISK INTERRUPT ROUTINE                            :
                                <1> ;                                                       :
                                <1> ;-------------------------------------------------------------
                                <1>
                                <1> ; 22/12/2014
                                <1> ; IBM PC-XT Model 286 System BIOS Source Code - DISK.ASM (HD_INT)
                                <1> ;      '11/15/85'
                                <1> ; AWARD BIOS 1999 (D1A0622)
                                <1> ;     Source Code - ATORGS.ASM (INT_HDISK, INT_HDISK1)
                                <1>
                                <1> ;int_76h:
                                <1> HD_INT:
```

```
5297 00004B2C 6650                <1>       PUSH   AX
5298 00004B2E 1E                  <1>       PUSH   DS
5299                              <1>       ;CALL  DDS
5300                              <1>       ; 21/02/2015 (32 bit, 386 pm modification)
5301 00004B2F 66B81000            <1>       mov    ax, KDATA
5302 00004B33 8ED8                <1>       mov    ds, ax
5303                              <1>       ;
5304                              <1>       ;;MOV  @HF_INT_FLAG,0FFH   ; ALL DONE
5305                              <1>         ;mov     byte [CS:HF_INT_FLAG], 0FFh
5306 00004B35 C605[93520100]FF    <1>       mov    byte [HF_INT_FLAG], 0FFh
5307                              <1>       ;
5308 00004B3C 6652                <1>       push   dx
5309 00004B3E 66BAF701            <1>       mov    dx, HDC1_BASEPORT+7 ; Status Register (1F7h)
5310                              <1>                                  ; Clear Controller
5311                              <1> Clear_IRQ1415:                        ; (Award BIOS - 1999)
5312 00004B42 EC                  <1>       in     al, dx              ;
5313 00004B43 665A                <1>       pop    dx
5314                              <1>       NEWIODELAY
5314 00004B45 E6EB                <2>  out 0ebh,al
5315                              <1>       ;
5316 00004B47 B020                <1>       MOV    AL,EOI              ; NON-SPECIFIC END OF INTERRUPT
5317 00004B49 E6A0                <1>       OUT    INTB00,AL           ; FOR CONTROLLER #2
5318                              <1>       ;JMP   $+2                 ; WAIT
5319                              <1>       NEWIODELAY
5319 00004B4B E6EB                <2>  out 0ebh,al
5320 00004B4D E620                <1>       OUT    INTA00,AL           ; FOR CONTROLLER #1
5321 00004B4F 1F                  <1>       POP    DS
5322                              <1>       ;STI                       ; RE-ENABLE INTERRUPTS
5323                              <1>       ;MOV   AX,9100H            ; DEVICE POST
5324                              <1>       ;INT   15H                 ;  INTERRUPT
5325                              <1> irq15_iret: ; 25/02/2015
5326 00004B50 6658                <1>       POP    AX
5327 00004B52 CF                  <1>       IRETd                      ; RETURN FROM INTERRUPT
5328                              <1>
5329                              <1> hdc2_int: ; 21/02/2015
5330                              <1> ;++++ HARDWARE INT 77H ++ ( IRQ LEVEL  15 ) ++++++++++++++++++++
5331                              <1> ;                                                    :
5332                              <1> ;     FIXED DISK INTERRUPT ROUTINE                        :
5333                              <1> ;                                                    :
5334                              <1> ;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
5335                              <1>
5336                              <1> ;int_77h:
5337                              <1> HD1_INT:
5338 00004B53 6650                <1>       PUSH   AX
5339                              <1>       ; Check if that is a spurious IRQ (from slave PIC)
5340                              <1>       ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
5341 00004B55 B00B                <1>       mov    al, 0Bh  ; In-Service Register
5342 00004B57 E6A0                <1>       out    0A0h, al
5343 00004B59 EB00                <1>         jmp short $+2
5344 00004B5B EB00                <1>       jmp short $+2
5345 00004B5D E4A0                <1>       in     al, 0A0h
5346 00004B5F 2480                <1>       and    al, 80h ; bit 7 (is it real IRQ 15 or fake?)
5347 00004B61 74ED                <1>       jz     short irq15_iret ; Fake (spurious)IRQ, do not send EOI)
5348                              <1>       ;
5349 00004B63 1E                  <1>       PUSH   DS
5350                              <1>       ;CALL  DDS
5351                              <1>       ; 21/02/2015 (32 bit, 386 pm modification)
5352 00004B64 66B81000            <1>       mov    ax, KDATA
5353 00004B68 8ED8                <1>       mov    ds, ax
5354                              <1>       ;
5355                              <1>       ;;MOV  @HF_INT_FLAG,0FFH   ; ALL DONE
5356                              <1>         ;or      byte [CS:HF_INT_FLAG],0C0h
5357 00004B6A 800D[93520100]C0    <1>       or     byte [HF_INT_FLAG], 0C0h
5358                              <1>       ;
5359 00004B71 6652                <1>       push   dx
5360 00004B73 66BA7701            <1>       mov    dx, HDC2_BASEPORT+7 ; Status Register (177h)
5361                              <1>                                  ; Clear Controller (Award BIOS 1999)
5362 00004B77 EBC9                <1>       jmp    short Clear_IRQ1415
5363                              <1>
5364                              <1>
5365                              <1> ;%include 'diskdata.inc' ; 11/03/2015
5366                              <1> ;%include 'diskbss.inc' ; 11/03/2015
5367                              <1>
5368                              <1>
5369                              <1> ;////////////////////////////////////////////////////////////////
5370                              <1> ;; END OF DISK I/O SYTEM ///
2158                                  %include 'memory.s'  ; 09/03/2015
   1                              <1> ; ****************************************************************************
   2                              <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - memory.s
   3                              <1> ; ----------------------------------------------------------------------------
   4                              <1> ; Last Update: 22/07/2017
   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> ; 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
38                                  <1> PAGE_OFF     equ 0FFFh          ; 12 bit byte offset in page frame
39                                  <1> PTE_MASK     equ 03FFh          ; page table entry mask
40                                  <1> PTE_DUPLICATED  equ 200h        ; duplicated page sign (AVL bit 0)
41                                  <1> PDE_A_CLEAR  equ 0F000h         ; to clear PDE attribute bits
42                                  <1> PTE_A_CLEAR  equ 0F000h         ; to clear PTE attribute bits
43                                  <1> LOGIC_SECT_SIZE equ 512              ; logical sector size
44                                  <1> ERR_MAJOR_PF equ 0E0h           ; major error: page fault
45                                  <1> ERR_MINOR_IM equ 4 ;15/10/2016 (1->4); insufficient (out of) memory
46                                  <1> ERR_MINOR_PV equ 6 ;15/10/2016 (1->4); protection violation
47                                  <1> SWP_DISK_READ_ERR     equ 40
48                                  <1> SWP_DISK_NOT_PRESENT_ERR   equ 41
49                                  <1> SWP_SECTOR_NOT_PRESENT_ERR equ 42
50                                  <1> SWP_NO_FREE_SPACE_ERR      equ 43
51                                  <1> SWP_DISK_WRITE_ERR         equ 44
52                                  <1> SWP_NO_PAGE_TO_SWAP_ERR    equ 45
53                                  <1> PTE_A_ACCESS_BIT equ 5  ; Bit 5 (accessed flag)
54                                  <1> SECTOR_SHIFT     equ 3  ; sector shift (to convert page block number)
55                                  <1> ; 12/07/2016
56                                  <1> PTE_SHARED    equ 400h          ; AVL bit 1, direct memory access bit
57                                  <1>                                 ; (Indicates that the page is not allocated
58                                  <1>                                   ; for the process, it is a shared or system
59                                  <1>                                          ; page, it must not be deallocated!)
60                                  <1> ;
61                                  <1> ;; Retro Unix 386 v1 - paging method/principles
62                                  <1> ;;
63                                  <1> ;; 10/10/2014
64                                  <1> ;; RETRO UNIX 386 v1 - PAGING METHOD/PRINCIPLES
65                                  <1> ;;
66                                  <1> ;; KERNEL PAGE MAP: 1 to 1 physical memory page map
67                                  <1> ;;    (virtual address = physical address)
68                                  <1> ;; KERNEL PAGE TABLES:
69                                  <1> ;;    Kernel page directory and all page tables are
70                                  <1> ;;    on memory as initialized, as equal to physical memory
71                                  <1> ;;    layout. Kernel pages can/must not be swapped out/in.
72                                  <1> ;;
73                                  <1> ;;    what for: User pages may be swapped out, when accessing
74                                  <1> ;;    a page in kernel/system mode, if it would be swapped out,
75                                  <1> ;;    kernel would have to swap it in! But it is also may be
76                                  <1> ;;    in use by a user process. (In system/kernel mode
77                                  <1> ;;    kernel can access all memory pages even if they are
78                                  <1> ;;    reserved/allocated for user processes. Swap out/in would
79                                  <1> ;;    cause conflicts.)
80                                  <1> ;;
81                                  <1> ;;    As result of these conditions,
82                                  <1> ;;    all kernel pages must be initialized as equal to
83                                  <1> ;;    physical layout for preventing page faults.
84                                  <1> ;;    Also, calling "allocate page" procedure after
85                                  <1> ;;    a page fault can cause another page fault (double fault)
86                                  <1> ;;    if all kernel page tables would not be initialized.
87                                  <1> ;;
88                                  <1> ;;    [first_page] = Beginning of users space, as offset to
89                                  <1> ;;    memory allocation table. (double word aligned)
90                                  <1> ;;
91                                  <1> ;;    [next_page] = first/next free space to be searched
92                                  <1> ;;    as offset to memory allocation table. (dw aligned)
93                                  <1> ;;
94                                  <1> ;;    [last_page] = End of memory (users space), as offset
95                                  <1> ;;    to memory allocation table. (double word aligned)
96                                  <1> ;;
97                                  <1> ;; USER PAGE TABLES:
98                                  <1> ;;    Demand paging (& 'copy on write' allocation method) ...
99                                  <1> ;;           'ready only' marked copies of the
100                                 <1> ;;           parent process's page table entries (for
101                                 <1> ;;           same physical memory).
102                                 <1> ;;           (A page will be copied to a new page after
103                                 <1> ;;            if it causes R/W page fault.)
104                                 <1> ;;
105                                 <1> ;;    Every user process has own (different)
106                                 <1> ;;    page directory and page tables.
107                                 <1> ;;
108                                 <1> ;;    Code starts at virtual address 0, always.
109                                 <1> ;;    (Initial value of EIP is 0 in user mode.)
110                                 <1> ;;    (Programs can be written/developed as simple
111                                 <1> ;;     flat memory programs.)
112                                 <1> ;;
113                                 <1> ;; MEMORY ALLOCATION STRATEGY:
114                                 <1> ;;    Memory page will be allocated by kernel only
115                                 <1> ;;           (in kernel/system mode only).
116                                 <1> ;;    * After a
117                                 <1> ;;      - 'not present' page fault
118                                 <1> ;;      - 'writing attempt on read only page' page fault
119                                 <1> ;;    * For loading (opening, reading) a file or disk/drive
120                                 <1> ;;    * As responce to 'allocate additional memory blocks'
121                                 <1> ;;      request by running process.
122                                 <1> ;;    * While creating a process, allocating a new buffer,
123                                 <1> ;;      new page tables etc.
124                                 <1> ;;
125                                 <1> ;;    At first,
126                                 <1> ;;    - 'allocate page' procedure will be called;
127                                 <1> ;;       if it will return with a valid (>0) physical address
```

```
128                     <1> ;;       (that means the relevant M.A.T. bit has been RESET)
129                     <1> ;;       relevant memory page/block will be cleared (zeroed).
130                     <1> ;;    - 'allocate page' will be called for allocating page
131                     <1> ;;       directory, page table and running space (data/code).
132                     <1> ;;    - every successful 'allocate page' call will decrease
133                     <1> ;;      'free_pages' count (pointer).
134                     <1> ;;    - 'out of (insufficient) memory error' will be returned
135                     <1> ;;      if 'free_pages' points to a ZERO.
136                     <1> ;;    - swapping out and swapping in (if it is not a new page)
137                     <1> ;;      procedures will be called as responce to 'out of memory'
138                     <1> ;;      error except errors caused by attribute conflicts.
139                     <1> ;;     (swapper functions)
140                     <1> ;;
141                     <1> ;;    At second,
142                     <1> ;;    - page directory entry will be updated then page table
143                     <1> ;;      entry will be updated.
144                     <1> ;;
145                     <1> ;; MEMORY ALLOCATION TABLE FORMAT:
146                     <1> ;;    - M.A.T. has a size according to available memory as
147                     <1> ;;      follows:
148                     <1> ;;             - 1 (allocation) bit per 1 page (4096 bytes)
149                     <1> ;;             - a bit with value of 0 means allocated page
150                     <1> ;;             - a bit with value of 1 means a free page
151                     <1> ;,    - 'free_pages' pointer holds count of free pages
152                     <1> ;;      depending on M.A.T.
153                     <1> ;;          (NOTE: Free page count will not be checked
154                     <1> ;;          again -on M.A.T.- after initialization.
155                     <1> ;;          Kernel will trust on initial count.)
156                     <1> ;,    - 'free_pages' count will be decreased by allocation
157                     <1> ;;      and it will be increased by deallocation procedures.
158                     <1> ;;
159                     <1> ;;    - Available memory will be calculated during
160                     <1> ;;      the kernel's initialization stage (in real mode).
161                     <1> ;;      Memory allocation table and kernel page tables
162                     <1> ;;      will be formatted/sized as result of available
163                     <1> ;;      memory calculation before paging is enabled.
164                     <1> ;;
165                     <1> ;; For 4GB Available/Present Memory: (max. possible memory size)
166                     <1> ;;    - Memory Allocation Table size will be 128 KB.
167                     <1> ;;    - Memory allocation for kernel page directory size
168                     <1> ;;      is always 4 KB. (in addition to total allocation size
169                     <1> ;;      for page tables)
170                     <1> ;;    - Memory allocation for kernel page tables (1024 tables)
171                     <1> ;;      is 4 MB (1024*4*1024 bytes).
172                     <1> ;;    - User (available) space will be started
173                     <1> ;;      at 6th MB of the memory (after 1MB+4MB).
174                     <1> ;;    - The first 640 KB is for kernel's itself plus
175                     <1> ;;      memory allocation table and kernel's page directory
176                     <1> ;;      (D0000h-EFFFFh may be used as kernel space...)
177                     <1> ;;    - B0000h to B7FFFh address space (32 KB) will be used
178                     <1> ;;      for buffers.
179                     <1> ;;    - ROMBIOS, VIDEO BUFFER and VIDEO ROM space are reserved.
180                     <1> ;,      (A0000h-AFFFFh, C0000h-CFFFFh, F0000h-FFFFFh)
181                     <1> ;;    - Kernel page tables start at 100000h (2nd MB)
182                     <1> ;;
183                     <1> ;; For 1GB Available Memory:
184                     <1> ;;    - Memory Allocation Table size will be 32 KB.
185                     <1> ;;    - Memory allocation for kernel page directory size
186                     <1> ;;      is always 4 KB. (in addition to total allocation size
187                     <1> ;;      for page tables)
188                     <1> ;;    - Memory allocation for kernel page tables (256 tables)
189                     <1> ;;      is 1 MB (256*4*1024 bytes).
190                     <1> ;;    - User (available) space will be started
191                     <1> ;;      at 3th MB of the memory (after 1MB+1MB).
192                     <1> ;;    - The first 640 KB is for kernel's itself plus
193                     <1> ;;      memory allocation table and kernel's page directory
194                     <1> ;;      (D0000h-EFFFFh may be used as kernel space...)
195                     <1> ;;    - B0000h to B7FFFh address space (32 KB) will be used
196                     <1> ;;      for buffers.
197                     <1> ;;    - ROMBIOS, VIDEO BUFFER and VIDEO ROM space are reserved.
198                     <1> ;,      (A0000h-AFFFFh, C0000h-CFFFFh, F0000h-FFFFFh)
199                     <1> ;;    - Kernel page tables start at 100000h (2nd MB).
200                     <1> ;;
201                     <1> ;;
202                     <1>
203                     <1>
204                     <1> ;;*****************************************************************************
205                     <1> ;;
206                     <1> ;; RETRO UNIX 386 v1 - Paging (Method for Copy On Write paging principle)
207                     <1> ;; DEMAND PAGING - PARENT&CHILD PAGE TABLE DUPLICATION PRINCIPLES (23/04/2015)
208                     <1>
209                     <1> ;; Main factor: "sys fork" system call
210                     <1> ;;
211                     <1> ;;          FORK
212                     <1> ;;                      |----> parent - duplicated PTEs, read only pages
213                     <1> ;;  writable pages ---->|
214                     <1> ;;                      |----> child - duplicated PTEs, read only pages
215                     <1> ;;
216                     <1> ;; AVL bit (0) of Page Table Entry is used as duplication sign
217                     <1> ;;
218                     <1> ;; AVL Bit 0 [PTE Bit 9] = 'Duplicated PTE belongs to child' sign/flag (if it is set)
219                     <1> ;; Note: Dirty bit (PTE bit 6) may be used instead of AVL bit 0 (PTE bit 9)
220                     <1> ;;       -while R/W bit is 0-.
221                     <1> ;;
222                     <1> ;; Duplicate page tables with writable pages (the 1st sys fork in the process):
223                     <1> ;; # Parent's Page Table Entries are updated to point same pages as read only,
224                     <1> ;;   as duplicated PTE bit  -AVL bit 0, PTE bit 9- are reset/clear.
225                     <1> ;; # Then Parent's Page Table is copied to Child's Page Table.
226                     <1> ;; # Child's Page Table Entries are updated as duplicated child bit
227                     <1> ;;   -AVL bit 0, PTE bit 9- is set.
228                     <1> ;;
229                     <1> ;; Duplicate page tables with read only pages (several sys fork system calls):
```

```
                        <1> ;; # Parent's read only pages are copied to new child pages.
                        <1> ;;   Parent's PTE attributes are not changed.
                        <1> ;;   (Because, there is another parent-child fork before this fork! We must not
                        <1> ;;    destroy/mix previous fork result).
                        <1> ;; # Child's Page Table Entries (which are corresponding to Parent's
                        <1> ;;   read only pages) are set as writable (while duplicated PTE bit is clear).
                        <1> ;; # Parent's PTEs with writable page attribute are updated to point same pages
                        <1> ;;   as read only, (while) duplicated PTE bit is reset (clear).
                        <1> ;; # Parent's Page Table Entries (with writable page attribute) are duplicated
                        <1> ;;   as Child's Page Table Entries without copying actual page.
                        <1> ;; # Child 's Page Table Entries (which are corresponding to Parent's writable
                        <1> ;;   pages) are updated as duplicated PTE bit (AVL bit 0, PTE bit 9- is set.
                        <1> ;;
                        <1> ;; !? WHAT FOR (duplication after duplication):
                        <1> ;; In UNIX method for sys fork (a typical 'fork' application in /etc/init)
                        <1> ;; program/executable code continues from specified location as child process,
                        <1> ;; returns back previous code location as parent process, every child after
                        <1> ;; every sys fork uses last image of code and data just prior the fork.
                        <1> ;; Even if the parent code changes data, the child will not see the changed data
                        <1> ;; after the fork. In Retro UNIX 8086 v1, parent's process segment (32KB)
                        <1> ;; was copied to child's process segment (all of code and data) according to
                        <1> ;; original UNIX v1 which copies all of parent process code and data -core-
                        <1> ;; to child space -core- but swaps that core image -of child- on to disk.
                        <1> ;; If I (Erdogan Tan) would use a method of to copy parent's core
                        <1> ;; (complete running image of parent process) to the child process;
                        <1> ;; for big sizes, i would force Retro UNIX 386 v1 to spend many memory pages
                        <1> ;; and times only for a sys fork. (It would excessive reservation for sys fork,
                        <1> ;; because sys fork usually is prior to sys exec; sys exec always establishes
                        <1> ;; a new/fresh core -running space-, by clearing all code/data content).
                        <1> ;; 'Read Only' page flag ensures page fault handler is needed only for a few write
                        <1> ;; attempts between sys fork and sys exec, not more... (I say so by thinking
                        <1> ;; of "/etc/init" content, specially.) sys exec will clear page tables and
                        <1> ;; new/fresh pages will be used to load and run new executable/program.
                        <1> ;; That is what for i have preferred "copy on write", "duplication" method
                        <1> ;; for sharing same read only pages between parent and child processes.
                        <1> ;; That is a pitty i have to use new private flag (AVL bit 0, "duplicated PTE
                        <1> ;; belongs to child" sign) for cooperation on duplicated pages between a parent
                        <1> ;; and it's child processes; otherwise parent process would destroy data belongs
                        <1> ;; to its child or vice versa; or some pages would remain unclaimed
                        <1> ;; -deallocation problem-.
                        <1> ;; Note: to prevent conflicts, read only pages must not be swapped out...
                        <1> ;;
                        <1> ;; WHEN PARENT TRIES TO WRITE IT'S READ ONLY (DUPLICATED) PAGE:
                        <1> ;; # Page fault handler will do those:
                        <1> ;;   - 'Duplicated PTE' flag (PTE bit 9) is checked (on the failed PTE).
                        <1> ;;   - If it is reset/clear, there is a child uses same page.
                        <1> ;;   - Parent's read only page -previous page- is copied to a new writable page.
                        <1> ;;   - Parent's PTE is updated as writable page, as unique page (AVL=0)
                        <1> ;;   - (Page fault handler whill check this PTE later, if child process causes to
                        <1> ;;     page fault due to write attempt on read only page. Of course, the previous
                        <1> ;;     read only page will be converted to writable and unique page which belongs
                        <1> ;;     to child process.)
                        <1> ;; WHEN CHILD TRIES TO WRITE IT'S READ ONLY (DUPLICATED) PAGE:
                        <1> ;; # Page fault handler will do those:
                        <1> ;;   - 'Duplicated PTE' flag (PTE bit 9) is checked (on the failed PTE).
                        <1> ;;   - If it is set, there is a parent uses -or was using- same page.
                        <1> ;;   - Same PTE address within parent's page table is checked if it has same page
                        <1> ;;     address or not.
                        <1> ;;   - If parent's PTE has same address, child will continue with a new writable page.
                        <1> ;;     Parent's PTE will point to same (previous) page as writable, unique (AVL=0).
                        <1> ;;   - If parent's PTE has different address, child will continue with it's
                        <1> ;;     own/same page but read only flag (0) will be changed to writable flag (1) and
                        <1> ;;     'duplicated PTE (belongs to child)' flag/sign will be cleared/reset.
                        <1> ;;
                        <1> ;; NOTE: When a child process is terminated, read only flags of parent's page tables
                        <1> ;;       will be set as writable (and unique) in case of child process was using
                        <1> ;;       same pages with duplicated child PTE sign... Depending on sys fork and
                        <1> ;;       duplication method details, it is not possible multiple child processes
                        <1> ;;       were using same page with duplicated PTEs.
                        <1> ;;
                        <1> ;;////////////////////////////////////////////////////////////////////////////////
                        <1>
                        <1> ;; 08/10/2014
                        <1> ;; 11/09/2014 - Retro UNIX 386 v1 PAGING (further) draft
                        <1> ;;           by Erdogan Tan (Based on KolibriOS 'memory.inc')
                        <1>
                        <1> ;; 'allocate_page' code is derived and modified from KolibriOS
                        <1> ;; 'alloc_page' procedure in 'memory.inc'
                        <1> ;; (25/08/2014, Revision: 5057) file
                        <1> ;; by KolibriOS Team (2004-2012)
                        <1>
                        <1> allocate_page:
                        <1>       ; 01/07/2015
                        <1>       ; 05/05/2015
                        <1>       ; 30/04/2015
                        <1>       ; 16/10/2014
                        <1>       ; 08/10/2014
                        <1>       ; 09/09/2014 (Retro UNIX 386 v1 - beginning)
                        <1>       ;
                        <1>       ; INPUT -> none
                        <1>       ;
                        <1>       ; OUTPUT ->
                        <1>       ;     EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
                        <1>       ;     (corresponding MEMORY ALLOCATION TABLE bit is RESET)
                        <1>       ;
                        <1>       ;     CF = 1 and EAX = 0
                        <1>       ;               if there is not a free page to be allocated
                        <1>       ;
                        <1>       ; Modified Registers -> none (except EAX)
                        <1>       ;
00004B79 A1[08520100]   <1>       mov   eax, [free_pages]
00004B7E 21C0           <1>       and   eax, eax
```

```
00004B80 7438                <1>       jz     short out_of_memory
                             <1>       ;
00004B82 53                  <1>       push   ebx
00004B83 51                  <1>       push   ecx
                             <1>       ;
00004B84 BB00001000          <1>       mov    ebx, MEM_ALLOC_TBL   ; Memory Allocation Table offset
00004B89 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)
00004B8B 031D[0C520100]      <1>       add    ebx, [next_page] ; Free page searching starts from here
                             <1>                                   ; next_free_page >> 5
00004B91 030D[10520100]      <1>       add    ecx, [last_page] ; Free page searching ends here
                             <1>                                   ; (total_pages - 1) >> 5
                             <1> al_p_scan:
00004B97 39CB                <1>       cmp    ebx, ecx
00004B99 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>       ;
00004B9B 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.
00004B9E 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!)
00004BA0 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.
00004BA3 EBF2                <1>         jmp     short al_p_scan
                             <1>       ;
                             <1> al_p_notfound:
00004BA5 81E900001000        <1>       sub    ecx, MEM_ALLOC_TBL
00004BAB 890D[0C520100]      <1>       mov    [next_page], ecx ; next/first free page = last page
                             <1>                               ; (deallocate_page procedure will change it)
00004BB1 31C0                <1>       xor    eax, eax
00004BB3 A3[08520100]        <1>       mov    [free_pages], eax ; 0
00004BB8 59                  <1>       pop    ecx
00004BB9 5B                  <1>       pop    ebx
                             <1>       ;
                             <1> out_of_memory:
00004BBA E85B040000          <1>       call   swap_out
00004BBF 7325                <1>       jnc    short al_p_ok  ; [free_pages] = 0, re-allocation by swap_out
                             <1>       ;
00004BC1 29C0                <1>       sub    eax, eax ; 0
00004BC3 F9                  <1>       stc
00004BC4 C3                  <1>       retn
                             <1>
                             <1> al_p_found:
00004BC5 89D9                <1>       mov    ecx, ebx
00004BC7 81E900001000        <1>       sub    ecx, MEM_ALLOC_TBL
00004BCD 890D[0C520100]      <1>       mov    [next_page], ecx ; Set first free page searching start
                             <1>                               ; address/offset (to the next)
00004BD3 FF0D[08520100]      <1>         dec     dword [free_pages] ; 1 page has been allocated (X = X-1)
                             <1>       ;
00004BD9 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)
00004BDC C1E103              <1>       shl    ecx, 3        ; (page block offset * 32) + page index
00004BDF 01C8                <1>       add    eax, ecx      ; = page number
00004BE1 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...
00004BE4 59                  <1>       pop    ecx
00004BE5 5B                  <1>       pop    ebx
                             <1> al_p_ok:
00004BE6 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>       ;
00004BE7 E88DFFFFFF             <1>       call  allocate_page
00004BEC 7216                   <1>       jc    short mkpd_error
                                <1>       ;
00004BEE 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>       ;
00004BF3 57                     <1>       push  edi
00004BF4 51                     <1>       push  ecx
00004BF5 50                     <1>       push  eax
00004BF6 B900040000             <1>       mov   ecx, PAGE_SIZE / 4
00004BFB 89C7                   <1>       mov   edi, eax
00004BFD 31C0                   <1>       xor   eax, eax
00004BFF F3AB                   <1>       rep   stosd
00004C01 58                     <1>       pop   eax
00004C02 59                     <1>       pop   ecx
00004C03 5F                     <1>       pop   edi
                                <1> mkpd_error:
                                <1> mkpt_error:
00004C04 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>       ;
00004C05 E86FFFFFFF             <1>       call  allocate_page
00004C0A 72F8                   <1>       jc    short mkpt_error
00004C0C E811000000             <1>       call  set_pde
00004C11 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>       ;
00004C13 E861FFFFFF             <1>       call  allocate_page
00004C18 7207                   <1>       jc    short mkp_err
00004C1A E821000000             <1>       call  set_pte
00004C1F 73D2                   <1>       jnc   short clear_page ; 18/04/2015
                                <1> mkp_err:
00004C21 C3                     <1>       retn
                                <1>
                                <1>
                                <1> set_pde:    ; Set page directory entry (PDE)
                                <1>       ; 20/07/2015
                                <1>       ; 18/04/2015
                                <1>       ; 12/04/2015
                                <1>       ; 23/10/2014
                                <1>       ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
```

```
                                <1>       ;
                                <1>       ; INPUT ->
                                <1>       ;     EAX = physical address
                                <1>       ;           (use present value if EAX = 0)
                                <1>       ;     EBX = virtual (linear) address
                                <1>       ;     ECX = page table attributes (lower 12 bits)
                                <1>       ;           (higher 20 bits must be ZERO)
                                <1>       ;           (bit 0 must be 1)
                                <1>       ;     u.pgdir = page directory (physical) address
                                <1>       ; OUTPUT ->
                                <1>       ;     EDX = PDE address
                                <1>       ;     EAX = page table address (physical)
                                <1>       ;     ;(CF=1 -> Invalid page address)
                                <1>       ;
                                <1>       ; Modified Registers -> EDX
                                <1>       ;
00004C22 89DA                   <1>       mov   edx, ebx
00004C24 C1EA16                 <1>       shr   edx, PAGE_D_SHIFT ; 22
00004C27 C1E202                 <1>       shl   edx, 2 ; offset to page directory (1024*4)
00004C2A 0315[B8030300]         <1>       add   edx, [u.pgdir]
                                <1>       ;
00004C30 21C0                   <1>       and   eax, eax
00004C32 7506                   <1>       jnz   short spde_1
                                <1>       ;
00004C34 8B02                   <1>       mov   eax, [edx]  ; old PDE value
                                <1>       ;test  al, 1
                                <1>       ;jz    short spde_2
00004C36 662500F0               <1>       and   ax, PDE_A_CLEAR ; 0F000h  ; clear lower 12 bits
                                <1> spde_1:
                                <1>       ;and   cx, 0FFFh
00004C3A 8902                   <1>       mov   [edx], eax
00004C3C 66090A                 <1>       or    [edx], cx
00004C3F 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>       ;
00004C40 50                     <1>       push  eax
00004C41 A1[B8030300]           <1>       mov   eax, [u.pgdir] ; 20/07/2015
00004C46 E837000000             <1>       call  get_pde
                                <1>             ; EDX = PDE address
                                <1>             ; EAX = PDE value
00004C4B 5A                     <1>       pop   edx ; physical page address
00004C4C 722A                   <1>       jc    short spte_err ; PDE not present
                                <1>       ;
00004C4E 53                     <1>       push  ebx ; 24/07/2015
00004C4F 662500F0               <1>       and   ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 bits
                                <1>                       ; EDX = PT address (physical)
00004C53 C1EB0C                 <1>       shr   ebx, PAGE_SHIFT ; 12
00004C56 81E3FF030000           <1>       and   ebx, PTE_MASK; 03FFh
                                <1>                     ; clear higher 10 bits (PD bits)
00004C5C C1E302                 <1>       shl   ebx, 2   ; offset to page table (1024*4)
00004C5F 01C3                   <1>       add   ebx, eax
                                <1>       ;
00004C61 8B03                   <1>       mov   eax, [ebx] ; Old PTE value
00004C63 A801                   <1>       test  al, 1
00004C65 740C                   <1>       jz    short spte_0
00004C67 09D2                   <1>       or    edx, edx
00004C69 750F                   <1>       jnz   short spte_1
00004C6B 662500F0               <1>       and   ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 bits
00004C6F 89C2                   <1>       mov   edx, eax
00004C71 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!
00004C73 21C0                   <1>       and   eax, eax
00004C75 7403                   <1>       jz    short spte_1
                                <1>       ; 24/07/2015
                                <1>       ; swapped page ! (on disk)
00004C77 5B                     <1>       pop   ebx
                                <1> spte_err:
00004C78 F9                     <1>       stc
00004C79 C3                     <1>       retn
                                <1> spte_1:
00004C7A 89D0                   <1>       mov   eax, edx
                                <1> spte_2:
```

```
638 00004C7C 09CA               <1>       or    edx, ecx
639                             <1>       ; 23/06/2015
640 00004C7E 8913               <1>       mov   [ebx], edx ; PTE value in EDX
641                             <1>       ; 24/07/2015
642 00004C80 5B                 <1>       pop   ebx
643 00004C81 C3                 <1>       retn
644                             <1>
645                             <1> get_pde:    ; Get present value of the relevant PDE
646                             <1>       ; 20/07/2015
647                             <1>       ; 18/04/2015
648                             <1>       ; 12/04/2015
649                             <1>       ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
650                             <1>       ;
651                             <1>       ; INPUT ->
652                             <1>       ;     EBX = virtual (linear) address
653                             <1>       ;     EAX = page directory (physical) address
654                             <1>       ; OUTPUT ->
655                             <1>       ;     EDX = Page directory entry address
656                             <1>       ;     EAX = Page directory entry value
657                             <1>       ;     CF = 1 -> PDE not present or invalid ?
658                             <1>       ; Modified Registers -> EDX, EAX
659                             <1>       ;
660 00004C82 89DA               <1>       mov   edx, ebx
661 00004C84 C1EA16             <1>       shr   edx, PAGE_D_SHIFT ; 22  (12+10)
662 00004C87 C1E202             <1>       shl   edx, 2 ; offset to page directory (1024*4)
663 00004C8A 01C2               <1>       add   edx, eax ; page directory address (physical)
664 00004C8C 8B02               <1>       mov   eax, [edx]
665 00004C8E A801               <1>       test  al, PDE_A_PRESENT ; page table is present or not !
666 00004C90 751F               <1>       jnz   short gpte_retn
667 00004C92 F9                 <1>       stc
668                             <1> gpde_retn:
669 00004C93 C3                 <1>       retn
670                             <1>
671                             <1> get_pte:
672                             <1>             ; Get present value of the relevant PTE
673                             <1>       ; 29/07/2015
674                             <1>       ; 20/07/2015
675                             <1>       ; 18/04/2015
676                             <1>       ; 12/04/2015
677                             <1>       ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
678                             <1>       ;
679                             <1>       ; INPUT ->
680                             <1>       ;     EBX = virtual (linear) address
681                             <1>       ;     EAX = page directory (physical) address
682                             <1>       ; OUTPUT ->
683                             <1>       ;     EDX = Page table entry address (if CF=0)
684                             <1>       ;           Page directory entry address (if CF=1)
685                             <1>       ;            (Bit 0 value is 0 if PT is not present)
686                             <1>       ;     EAX = Page table entry value (page address)
687                             <1>       ;     CF = 1 -> PDE not present or invalid ?
688                             <1>       ; Modified Registers -> EAX, EDX
689                             <1>       ;
690 00004C94 E8E9FFFFFF           <1>       call  get_pde
691 00004C99 72F8               <1>       jc    short gpde_retn     ; page table is not present
692                             <1>       ;jnc  short gpte_1
693                             <1>       ;retn
694                             <1> ;gpte_1:
695 00004C9B 662500F0           <1>       and   ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 bits
696 00004C9F 89DA               <1>       mov   edx, ebx
697 00004CA1 C1EA0C             <1>       shr   edx, PAGE_SHIFT ; 12
698 00004CA4 81E2FF030000       <1>       and   edx, PTE_MASK; 03FFh
699                             <1>                    ; clear higher 10 bits (PD bits)
700 00004CAA C1E202             <1>       shl   edx, 2 ; offset from start of page table (1024*4)
701 00004CAD 01C2               <1>       add   edx, eax
702 00004CAF 8B02               <1>       mov   eax, [edx]
703                             <1> gpte_retn:
704 00004CB1 C3                 <1>       retn
705                             <1>
706                             <1> deallocate_page_dir:
707                             <1>       ; 15/09/2015
708                             <1>       ; 05/08/2015
709                             <1>       ; 30/04/2015
710                             <1>       ; 28/04/2015
711                             <1>       ; 17/10/2014
712                             <1>       ; 12/10/2014 (Retro UNIX 386 v1 - beginning)
713                             <1>       ;
714                             <1>       ; INPUT ->
715                             <1>       ;     EAX = PHYSICAL ADDRESS OF THE PAGE DIRECTORY (CHILD)
716                             <1>       ;     EBX = PHYSICAL ADDRESS OF THE PARENT'S PAGE DIRECTORY
717                             <1>       ; OUTPUT ->
718                             <1>       ;     All of page tables in the page directory
719                             <1>       ;     and page dir's itself will be deallocated
720                             <1>       ;     except 'read only' duplicated pages (will be converted
721                             <1>       ;     to writable pages).
722                             <1>       ;
723                             <1>       ; Modified Registers -> EAX
724                             <1>       ;
725                             <1>       ;
726 00004CB2 56                 <1>       push  esi
727 00004CB3 51                 <1>       push  ecx
728 00004CB4 50                 <1>       push  eax
729 00004CB5 89C6               <1>       mov   esi, eax
730 00004CB7 31C9               <1>       xor   ecx, ecx
731                             <1>       ; The 1st PDE points to Kernel Page Table 0 (the 1st 4MB),
732                             <1>       ; it must not be deallocated
733 00004CB9 890E               <1>       mov   [esi], ecx ; 0 ; clear PDE 0
734                             <1> dapd_0:
735 00004CBB AD                 <1>       lodsd
736 00004CBC A801               <1>       test  al, PDE_A_PRESENT ; bit 0, present flag (must be 1)
737 00004CBE 7409               <1>       jz    short dapd_1
738 00004CC0 662500F0           <1>       and   ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
739 00004CC4 E812000000         <1>       call  deallocate_page_table
```

```
                        <1> dapd_1:
00004CC9 41             <1>         inc     ecx ; page directory entry index
00004CCA 81F900040000   <1>         cmp     ecx, PAGE_SIZE / 4 ; 1024
00004CD0 72E9           <1>         jb      short dapd_0
                        <1> dapd_2:
00004CD2 58             <1>         pop     eax
00004CD3 E87F000000     <1>         call    deallocate_page     ; deallocate the page dir's itself
00004CD8 59             <1>         pop     ecx
00004CD9 5E             <1>         pop     esi
00004CDA 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>         ;
00004CDB 56             <1>         push    esi
00004CDC 57             <1>         push    edi
00004CDD 52             <1>         push    edx
00004CDE 50             <1>         push    eax ; *
00004CDF 89C6           <1>         mov     esi, eax
00004CE1 31FF           <1>         xor     edi, edi ; 0
                        <1> dapt_0:
00004CE3 AD             <1>         lodsd
00004CE4 A801           <1>         test    al, PTE_A_PRESENT ; bit 0, present flag (must be 1)
00004CE6 7441           <1>         jz      short dapt_1
                        <1>         ;
00004CE8 A802           <1>         test    al, PTE_A_WRITE   ; bit 1, writable (r/w) flag
                        <1>                                   ; (must be 1)
00004CEA 754C           <1>         jnz     short dapt_3
                        <1>         ; Read only -duplicated- page (belongs to a parent or a child)
00004CEC 66A90002       <1>           test    ax, PTE_DUPLICATED ; Was this page duplicated
                        <1>                                      ; as child's page ?
00004CF0 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)
00004CF2 53             <1>         push    ebx
00004CF3 51             <1>         push    ecx
00004CF4 66C1E102       <1>         shl     cx, 2 ; *4
00004CF8 01CB           <1>         add     ebx, ecx ; PDE offset (for the parent)
00004CFA 8B0B           <1>         mov     ecx, [ebx]
00004CFC F6C101         <1>         test    cl, PDE_A_PRESENT ; present (valid) or not ?
00004CFF 7435           <1>         jz      short dapt_2 ; parent process does not use this page
00004D01 6681E100F0     <1>         and     cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
                        <1>         ; EDI = page table entry index (0-1023)
00004D06 89FA           <1>         mov     edx, edi
00004D08 66C1E202       <1>         shl     dx, 2 ; *4
00004D0C 01CA           <1>         add     edx, ecx ; PTE offset (for the parent)
00004D0E 8B1A           <1>         mov     ebx, [edx]
00004D10 F6C301         <1>         test    bl, PTE_A_PRESENT ; present or not ?
00004D13 7421           <1>         jz      short dapt_2 ; parent process does not use this page
00004D15 662500F0       <1>         and     ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
00004D19 6681E300F0     <1>         and     bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
00004D1E 39D8           <1>         cmp     eax, ebx     ; parent's and child's pages are same ?
00004D20 7514           <1>         jne     short dapt_2 ; not same page
                        <1>                              ; deallocate the child's page
00004D22 800A02         <1>           or      byte [edx], PTE_A_WRITE ; convert to writable page (parent)
00004D25 59             <1>         pop     ecx
00004D26 5B             <1>         pop     ebx
00004D27 EB1A           <1>         jmp     short dapt_4
                        <1> dapt_1:
00004D29 09C0           <1>         or      eax, eax     ; swapped page ?
00004D2B 741D           <1>         jz      short dapt_5 ; no
                        <1>                              ; yes
00004D2D D1E8           <1>         shr     eax, 1
00004D2F E8CA040000     <1>         call    unlink_swap_block ; Deallocate swapped page block
                        <1>                                   ; on the swap disk (or in file)
00004D34 EB14           <1>         jmp     short dapt_5
                        <1> dapt_2:
00004D36 59             <1>         pop     ecx
00004D37 5B             <1>         pop     ebx
                        <1> dapt_3:
                        <1>         ; 12/07/2016
00004D38 66A90004       <1>         test    ax, PTE_SHARED ; shared or direct memory access indicator
00004D3C 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
00004D3E E814000000     <1>         call    deallocate_page ; set the mem allocation bit of this page
                        <1> dapt_4:
00004D43 C746FC00000000 <1>         mov     dword [esi-4], 0 ; clear/reset PTE (child, dupl. as parent)
                        <1> dapt_5:
00004D4A 47             <1>         inc     edi ; page table entry index
00004D4B 81FF00040000   <1>         cmp     edi, PAGE_SIZE / 4 ; 1024
00004D51 7290           <1>         jb      short dapt_0
                        <1>         ;
```

```
842 00004D53 58                 <1>       pop    eax ; *
843 00004D54 5A                 <1>       pop    edx
844 00004D55 5F                 <1>       pop    edi
845 00004D56 5E                 <1>       pop    esi
846                             <1>       ;
847                             <1>       ;call  deallocate_page     ; deallocate the page table's itself
848                             <1>       ;retn
849                             <1>
850                             <1> deallocate_page:
851                             <1>       ; 15/09/2015
852                             <1>       ; 28/04/2015
853                             <1>       ; 10/03/2015
854                             <1>       ; 17/10/2014
855                             <1>       ; 12/10/2014 (Retro UNIX 386 v1 - beginning)
856                             <1>       ;
857                             <1>       ; INPUT ->
858                             <1>       ;     EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
859                             <1>       ; OUTPUT ->
860                             <1>       ;     [free_pages] is increased
861                             <1>       ;     (corresponding MEMORY ALLOCATION TABLE bit is SET)
862                             <1>       ;     CF = 1 if the page is already deallocated
863                             <1>       ;            (or not allocated) before.
864                             <1>       ;
865                             <1>       ; Modified Registers -> EAX
866                             <1>       ;
867 00004D57 53                 <1>       push   ebx
868 00004D58 52                 <1>       push   edx
869                             <1>       ;
870 00004D59 C1E80C             <1>       shr    eax, PAGE_SHIFT      ; shift physical address to
871                             <1>                                   ; 12 bits right
872                             <1>                                   ; to get page number
873 00004D5C 89C2               <1>       mov    edx, eax
874                             <1>       ; 15/09/2015
875 00004D5E C1EA03             <1>       shr    edx, 3            ; to get offset to M.A.T.
876                             <1>                                   ; (1 allocation bit = 1 page)
877                             <1>                                   ; (1 allocation bytes = 8 pages)
878 00004D61 80E2FC             <1>       and    dl, 0FCh          ; clear lower 2 bits
879                             <1>                                   ; (to get 32 bit position)
880                             <1>       ;
881 00004D64 BB00001000         <1>       mov    ebx, MEM_ALLOC_TBL   ; Memory Allocation Table address
882 00004D69 01D3               <1>       add    ebx, edx
883 00004D6B 83E01F             <1>       and    eax, 1Fh          ; lower 5 bits only
884                             <1>                                   ; (allocation bit position)
885 00004D6E 3B15[0C520100]     <1>       cmp    edx, [next_page]     ; is the new free page address lower
886                             <1>                                   ; than the address in 'next_page' ?
887                             <1>                                   ; (next/first free page value)
888 00004D74 7306               <1>       jnb    short dap_1       ; no
889 00004D76 8915[0C520100]     <1>       mov    [next_page], edx     ; yes
890                             <1> dap_1:
891 00004D7C 0FAB03             <1>       bts    [ebx], eax        ; unlink/release/deallocate page
892                             <1>                                   ; set relevant bit to 1.
893                             <1>                                   ; set CF to the previous bit value
894                             <1>       ;cmc                        ; complement carry flag
895                             <1>       ;jc    short dap_2        ; do not increase free_pages count
896                             <1>                                   ; if the page is already deallocated
897                             <1>                                   ; before.
898 00004D7F FF05[08520100]     <1>         inc     dword [free_pages]
899                             <1> dap_2:
900 00004D85 5A                 <1>       pop    edx
901 00004D86 5B                 <1>       pop    ebx
902 00004D87 C3                 <1>       retn
903                             <1>
904                             <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
905                             <1> ;;                                                                  ;;
906                             <1> ;; Copyright (C) KolibriOS team 2004-2012. All rights reserved.     ;;
907                             <1> ;; Distributed under terms of the GNU General Public License        ;;
908                             <1> ;;                                                                  ;;
909                             <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
910                             <1>
911                             <1> ;;$Revision: 5057 $
912                             <1>
913                             <1>
914                             <1> ;;align 4
915                             <1> ;;proc alloc_page
916                             <1>
917                             <1> ;;        pushfd
918                             <1> ;;        cli
919                             <1> ;;        push    ebx
920                             <1> ;;;//-
921                             <1> ;;        cmp     [pg_data.pages_free], 1
922                             <1> ;;        jle     .out_of_memory
923                             <1> ;;;//-
924                             <1> ;;
925                             <1> ;;        mov     ebx, [page_start]
926                             <1> ;;        mov     ecx, [page_end]
927                             <1> ;;.l1:
928                             <1> ;;        bsf     eax, [ebx];
929                             <1> ;;        jnz     .found
930                             <1> ;;        add     ebx, 4
931                             <1> ;;        cmp     ebx, ecx
932                             <1> ;;        jb      .l1
933                             <1> ;;        pop     ebx
934                             <1> ;;        popfd
935                             <1> ;;        xor     eax, eax
936                             <1> ;;        ret
937                             <1> ;;.found:
938                             <1> ;;;//-
939                             <1> ;;        dec     [pg_data.pages_free]
940                             <1> ;;        jz      .out_of_memory
941                             <1> ;;;//-
942                             <1> ;;        btr     [ebx], eax
943                             <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>       ;
00004D88 E8ECFDFFFF     <1>       call  allocate_page
00004D8D 723E           <1>       jc    short dpd_err
                        <1>       ;
00004D8F 55             <1>       push  ebp ; 20/07/2015
00004D90 56             <1>       push  esi
00004D91 57             <1>       push  edi
00004D92 53             <1>       push  ebx
00004D93 51             <1>       push  ecx
00004D94 8B35[B8030300] <1>       mov   esi, [u.pgdir]
00004D9A 89C7           <1>       mov   edi, eax
00004D9C 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)
00004D9D A5             <1>       movsd
00004D9E BD00004000     <1>       mov   ebp, 1024*4096 ; pass the 1st 4MB (system space)
00004DA3 B9FF030000     <1>       mov   ecx, (PAGE_SIZE / 4) - 1 ; 1023
                        <1> dpd_0:
00004DA8 AD             <1>       lodsd
                        <1>       ;or   eax, eax
                        <1>         ;jnz     short dpd_1
00004DA9 A801           <1>       test  al, PDE_A_PRESENT ;  bit 0 =  1
00004DAB 7508           <1>       jnz   short dpd_1
                        <1>       ; 20/07/2015 (virtual address at the end of the page table)
00004DAD 81C500004000   <1>       add   ebp, 1024*4096 ; page size * PTE count
00004DB3 EB0F           <1>       jmp   short dpd_2
                        <1> dpd_1:
00004DB5 662500F0       <1>       and   ax, PDE_A_CLEAR ; 0F000h ; clear attribute bits
00004DB9 89C3           <1>       mov   ebx, eax
                        <1>       ; EBX = Parent's page table address
00004DBB E81F000000     <1>       call  duplicate_page_table
00004DC0 720C           <1>       jc    short dpd_p_err
                        <1>       ; EAX = Child's page table address
00004DC2 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:
00004DC4 AB             <1>       stosd
00004DC5 E2E1           <1>       loop  dpd_0
                        <1>       ;
00004DC7 58             <1>       pop   eax  ; restore child's page directory address
                        <1> dpd_3:
00004DC8 59             <1>       pop   ecx
00004DC9 5B             <1>       pop   ebx
00004DCA 5F             <1>       pop   edi
00004DCB 5E             <1>       pop   esi
00004DCC 5D             <1>       pop   ebp ; 20/07/2015
                        <1> dpd_err:
00004DCD C3             <1>       retn
                        <1> dpd_p_err:
                        <1>       ; release the allocated pages missing (recover free space)
00004DCE 58             <1>       pop   eax  ; the new page directory address (physical)
00004DCF 8B1D[B8030300] <1>       mov   ebx, [u.pgdir] ; parent's page directory address
00004DD5 E8D8FEFFFF     <1>       call  deallocate_page_dir
00004DDA 29C0           <1>       sub   eax, eax ; 0
00004DDC F9             <1>       stc
00004DDD EBE9           <1>       jmp   short dpd_3
                        <1>
                        <1> duplicate_page_table:
                        <1>       ; 20/02/2017
                        <1>       ; 21/09/2015
```

```
1046                                    <1>       ; 20/07/2015
1047                                    <1>       ; 05/05/2015
1048                                    <1>       ; 28/04/2015
1049                                    <1>       ; 27/04/2015
1050                                    <1>       ; 18/04/2015
1051                                    <1>       ; 18/10/2014
1052                                    <1>       ; 16/10/2014 (Retro UNIX 386 v1 - beginning)
1053                                    <1>       ;
1054                                    <1>       ; INPUT ->
1055                                    <1>       ;     EBX = PHYSICAL (real/flat) ADDRESS of the parent's page table.
1056                                    <1>       ;       20/02/2017
1057                                    <1>       ;     EBP = Linear address of the page (from 'duplicate_page_dir')
1058                                    <1>       ;           (Linear address = CORE + user's virtual address)
1059                                    <1>       ; OUTPUT ->
1060                                    <1>       ;     EAX = PHYSICAL (real/flat) ADDRESS of the child's page table.
1061                                    <1>       ;           (with 'read only' attribute of page table entries)
1062                                    <1>       ;     20/02/2017
1063                                    <1>       ;     EBP = Next linear page address (for 'duplicate_page_dir')
1064                                    <1>       ;
1065                                    <1>       ;     CF = 1 -> error
1066                                    <1>       ;
1067                                    <1>       ; Modified Registers -> EBP (except EAX)
1068                                    <1>       ;
1069 00004DDF E895FDFFFF               <1>       call  allocate_page
1070 00004DE4 726A                     <1>       jc    short dpt_err
1071                                    <1>       ;
1072 00004DE6 50                       <1>       push  eax ; *
1073 00004DE7 56                       <1>       push  esi
1074 00004DE8 57                       <1>       push  edi
1075 00004DE9 52                       <1>       push  edx
1076 00004DEA 51                       <1>       push  ecx
1077                                    <1>       ;
1078 00004DEB 89DE                     <1>       mov   esi, ebx
1079 00004DED 89C7                     <1>       mov   edi, eax
1080 00004DEF 89C2                     <1>       mov   edx, eax
1081 00004DF1 81C200100000             <1>       add   edx, PAGE_SIZE
1082                                    <1> dpt_0:
1083 00004DF7 AD                       <1>       lodsd
1084 00004DF8 21C0                     <1>       and   eax, eax
1085 00004DFA 7444                     <1>       jz    short dpt_3
1086 00004DFC A801                     <1>       test  al, PTE_A_PRESENT ;  bit 0 =  1
1087 00004DFE 7507                     <1>       jnz   short dpt_1
1088                                    <1>       ; 20/07/2015
1089                                    <1>       ; ebp = virtual (linear) address of the memory page
1090 00004E00 E83F050000               <1>       call  reload_page ; 28/04/2015
1091 00004E05 7244                     <1>       jc    short dpt_p_err
1092                                    <1> dpt_1:
1093                                    <1>       ; 21/09/2015
1094 00004E07 89C1                     <1>       mov   ecx, eax
1095 00004E09 662500F0                 <1>       and   ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
1096 00004E0D F6C102                   <1>       test  cl, PTE_A_WRITE ; writable page ?
1097 00004E10 7525                     <1>       jnz   short dpt_2
1098                                    <1>       ; Read only (parent) page
1099                                    <1>       ;     - there is a third process which uses this page -
1100                                    <1>       ; Allocate a new page for the child process
1101 00004E12 E862FDFFFF               <1>       call  allocate_page
1102 00004E17 7232                     <1>       jc    short dpt_p_err
1103 00004E19 57                       <1>       push  edi
1104 00004E1A 56                       <1>       push  esi
1105 00004E1B 89CE                     <1>       mov   esi, ecx
1106 00004E1D 89C7                     <1>       mov   edi, eax
1107 00004E1F B900040000               <1>       mov   ecx, PAGE_SIZE/4
1108 00004E24 F3A5                     <1>       rep   movsd  ; copy page (4096 bytes)
1109 00004E26 5E                       <1>       pop   esi
1110 00004E27 5F                       <1>       pop   edi
1111                                    <1>       ;
1112 00004E28 53                       <1>       push  ebx
1113 00004E29 50                       <1>       push  eax
1114                                    <1>       ; 20/07/2015
1115 00004E2A 89EB                     <1>       mov   ebx, ebp
1116                                    <1>       ; ebx = virtual (linear) address of the memory page
1117 00004E2C E887030000               <1>       call  add_to_swap_queue
1118 00004E31 58                       <1>       pop   eax
1119 00004E32 5B                       <1>       pop   ebx
1120                                    <1>       ; 21/09/2015
1121 00004E33 0C07                     <1>       or    al, PTE_A_USER+PTE_A_WRITE+PTE_A_PRESENT
1122                                    <1>             ; user + writable + present page
1123 00004E35 EB09                     <1>       jmp   short dpt_3
1124                                    <1> dpt_2:
1125                                    <1>       ;or   ax, PTE_A_USER+PTE_A_PRESENT
1126 00004E37 0C05                     <1>       or    al, PTE_A_USER+PTE_A_PRESENT
1127                                    <1>             ; (read only page!)
1128 00004E39 8946FC                   <1>       mov   [esi-4], eax ; update parent's PTE
1129 00004E3C 660D0002                 <1>       or     ax, PTE_DUPLICATED  ; (read only page & duplicated PTE!)
1130                                    <1> dpt_3:
1131 00004E40 AB                       <1>       stosd  ; EDI points to child's PTE
1132                                    <1>       ;
1133 00004E41 81C500100000             <1>       add   ebp, 4096 ; 20/07/2015 (next page)
1134                                    <1>       ;
1135 00004E47 39D7                     <1>       cmp   edi, edx
1136 00004E49 72AC                     <1>       jb    short dpt_0
1137                                    <1> dpt_p_err:
1138 00004E4B 59                       <1>       pop   ecx
1139 00004E4C 5A                       <1>       pop   edx
1140 00004E4D 5F                       <1>       pop   edi
1141 00004E4E 5E                       <1>       pop   esi
1142 00004E4F 58                       <1>       pop   eax ; *
1143                                    <1> dpt_err:
1144 00004E50 C3                       <1>       retn
1145                                    <1>
1146                                    <1> page_fault_handler:      ; CPU EXCEPTION 0Eh (14) : Page Fault !
1147                                    <1>       ; 21/09/2015
```

```
1148                                    <1>       ; 19/09/2015
1149                                    <1>       ; 17/09/2015
1150                                    <1>       ; 28/08/2015
1151                                    <1>       ; 20/07/2015
1152                                    <1>       ; 28/06/2015
1153                                    <1>       ; 03/05/2015
1154                                    <1>       ; 30/04/2015
1155                                    <1>       ; 18/04/2015
1156                                    <1>       ; 12/04/2015
1157                                    <1>       ; 30/10/2014
1158                                    <1>       ; 11/09/2014
1159                                    <1>       ; 10/09/2014 (Retro UNIX 386 v1 - beginning)
1160                                    <1>       ;
1161                                    <1>       ; Note: This is not an interrupt/exception handler.
1162                                    <1>       ;     This is a 'page fault remedy' subroutine
1163                                    <1>       ;     which will be called by standard/uniform
1164                                    <1>       ;     exception handler.
1165                                    <1>       ;
1166                                    <1>       ; INPUT ->
1167                                    <1>       ;     [error_code] = 32 bit ERROR CODE (lower 5 bits are valid)
1168                                    <1>       ;
1169                                    <1>       ;     cr2 = the virtual (linear) address
1170                                    <1>       ;           which has caused to page fault (19/09/2015)
1171                                    <1>       ;
1172                                    <1>       ; OUTPUT ->
1173                                    <1>       ;     (corresponding PAGE TABLE ENTRY is mapped/set)
1174                                    <1>       ;     EAX = 0 -> no error
1175                                    <1>       ;     EAX > 0 -> error code in EAX (also CF = 1)
1176                                    <1>       ;
1177                                    <1>       ; Modified Registers -> none (except EAX)
1178                                    <1>       ;
1179                                    <1>         ;
1180                                    <1>         ; ERROR CODE:
1181                                    <1>       ;      31  .....   4   3 2   1 0
1182                                    <1>       ;     +---+-- --+---+---+---+---+---+---+
1183                                    <1>       ;     |   Reserved  | I | R | U | W | P |
1184                                    <1>       ;     +---+-- --+---+---+---+---+---+---+
1185                                    <1>       ;
1186                                    <1>       ; P : PRESENT -     When set, the page fault was caused by
1187                                    <1>       ;             a page-protection violation. When not set,
1188                                    <1>       ;             it was caused by a non-present page.
1189                                    <1>       ; W : WRITE   -     When set, the page fault was caused by
1190                                    <1>       ;             a page write. When not set, it was caused
1191                                    <1>       ;             by a page read.
1192                                    <1>       ; U : USER    -     When set, the page fault was caused
1193                                    <1>       ;             while CPL = 3.
1194                                    <1>       ;             This does not necessarily mean that
1195                                    <1>       ;             the page fault was a privilege violation.
1196                                    <1>       ; R : RESERVD -     When set, the page fault was caused by
1197                                    <1>       ;     WRITE  reading a 1 in a reserved field.
1198                                    <1>       ; I : INSTRUC -     When set, the page fault was caused by
1199                                    <1>       ;     FETCH  an instruction fetch
1200                                    <1>       ;
1201                                    <1>       ;; x86 (32 bit) VIRTUAL ADDRESS TRANSLATION
1202                                    <1>       ;  31               22                  12 11                    0
1203                                    <1>       ; +-------------------+-------------------+-----------------------+
1204                                    <1>             ; | PAGE DIR. ENTRY # | PAGE TAB. ENTRY # |        OFFSET         |
1205                                    <1>             ; +-------------------+-------------------+-----------------------+
1206                                    <1>       ;
1207                                    <1>
1208                                    <1>       ;; CR3 REGISTER (Control Register 3)
1209                                    <1>       ;  31                                     12             5 4 3 2   0
1210                                    <1>       ; +---------------------------------------+-------------+---+-----+
1211                                    <1>             ; |                                       |         |P|P|     |
1212                                    <1>             ; |   PAGE DIRECTORY TABLE BASE ADDRESS   |  reserved     |C|W|rsvrd|
1213                                    <1>             ; |                                       |         |D|T|     |
1214                                    <1>       ; +---------------------------------------+-------------+---+-----+
1215                                    <1>       ;
1216                                    <1>       ;     PWT    - WRITE THROUGH
1217                                    <1>       ;     PCD    - CACHE DISABLE
1218                                    <1>       ;
1219                                    <1>       ;
1220                                    <1>       ;; x86 PAGE DIRECTORY ENTRY (4 KByte Page)
1221                                    <1>       ;  31                                   12 11  9 8 7 6 5 4 3 2 1 0
1222                                    <1>       ; +---------------------------------------+-----+---+-+-+---+-+-+-+
1223                                    <1>             ; |                                       |     | | | | |P|P|U|R| |
1224                                    <1>             ; |     PAGE TABLE BASE ADDRESS 31..12    | AVL |G|0|D|A|C|W|/|/|P|
1225                                    <1>             ; |                                       |     | | | | |D|T|S|W| |
1226                                    <1>       ; +---------------------------------------+-----+---+-+-+---+-+-+-+
1227                                    <1>       ;
1228                                    <1>         ;       P      - PRESENT
1229                                    <1>         ;       R/W    - READ/WRITE
1230                                    <1>         ;       U/S    - USER/SUPERVISOR
1231                                    <1>       ;     PWT    - WRITE THROUGH
1232                                    <1>       ;     PCD    - CACHE DISABLE
1233                                    <1>       ;     A      - ACCESSED
1234                                    <1>         ;       D      - DIRTY (IGNORED)
1235                                    <1>       ;     PAT    - PAGE ATTRIBUTE TABLE INDEX (CACHE BEHAVIOR)
1236                                    <1>       ;     G      - GLOBAL     (IGNORED)
1237                                    <1>         ;       AVL    - AVAILABLE FOR SYSTEMS PROGRAMMER USE
1238                                    <1>       ;
1239                                    <1>       ;
1240                                    <1>       ;; x86 PAGE TABLE ENTRY (4 KByte Page)
1241                                    <1>       ;  31                                   12 11  9 8 7 6 5 4 3 2 1 0
1242                                    <1>       ; +---------------------------------------+-----+---+-+-+---+-+-+-+
1243                                    <1>             ; |                                       |     | |P| | |P|P|U|R| |
1244                                    <1>             ; |     PAGE FRAME BASE ADDRESS 31..12    | AVL |G|A|D|A|C|W|/|/|P|
1245                                    <1>             ; |                                       |     | |T| | |D|T|S|W| |
1246                                    <1>       ; +---------------------------------------+-----+---+-+-+---+-+-+-+
1247                                    <1>       ;
1248                                    <1>         ;       P      - PRESENT
1249                                    <1>         ;       R/W    - READ/WRITE
```

```
                        <1>         ;       U/S     - USER/SUPERVISOR
                        <1>       ;     PWT     - WRITE THROUGH
                        <1>       ;     PCD     - CACHE DISABLE
                        <1>       ;     A       - ACCESSED
                        <1>         ;       D       - DIRTY
                        <1>       ;     PAT     - PAGE ATTRIBUTE TABLE INDEX (CACHE BEHAVIOR)
                        <1>       ;     G       - GLOBAL
                        <1>         ;       AVL     - AVAILABLE FOR SYSTEMS PROGRAMMER USE
                        <1>       ;
                        <1>       ;
                        <1>       ;; 80386 PAGE TABLE ENTRY (4 KByte Page)
                        <1>       ;  31                                   12 11  9 8 7 6 5 4 3 2 1 0
                        <1>       ; +---------------------------------------+-----+-+-+-+-+---+-+-+-+
                        <1>             ; |                                       |     | | | | | | |U|R| |
                        <1>             ; |     PAGE FRAME BASE ADDRESS 31..12    | AVL |0|0|D|A|0|0|/|/|P|
                        <1>             ; |                                       |     | | | | | | |S|W| |
                        <1>             ; +---------------------------------------+-----+-+-+-+-+---+-+-+-+
                        <1>       ;
                        <1>         ;       P       - PRESENT
                        <1>         ;       R/W     - READ/WRITE
                        <1>         ;       U/S     - USER/SUPERVISOR
                        <1>         ;       D       - DIRTY
                        <1>         ;       AVL     - AVAILABLE FOR SYSTEMS PROGRAMMER USE
                        <1>       ;
                        <1>         ;       NOTE: 0 INDICATES INTEL RESERVED. DO NOT DEFINE.
                        <1>       ;
                        <1>       ;
                        <1>       ;; Invalid Page Table Entry
                        <1>       ; 31                                                    1 0
                        <1>             ; +-------------------------------------------------------------+-+
                        <1>             ; |                                                             | |
                        <1>             ; |                           AVAILABLE                         |0|
                        <1>             ; |                                                             | |
                        <1>             ; +-------------------------------------------------------------+-+
                        <1>       ;
                        <1>
00004E51 53             <1>       push  ebx
00004E52 52             <1>       push  edx
00004E53 51             <1>       push  ecx
                        <1>       ;
                        <1>       ; 21/09/2015 (debugging)
00004E54 FF05[CC030300] <1>       inc   dword [u.pfcount] ; page fault count for running process
00004E5A FF05[80050300] <1>       inc   dword [PF_Count] ; total page fault count
                        <1>       ; 28/06/2015
                        <1>       ;mov  edx, [error_code] ; Lower 5 bits are valid
00004E60 8A15[78050300] <1>       mov   dl, [error_code]
                        <1>       ;
00004E66 F6C201         <1>       test  dl, 1  ; page fault was caused by a non-present page
                        <1>                    ; sign
00004E69 7422           <1>       jz    short pfh_alloc_np
                        <1>       ;
                        <1>       ; If it is not a 'write on read only page' type page fault
                        <1>       ; major page fault error with minor reason must be returned without
                        <1>       ; fixing the problem. 'sys_exit with error' will be needed
                        <1>       ; after return here!
                        <1>       ; Page fault will be remedied, by copying page contents
                        <1>       ; to newly allocated page with write permission;
                        <1>       ; sys_fork -> sys_exec -> copy on write, demand paging method is
                        <1>       ; used for working with minimum possible memory usage.
                        <1>       ; sys_fork will duplicate page directory and tables of parent
                        <1>       ; process with 'read only' flag. If the child process attempts to
                        <1>       ; write on these read only pages, page fault will be directed here
                        <1>       ; for allocating a new page with same data/content.
                        <1>       ;
                        <1>       ; IMPORTANT : Retro UNIX 386 v1 (and SINGLIX and TR-DOS)
                        <1>       ; will not force to separate CODE and DATA space
                        <1>       ; in a process/program...
                        <1>       ; CODE segment/section may contain DATA!
                        <1>       ; It is flat, smoth and simplest programming method already as in
                        <1>       ; Retro UNIX 8086 v1 and MS-DOS programs.
                        <1>       ;
00004E6B F6C202         <1>       test  dl, 2  ; page fault was caused by a page write
                        <1>                    ; sign
00004E6E 0F84AB000000   <1>         jz      pfh_p_err
                        <1>       ; 31/08/2015
00004E74 F6C204         <1>       test  dl, 4  ; page fault was caused while CPL = 3 (user mode)
                        <1>                    ; sign.  (U+W+P = 4+2+1 = 7)
00004E77 0F84A2000000   <1>         jz  pfh_pv_err
                        <1>       ;
                        <1>       ; make a new page and copy the parent's page content
                        <1>       ; as the child's new page content
                        <1>       ;
00004E7D 0F20D3         <1>       mov   ebx, cr2 ; CR2 contains the linear address
                        <1>                      ; which has caused to page fault
00004E80 E8A2000000     <1>       call  copy_page
00004E85 0F828D000000   <1>         jc      pfh_im_err ; insufficient memory
                        <1>       ;
00004E8B EB7D           <1>         jmp     pfh_cpp_ok
                        <1>       ;
                        <1> pfh_alloc_np:
00004E8D E8E7FCFFFF     <1>       call  allocate_page; (allocate a new page)
00004E92 0F8280000000   <1>         jc      pfh_im_err      ; 'insufficient memory' error
                        <1> pfh_chk_cpl:
                        <1>       ; EAX = Physical (base) address of the allocated (new) page
                        <1>             ; (Lower 12 bits are ZERO, because
                        <1>             ;     the address is on a page boundary)
00004E98 80E204         <1>       and   dl, 4  ; CPL = 3 ?
00004E9B 7505           <1>       jnz   short pfh_um
                        <1>                    ; Page fault handler for kernel/system mode (CPL=0)
00004E9D 0F20DB         <1>       mov   ebx, cr3 ; CR3 (Control Register 3) contains physical address
                        <1>                      ; of the current/active page directory
                        <1>                      ; (Always kernel/system mode page directory, here!)
```

```
                                <1>                     ; Note: Lower 12 bits are 0. (page boundary)
00004EA0 EB06                   <1>       jmp    short pfh_get_pde
                                <1>       ;
                                <1> pfh_um:                     ; Page fault handler for user/appl. mode (CPL=3)
00004EA2 8B1D[B8030300]         <1>       mov    ebx, [u.pgdir] ; Page directory of current/active process
                                <1>                       ; Physical address of the USER's page directory
                                <1>                       ; Note: Lower 12 bits are 0. (page boundary)
                                <1> pfh_get_pde:
00004EA8 80CA03                 <1>       or     dl, 3  ; USER + WRITE + PRESENT or SYSTEM + WRITE + PRESENT
00004EAB 0F20D1                 <1>       mov    ecx, cr2 ; CR2 contains the virtual address
                                <1>                       ; which has been caused to page fault
                                <1>                       ;
00004EAE C1E914                 <1>       shr    ecx, 20       ; shift 20 bits right
00004EB1 80E1FC                 <1>       and    cl, 0FCh ; mask lower 2 bits to get PDE offset
                                <1>       ;
00004EB4 01CB                   <1>       add    ebx, ecx ; now, EBX points to the relevant page dir entry
00004EB6 8B0B                   <1>       mov    ecx, [ebx] ; physical (base) address of the page table
00004EB8 F6C101                 <1>       test   cl, 1   ; check bit 0 is set (1) or not (0).
00004EBB 740B                   <1>       jz     short pfh_set_pde ; Page directory entry is not valid,
                                <1>                              ; set/validate page directory entry
00004EBD 6681E100F0             <1>       and    cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
00004EC2 89CB                   <1>       mov    ebx, ecx ; Physical address of the page table
00004EC4 89C1                   <1>       mov    ecx, eax ; new page address (physical)
00004EC6 EB16                   <1>       jmp    short pfh_get_pte
                                <1> pfh_set_pde:
                                <1>       ;; NOTE: Page directories and page tables never be swapped out!
                                <1>       ;;     (So, we know this PDE is empty or invalid)
                                <1>       ;
00004EC8 08D0                   <1>       or     al, dl  ; lower 3 bits are used as U/S, R/W, P flags
00004ECA 8903                   <1>       mov    [ebx], eax ; Let's put the new page directory entry here !
00004ECC 30C0                   <1>       xor    al, al  ; clear lower (3..8) bits
00004ECE 89C3                   <1>       mov    ebx, eax
00004ED0 E8A4FCFFFF             <1>       call   allocate_page ; (allocate a new page)
00004ED5 7241                   <1>       jc     short pfh_im_err   ; 'insufficient memory' error
                                <1> pfh_spde_1:
                                <1>       ; EAX = Physical (base) address of the allocated (new) page
00004ED7 89C1                   <1>       mov    ecx, eax
00004ED9 E815FDFFFF             <1>       call   clear_page ; Clear page content
                                <1> pfh_get_pte:
00004EDE 0F20D0                 <1>       mov    eax, cr2 ; virtual address
                                <1>                       ; which has been caused to page fault
00004EE1 89C7                   <1>       mov    edi, eax ; 20/07/2015
00004EE3 C1E80C                 <1>       shr    eax, 12       ; shift 12 bit right to get
                                <1>                       ; higher 20 bits of the page fault address
00004EE6 25FF030000             <1>       and    eax, 3FFh ; mask PDE# bits, the result is PTE# (0 to 1023)
00004EEB C1E002                 <1>       shl    eax, 2 ; shift 2 bits left to get PTE offset
00004EEE 01C3                   <1>       add    ebx, eax ; now, EBX points to the relevant page table entry
00004EF0 8B03                   <1>       mov    eax, [ebx] ; get previous value of pte
                                <1>                       ; bit 0 of EAX is always 0 (otherwise we would not be here)
00004EF2 21C0                   <1>       and    eax, eax
00004EF4 7410                   <1>       jz     short pfh_gpte_1
                                <1>       ; 20/07/2015
00004EF6 87D9                   <1>       xchg   ebx, ecx ; new page address (physical)
00004EF8 55                     <1>       push   ebp ; 20/07/2015
00004EF9 0F20D5                 <1>       mov    ebp, cr2
                                <1>                       ; ECX = physical address of the page table entry
                                <1>                       ; EBX = Memory page address (physical!)
                                <1>                       ; EAX = Swap disk (offset) address
                                <1>                       ; EBP = virtual address (page fault address)
00004EFC E8B7000000             <1>       call   swap_in
00004F01 5D                     <1>       pop    ebp
00004F02 7210                   <1>       jc       short pfh_err_retn
00004F04 87CB                   <1>       xchg   ecx, ebx
                                <1>                       ; EBX = physical address of the page table entry
                                <1>                       ; ECX = new page
                                <1> pfh_gpte_1:
00004F06 08D1                   <1>       or     cl, dl ; lower 3 bits are used as U/S, R/W, P flags
00004F08 890B                   <1>       mov    [ebx], ecx ; Let's put the new page table entry here !
                                <1> pfh_cpp_ok:
                                <1>       ; 20/07/2015
00004F0A 0F20D3                 <1>       mov    ebx, cr2
00004F0D E8A6020000             <1>       call   add_to_swap_queue
                                <1>       ;
                                <1>       ; The new PTE (which contains the new page) will be added to
                                <1>       ; the swap queue, here.
                                <1>       ; (Later, if memory will become insufficient,
                                <1>       ; one page will be swapped out which is at the head of
                                <1>       ; the swap queue by using FIFO and access check methods.)
                                <1>       ;
00004F12 31C0                   <1>       xor    eax, eax  ; 0
                                <1>       ;
                                <1> pfh_err_retn:
00004F14 59                     <1>       pop    ecx
00004F15 5A                     <1>       pop    edx
00004F16 5B                     <1>       pop    ebx
00004F17 C3                     <1>       retn
                                <1>
                                <1> pfh_im_err:
00004F18 B8E4000000             <1>       mov    eax, ERR_MAJOR_PF + ERR_MINOR_IM ; Error code in AX
                                <1>                       ; Major (Primary) Error: Page Fault
                                <1>                       ; Minor (Secondary) Error: Insufficient Memory !
00004F1D EBF5                   <1>       jmp    short pfh_err_retn
                                <1>
                                <1>
                                <1> pfh_p_err: ; 09/03/2015
                                <1> pfh_pv_err:
                                <1>       ; Page fault was caused by a protection-violation
00004F1F B8E6000000             <1>       mov    eax, ERR_MAJOR_PF + ERR_MINOR_PV ; Error code in AX
                                <1>                       ; Major (Primary) Error: Page Fault
                                <1>                       ; Minor (Secondary) Error: Protection violation !
00004F24 F9                     <1>       stc
00004F25 EBED                   <1>       jmp    short pfh_err_retn
```

```
                                <1>
                                <1> copy_page:
                                <1>       ; 22/09/2015
                                <1>       ; 21/09/2015
                                <1>       ; 19/09/2015
                                <1>       ; 07/09/2015
                                <1>       ; 31/08/2015
                                <1>       ; 20/07/2015
                                <1>       ; 05/05/2015
                                <1>       ; 03/05/2015
                                <1>       ; 18/04/2015
                                <1>       ; 12/04/2015
                                <1>       ; 30/10/2014
                                <1>       ; 18/10/2014 (Retro UNIX 386 v1 - beginning)
                                <1>       ;
                                <1>       ; INPUT ->
                                <1>       ;     EBX = Virtual (linear) address of source page
                                <1>       ;          (Page fault address)
                                <1>       ; OUTPUT ->
                                <1>       ;     EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
                                <1>       ;     (corresponding PAGE TABLE ENTRY is mapped/set)
                                <1>       ;     EAX = 0 (CF = 1)
                                <1>       ;           if there is not a free page to be allocated
                                <1>       ;     (page content of the source page will be copied
                                <1>       ;     onto the target/new page)
                                <1>       ;
                                <1>       ; Modified Registers -> ecx, ebx (except EAX)
                                <1>       ;
00004F27 56                     <1>       push   esi
00004F28 57                     <1>       push   edi
                                <1>       ;push  ebx
                                <1>       ;push  ecx
00004F29 31F6                   <1>       xor    esi, esi
00004F2B C1EB0C                 <1>       shr    ebx, 12 ; shift 12 bits right to get PDE & PTE numbers
00004F2E 89D9                   <1>       mov    ecx, ebx ; save page fault address (as 12 bit shifted)
00004F30 C1EB08                 <1>       shr    ebx, 8  ; shift 8 bits right and then
00004F33 80E3FC                 <1>       and    bl, 0FCh ; mask lower 2 bits to get PDE offset
00004F36 89DF                   <1>       mov    edi, ebx ; save it for the parent of current process
00004F38 031D[B8030300]         <1>       add    ebx, [u.pgdir] ; EBX points to the relevant page dir entry
00004F3E 8B03                   <1>       mov    eax, [ebx] ; physical (base) address of the page table
00004F40 662500F0               <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
00004F44 89CB                   <1>       mov    ebx, ecx   ; (restore higher 20 bits of page fault address)
00004F46 81E3FF030000           <1>       and    ebx, 3FFh  ; mask PDE# bits, the result is PTE# (0 to 1023)
00004F4C 66C1E302               <1>       shl    bx, 2     ; shift 2 bits left to get PTE offset
00004F50 01C3                   <1>       add    ebx, eax   ; EBX points to the relevant page table entry
                                <1>       ; 07/09/2015
00004F52 66F7030002             <1>         test    word [ebx], PTE_DUPLICATED ; (Does current process share this
                                <1>                                        ; read only page as a child process?)
00004F57 7509                   <1>       jnz    short cpp_0 ; yes
00004F59 8B0B                   <1>       mov    ecx, [ebx] ; PTE value
00004F5B 6681E100F0             <1>       and    cx, PTE_A_CLEAR ; 0F000h  ; clear page attributes
00004F60 EB32                   <1>       jmp    short cpp_1
                                <1> cpp_0:
00004F62 89FE                   <1>       mov    esi, edi
00004F64 0335[BC030300]         <1>       add    esi, [u.ppgdir] ; the parent's page directory entry
00004F6A 8B06                   <1>       mov    eax, [esi] ; physical (base) address of the page table
00004F6C 662500F0               <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
00004F70 89CE                   <1>       mov    esi, ecx   ; (restore higher 20 bits of page fault address)
00004F72 81E6FF030000           <1>       and    esi, 3FFh  ; mask PDE# bits, the result is PTE# (0 to 1023)
00004F78 66C1E602               <1>       shl    si, 2     ; shift 2 bits left to get PTE offset
00004F7C 01C6                   <1>       add    esi, eax   ; EDX points to the relevant page table entry
00004F7E 8B0E                   <1>       mov    ecx, [esi] ; PTE value of the parent process
                                <1>       ; 21/09/2015
00004F80 8B03                   <1>       mov    eax, [ebx] ; PTE value of the child process
00004F82 662500F0               <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear page attributes
                                <1>       ;
00004F86 F6C101                 <1>       test   cl, PTE_A_PRESENT ; is it a present/valid page ?
00004F89 7424                   <1>       jz     short cpp_3 ; the parent's page is not same page
                                <1>       ;
00004F8B 6681E100F0             <1>       and    cx, PTE_A_CLEAR ; 0F000h ; clear page attributes
00004F90 39C8                   <1>       cmp    eax, ecx   ; Same page?
00004F92 751B                   <1>       jne    short cpp_3 ; Parent page and child page are not same
                                <1>                          ; Convert child's page to writable page
                                <1> cpp_1:
00004F94 E8E0FBFFFF             <1>       call   allocate_page
00004F99 721A                   <1>       jc     short cpp_4 ; 'insufficient memory' error
00004F9B 21F6                   <1>       and    esi, esi    ; check ESI is valid or not
00004F9D 7405                   <1>       jz     short cpp_2
                                <1>              ; Convert read only page to writable page
                                <1>              ;(for the parent of the current process)
                                <1>       ;and   word [esi], PTE_A_CLEAR ; 0F000h
                                <1>       ; 22/09/2015
00004F9F 890E                   <1>       mov    [esi], ecx
00004FA1 800E07                 <1>       or     byte [esi], PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER
                                <1>                            ; 1+2+4 = 7
                                <1> cpp_2:
00004FA4 89C7                   <1>       mov    edi, eax ; new page address of the child process
                                <1>       ; 07/09/2015
00004FA6 89CE                   <1>       mov    esi, ecx ; the page address of the parent process
00004FA8 B900040000             <1>       mov    ecx, PAGE_SIZE / 4
00004FAD F3A5                   <1>       rep    movsd ; 31/08/2015
                                <1> cpp_3:
00004FAF 0C07                   <1>       or     al, PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER ; 1+2+4 = 7
00004FB1 8903                   <1>       mov    [ebx], eax ; Update PTE
00004FB3 28C0                   <1>       sub    al, al ; clear attributes
                                <1> cpp_4:
                                <1>       ;pop   ecx
                                <1>       ;pop   ebx
00004FB5 5F                     <1>       pop    edi
00004FB6 5E                     <1>       pop    esi
00004FB7 C3                     <1>       retn
                                <1>
```

```
<1> ;; 28/04/2015
<1> ;; 24/10/2014
<1> ;; 21/10/2014 (Retro UNIX 386 v1 - beginning)
<1> ;; SWAP_PAGE_QUEUE (4096 bytes)
<1> ;;
<1> ;;   0000   0001   0002   0003   ....   1020   1021   1022   1023
<1> ;; +------+------+------+------+-    -+------+------+------+------+
<1> ;; |  pg1 |  pg2 |  pg3 |  pg4 | .... |pg1021|pg1022|pg1023|pg1024|
<1> ;; +------+------+------+------+-    -+------+------+------+------+
<1> ;;
<1> ;; [swpq_last] = 0 to 4096 (step 4) -> the last position on the queue
<1> ;;
<1> ;; Method:
<1> ;;    Swap page queue is a list of allocated pages with physical
<1> ;;    addresses (system mode virtual adresses = physical addresses).
<1> ;;    It is used for 'swap_in' and 'swap_out' procedures.
<1> ;;    When a new page is being allocated, swap queue is updated
<1> ;;    by 'swap_queue_shift' procedure, header of the queue (offset 0)
<1> ;;    is checked for 'accessed' flag. If the 1st page on the queue
<1> ;;    is 'accessed' or 'read only', it is dropped from the list;
<1> ;;    other pages from the 2nd to the last (in [swpq_last]) shifted
<1> ;;    to head then the 2nd page becomes the 1st and '[swpq_last]'
<1> ;;    offset value becomes it's previous offset value - 4.
<1> ;;    If the 1st page of the swap page queue is not 'accessed'
<1> ;;    the queue/list is not shifted.
<1> ;;    After the queue/list shift, newly allocated page is added
<1> ;;    to the tail of the queue at the [swpq_count*4] position.
<1> ;;    But, if [swpq_count] > 1023, the newly allocated page
<1> ;;    will not be added to the tail of swap page queue.
<1> ;;
<1> ;;    During 'swap_out' procedure, swap page queue is checked for
<1> ;;    the first non-accessed, writable page in the list,
<1> ;;    from the head to the tail. The list is shifted to left
<1> ;;    (to the head) till a non-accessed page will be found in the list.
<1> ;;    Then, this page     is swapped out (to disk) and then it is dropped
<1> ;;    from the list by a final swap queue shift. [swpq_count] value
<1> ;;    is changed. If all pages on the queue' are 'accessed',
<1> ;;    'insufficient memory' error will be returned ('swap_out'
<1> ;;    procedure will be failed)...
<1> ;;
<1> ;;    Note: If the 1st page of the queue is an 'accessed' page,
<1> ;;    'accessed' flag of the page will be reset (0) and that page
<1> ;;    (PTE) will be added to the tail of the queue after
<1> ;;    the check, if [swpq_count] < 1023. If [swpq_count] = 1024
<1> ;;    the queue will be rotated and the PTE in the head will be
<1> ;;    added to the tail after resetting 'accessed' bit.
<1> ;;
<1> ;;
<1> ;;
<1> ;; SWAP DISK/FILE (with 4096 bytes swapped page blocks)
<1> ;;
<1> ;;  00000000  00000004  00000008  0000000C   ...   size-8    size-4
<1> ;; +---------+---------+---------+---------+-- --+---------+---------+
<1> ;; |descriptr| page(1) | page(2) | page(3) | ... |page(n-1)| page(n) |
<1> ;; +---------+---------+---------+---------+-- --+---------+---------+
<1> ;;
<1> ;; [swpd_next] = the first free block address in swapped page records
<1> ;;                  for next free block search by 'swap_out' procedure.
<1> ;; [swpd_size] = swap disk/file size in sectors (512 bytes)
<1> ;;           NOTE: max. possible swap disk size is 1024 GB
<1> ;;           (entire swap space must be accessed by using
<1> ;;           31 bit offset address)
<1> ;; [swpd_free] = free block (4096 bytes) count in swap disk/file space
<1> ;; [swpd_start] = absolute/start address of the swap disk/file
<1> ;;            0 for file, or beginning sector of the swap partition
<1> ;; [swp_drv] = logical drive description table addr. of swap disk/file
<1> ;;
<1> ;;
<1> ;; Method:
<1> ;;    When the memory (ram) becomes insufficient, page allocation
<1> ;;    procedure swaps out a page from memory to the swap disk
<1> ;;    (partition) or swap file to get a new free page at the memory.
<1> ;;    Swapping out is performed by using swap page queue.
<1> ;;
<1> ;;    Allocation block size of swap disk/file is equal to page size
<1> ;;    (4096 bytes). Swapping address (in sectors) is recorded
<1> ;;    into relevant page file entry as 31 bit physical (logical)
<1> ;;    offset address as 1 bit shifted to left for present flag (0).
<1> ;;    Swapped page address is between 1 and swap disk/file size - 4.
<1> ;;    Absolute physical (logical) address of the swapped page is
<1> ;;    calculated by adding offset value to the swap partition's
<1> ;;    start address. If the swap device (disk) is a virtual disk
<1> ;;    or it is a file, start address of the swap disk/volume is 0,
<1> ;;    and offset value is equal to absolute (physical or logical)
<1> ;;    address/position. (It has not to be ZERO if the swap partition
<1> ;;    is in a partitioned virtual hard disk.)
<1> ;;
<1> ;;    Note: Swap addresses are always specified/declared in sectors,
<1> ;;    not in bytes or     in blocks/zones/clusters (4096 bytes) as unit.
<1> ;;
<1> ;;    Swap disk/file allocation is mapped via 'Swap Allocation Table'
<1> ;;    at memory as similar to 'Memory Allocation Table'.
<1> ;;
<1> ;;    Every bit of Swap Allocation Table repsesents one swap block
<1> ;;    (equal to page size) respectively. Bit 0 of the S.A.T. byte 0
<1> ;;    is reserved for swap disk/file block 0 as descriptor block
<1> ;;    (also for compatibility with PTE). If bit value is ZERO,
<1> ;;    it means relevant (respective) block is in use, and,
<1> ;;    of course, if bit value is 1, it means relevant (respective)
<1> ;;      swap disk/file block is free.
<1> ;;    For example: bit 1 of the byte 128 repsesents block 1025
<1> ;;    (128*8+1) or sector (offset) 8200 on the swap disk or
```

```
                                    <1> ;;    byte (offset/position) 4198400 in the swap file.
                                    <1> ;;    4GB swap space is represented via 128KB Swap Allocation Table.
                                    <1> ;;    Initial layout of Swap Allocation Table is as follows:
                                    <1> ;;    ------------------------------------------------------------
                                    <1> ;;    01111111111111111111111111 .... 11111111111111111111111111111111
                                    <1> ;;    ------------------------------------------------------------
                                    <1> ;;    (0 is reserved block, 1s represent free blocks respectively.)
                                    <1> ;;    (Note: Allocation cell/unit of the table is bit, not byte)
                                    <1> ;;
                                    <1> ;;    ............................................................
                                    <1> ;;
                                    <1> ;;    'swap_out' procedure checks 'free_swap_blocks' count at first,
                                    <1> ;;    then it searches Swap Allocation Table if free count is not
                                    <1> ;;    zero. From begining the [swpd_next] dword value, the first bit
                                    <1> ;;    position with value of 1 on the table is converted to swap
                                    <1> ;;    disk/file offset address, in sectors (not 4096 bytes block).
                                    <1> ;;    'ldrv_write' procedure is called with ldrv (logical drive
                                    <1> ;;    number of physical swap disk or virtual swap disk)
                                    <1> ;;    number, sector offset (not absolute sector -LBA- number),
                                    <1> ;;    and sector count (8, 512*8 = 4096) and buffer adress
                                    <1> ;;    (memory page). That will be a direct disk write procedure.
                                    <1> ;;    (for preventing late memory allocation, significant waiting).
                                    <1> ;;    If disk write procedure returns with error or free count of
                                    <1> ;;    swap blocks is ZERO, 'swap_out' procedure will return with
                                    <1> ;;    'insufficient memory error' (cf=1).
                                    <1> ;;
                                    <1> ;;    (Note: Even if free swap disk/file blocks was not zero,
                                    <1> ;;    any disk write error will not be fixed by 'swap_out' procedure,
                                    <1> ;;    in other words, 'swap_out' will not check the table for other
                                    <1> ;;    free blocks after a disk write error. It will return to
                                    <1> ;;    the caller with error (CF=1) which means swapping is failed.
                                    <1> ;;
                                    <1> ;;    After writing the page on to swap disk/file address/sector,
                                    <1> ;;    'swap_out' procesure returns with that swap (offset) sector
                                    <1> ;;    address (cf=0).
                                    <1> ;;
                                    <1> ;;    ............................................................
                                    <1> ;;
                                    <1> ;;    'swap_in' procedure loads addressed (relevant) swap disk or
                                    <1> ;;    file sectors at specified memory page. Then page allocation
                                    <1> ;;    procedure updates relevant page table entry with 'present'
                                    <1> ;;    attribute. If swap disk or file reading fails there is nothing
                                    <1> ;;    to do, except to terminate the process which is the owner of
                                    <1> ;;    the swapped page.
                                    <1> ;;
                                    <1> ;;    'swap_in' procedure sets the relevant/respective bit value
                                    <1> ;;    in the Swap Allocation Table (as free block). 'swap_in' also
                                    <1> ;;    updates [swpd_first] pointer if it is required.
                                    <1> ;;
                                    <1> ;;    ............................................................
                                    <1> ;;
                                    <1> ;;    Note: If [swap_enabled] value is ZERO, that means there is not
                                    <1> ;;    a swap disk or swap file in use... 'swap_in' and 'swap_out'
                                    <1> ;;    procedures ans 'swap page que' procedures will not be active...
                                    <1> ;;    'Insufficient memory' error will be returned by 'swap_out'
                                    <1> ;;    and 'general protection fault' will be returned by 'swap_in'
                                    <1> ;;    procedure, if it is called mistakenly (a wrong value in a PTE).
                                    <1> ;;
                                    <1>
                                    <1> swap_in:
                                    <1>       ; 31/08/2015
                                    <1>       ; 20/07/2015
                                    <1>       ; 28/04/2015
                                    <1>       ; 18/04/2015
                                    <1>       ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
                                    <1>       ;
                                    <1>       ; INPUT ->
                                    <1>       ;     EBX = PHYSICAL (real/flat) ADDRESS OF THE MEMORY PAGE
                                    <1>       ;     EBP = VIRTUAL (LINEAR) ADDRESS (page fault address)
                                    <1>       ;     EAX = Offset Address for the swapped page on the
                                    <1>       ;           swap disk or in the swap file.
                                    <1>       ;
                                    <1>       ; OUTPUT ->
                                    <1>       ;     EAX = 0 if loading at memory has been successful
                                    <1>       ;
                                    <1>       ;     CF = 1 -> swap disk reading error (disk/file not present
                                    <1>       ;              or sector not present or drive not ready
                                    <1>       ;          EAX = Error code
                                    <1>       ;          [u.error] = EAX
                                    <1>       ;                    = The last error code for the process
                                    <1>       ;                      (will be reset after returning to user)
                                    <1>       ;
                                    <1>       ; Modified Registers -> EAX
                                    <1>       ;
                                    <1>
00004FB8 833D[62050300]00           <1>         cmp     dword [swp_drv], 0
00004FBF 7646                       <1>       jna     short swpin_dnp_err
                                    <1>
00004FC1 3B05[66050300]             <1>       cmp     eax, [swpd_size]
00004FC7 734A                       <1>       jnb     short swpin_snp_err
                                    <1>
00004FC9 56                         <1>       push    esi
00004FCA 53                         <1>       push    ebx
00004FCB 51                         <1>       push    ecx
00004FCC 8B35[62050300]             <1>       mov     esi, [swp_drv]
00004FD2 B908000000                 <1>       mov     ecx, PAGE_SIZE / LOGIC_SECT_SIZE  ; 8 !
                                    <1>               ; Note: Even if corresponding physical disk's sector
                                    <1>               ; size different than 512 bytes, logical disk sector
                                    <1>               ; size is 512 bytes and disk reading procedure
                                    <1>               ; will be performed for reading 4096 bytes
                                    <1>               ; (2*2048, 8*512).
                                    <1>       ; ESI = Logical disk description table address
```

```
                                <1>       ; EBX = Memory page (buffer) address (physical!)
                                <1>       ; EAX = Sector adress (offset address, logical sector number)
                                <1>       ; ECX = Sector count ; 8 sectors
00004FD7 50                     <1>       push   eax
00004FD8 E8AF020000             <1>       call   logical_disk_read
00004FDD 58                     <1>       pop    eax
00004FDE 730C                   <1>       jnc    short swpin_read_ok
                                <1>       ;
00004FE0 B828000000             <1>       mov    eax, SWP_DISK_READ_ERR ; drive not ready or read error
00004FE5 A3[C8030300]           <1>       mov    [u.error], eax
00004FEA EB17                   <1>       jmp    short swpin_retn
                                <1>       ;
                                <1> swpin_read_ok:
                                <1>       ; EAX = Offset address (logical sector number)
00004FEC E80D020000             <1>       call   unlink_swap_block  ; Deallocate swap block
                                <1>       ;
                                <1>       ; EBX = Memory page (buffer) address (physical!)
                                <1>       ; 20/07/2015
00004FF1 89EB                   <1>       mov    ebx, ebp ; virtual address (page fault address)
00004FF3 6681E300F0             <1>         and     bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
00004FF8 8A1D[B3030300]         <1>       mov    bl, [u.uno] ; current process number
                                <1>       ; EBX = Virtual (Linear) address & process number combination
00004FFE E8DB000000             <1>       call   swap_queue_shift
                                <1>       ; eax = 0 ; 10/06/2016 (if ebx input > 0, eax output = 0)
                                <1>       ;sub   eax, eax  ; 0 ; Error Code = 0  (no error)
                                <1>       ; zf = 1
                                <1> swpin_retn:
00005003 59                     <1>       pop    ecx
00005004 5B                     <1>       pop    ebx
00005005 5E                     <1>       pop    esi
00005006 C3                     <1>       retn
                                <1>
                                <1> swpin_dnp_err:
00005007 B829000000             <1>       mov    eax, SWP_DISK_NOT_PRESENT_ERR
                                <1> swpin_err_retn:
0000500C A3[C8030300]           <1>       mov    [u.error], eax
00005011 F9                     <1>       stc
00005012 C3                     <1>       retn
                                <1>
                                <1> swpin_snp_err:
00005013 B82A000000             <1>       mov    eax, SWP_SECTOR_NOT_PRESENT_ERR
00005018 EBF2                   <1>       jmp    short swpin_err_retn
                                <1>
                                <1> swap_out:
                                <1>       ; 10/06/2016
                                <1>       ; 07/06/2016
                                <1>         ; 23/05/2016
                                <1>       ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; 24/10/2014 - 31/08/2015 (Retro UNIX 386 v1)
                                <1>       ;
                                <1>       ; INPUT ->
                                <1>       ;     none
                                <1>       ;
                                <1>       ; OUTPUT ->
                                <1>       ;     EAX = Physical page address (which is swapped out
                                <1>       ;           for allocating a new page)
                                <1>       ;     CF = 1 -> swap disk writing error (disk/file not present
                                <1>       ;             or sector not present or drive not ready
                                <1>       ;          EAX = Error code
                                <1>       ;          [u.error] = EAX
                                <1>       ;                  = The last error code for the process
                                <1>       ;                    (will be reset after returning to user)
                                <1>       ;
                                <1>       ; Modified Registers -> none (except EAX)
                                <1>       ;
0000501A 66833D[60050300]01     <1>       cmp    word [swpq_count], 1
00005022 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>
00005028 833D[6A050300]01       <1>         cmp     dword [swpd_free], 1
0000502F 0F828F000000           <1>         jc      swpout_nfspc_err ; 'no free space on swap disk'
                                <1>
00005035 53                     <1>       push   ebx ; *
                                <1> swpout_1:
                                <1>       ; 10/06/2016
00005036 31DB                   <1>       xor    ebx, ebx ; shift the queue and return a PTE value
00005038 E8A1000000             <1>       call   swap_queue_shift
0000503D 21C0                   <1>       and    eax, eax     ; 0 = empty queue (improper entries)
0000503F 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
00005045 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:
00005049 52                     <1>       push   edx ; **
                                <1>
                                <1>       ; 07/06/2016
0000504A 803D[975F0100]00       <1>       cmp    byte [timer_events], 0
00005051 762F                   <1>       jna    short swpout_2
                                <1>       ;
00005053 8A15[975F0100]         <1>       mov    dl, [timer_events]
                                <1>
00005059 51                     <1>       push   ecx ; ***
0000505A 53                     <1>       push   ebx ; ****
0000505B BB[60040300]           <1>       mov    ebx, timer_set ; beginning address of timer event
```

```
                                <1>                             ; structures
                                <1> swpout_timer_page_1:
00005060 8A0B                   <1>       mov    cl, [ebx]
00005062 08C9                   <1>       or     cl, cl ; 0 = free, >0 = process number
00005064 7415                   <1>       jz     short swpout_timer_page_3
00005066 8B4B0C                 <1>       mov    ecx, [ebx+12] ; response (signal return) address
00005069 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)
0000506E 39C8                   <1>       cmp    eax, ecx
00005070 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>       ;
00005072 5B                     <1>       pop    ebx ; ****
00005073 59                     <1>       pop    ecx ; ***
00005074 5A                     <1>       pop    edx ; **
00005075 EBBF                   <1>       jmp    short swpout_1      ; do not swap out this page !
                                <1>
                                <1> swpout_timer_page_2:
                                <1>       ; 07/06/2016
00005077 FECA                   <1>       dec    dl
00005079 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
0000507B 83C310                 <1>       add    ebx, 16
0000507E EBE0                   <1>       jmp    short swpout_timer_page_1
                                <1>
                                <1> swpout_timer_page_4:
00005080 5B                     <1>       pop    ebx ; ****
00005081 59                     <1>       pop    ecx ; ***
                                <1> swpout_2:
00005082 89DA                   <1>       mov    edx, ebx            ; Page table entry address
00005084 89C3                   <1>       mov    ebx, eax            ; Buffer (Page) Address
                                <1>       ;
00005086 E8A6010000             <1>       call   link_swap_block
0000508B 7304                   <1>       jnc    short swpout_3             ; It may not be needed here
                                <1>                                   ; because [swpd_free] value
                                <1>                                   ; was checked at the beginging.
0000508D 5A                     <1>       pop    edx ; **
0000508E 5B                     <1>       pop    ebx ; *
0000508F EB33                   <1>       jmp    short swpout_nfspc_err
                                <1> swpout_3:
00005091 A900000080             <1>       test   eax, 80000000h ; test bit 31 (this may not be needed!)
00005096 752C                   <1>       jnz    short swpout_nfspc_err  ; 10/06/2016 (bit 31 = 1 !)
                                <1>       ;
00005098 56                     <1>       push   esi ; **
00005099 51                     <1>       push   ecx ; ***
0000509A 50                     <1>       push   eax ; sector address ; (31 bit !, bit 31 = 0)
0000509B 8B35[62050300]         <1>       mov    esi, [swp_drv]
000050A1 B908000000             <1>       mov    ecx, PAGE_SIZE / LOGIC_SECT_SIZE  ; 8 !
                                <1>              ; Note: Even if corresponding physical disk's sector
                                <1>              ; size different than 512 bytes, logical disk sector
                                <1>              ; size is 512 bytes and disk writing procedure
                                <1>              ; will be performed for writing 4096 bytes
                                <1>              ; (2*2048, 8*512).
                                <1>       ; ESI = Logical disk description table address
                                <1>       ; EBX = Buffer (Page) address
                                <1>       ; EAX = Sector adress (offset address, logical sector number)
                                <1>       ; ECX = Sector count ; 8 sectors
                                <1>       ; edx = PTE address
000050A6 E8E2010000             <1>       call   logical_disk_write
                                <1>       ; edx = PTE address
000050AB 59                     <1>       pop    ecx ; sector address
000050AC 730C                   <1>       jnc    short swpout_write_ok
                                <1>       ;
                                <1>       ;; call      unlink_swap_block ; this block must be left as 'in use'
                                <1> swpout_dw_err:
000050AE B82C000000             <1>       mov    eax, SWP_DISK_WRITE_ERR ; drive not ready or write error
000050B3 A3[C8030300]           <1>       mov    [u.error], eax
000050B8 EB06                   <1>       jmp    short swpout_retn
                                <1>       ;
                                <1> swpout_write_ok:
                                <1>       ; EBX = Buffer (page) address
                                <1>       ; EDX = Page Table Entry address
                                <1>       ; ECX = Swap disk sector (file block) address (31 bit)
000050BA D1E1                   <1>       shl    ecx, 1  ; 31 bit sector address from bit 1 to bit 31
000050BC 890A                   <1>       mov    [edx], ecx
                                <1>              ; bit 0 = 0 (swapped page)
000050BE 89D8                   <1>       mov    eax, ebx
                                <1> swpout_retn:
000050C0 59                     <1>       pop    ecx ; ***
000050C1 5E                     <1>       pop    esi ; **
000050C2 5B                     <1>       pop    ebx ; *
000050C3 C3                     <1>       retn
                                <1>
                                <1> ;swpout_dnp_err:
                                <1> ;     mov    eax, SWP_DISK_NOT_PRESENT_ERR ; disk not present
                                <1> ;     jmp    short swpout_err_retn
                                <1> swpout_nfspc_err:
000050C4 B82B000000             <1>       mov    eax, SWP_NO_FREE_SPACE_ERR ; no free space
                                <1> swpout_err_retn:
000050C9 A3[C8030300]           <1>       mov    [u.error], eax
                                <1>       ;stc
```

```
1964 000050CE C3                <1>       retn
1965                            <1> swpout_npts_err:
1966 000050CF B82D000000        <1>       mov    eax, SWP_NO_PAGE_TO_SWAP_ERR
1967 000050D4 5B                <1>       pop    ebx
1968 000050D5 EBF2              <1>       jmp    short swpout_err_retn
1969                            <1> swpout_im_err:
1970 000050D7 B804000000        <1>       mov    eax, ERR_MINOR_IM ; insufficient (out of) memory
1971 000050DC EBEB              <1>       jmp    short swpout_err_retn
1972                            <1>
1973                            <1> swap_queue_shift:
1974                            <1>       ; 26/03/2017
1975                            <1>       ; 10/06/2016
1976                            <1>       ; 09/06/2016 - TRDOS 386 (TRDOS v2.0)
1977                            <1>       ; 23/10/2014 - 20/07/2015 (Retro UNIX 386 v1)
1978                            <1>       ;
1979                            <1>       ; INPUT ->
1980                            <1>       ;     EBX = Virtual (linear) address (bit 12 to 31)
1981                            <1>       ;           and process number combination (bit 0 to 11)
1982                            <1>       ;     EBX = 0 -> shift/drop from the head (offset 0)
1983                            <1>       ;
1984                            <1>       ; OUTPUT ->
1985                            <1>       ;     If EBX input > 0
1986                            <1>       ;        the queue will be shifted 4 bytes (dword),
1987                            <1>       ;        from the tail to the head, up to entry offset
1988                            <1>       ;        which points to EBX input value or nothing
1989                            <1>       ;        to do if EBX value is not found on the queue.
1990                            <1>       ;        (The entry -with EBX value- will be removed
1991                            <1>       ;        from the queue if it is found.)
1992                            <1>       ;
1993                            <1>       ;        EAX = 0
1994                            <1>       ;
1995                            <1>       ;     If EBX input = 0
1996                            <1>       ;        the queue will be shifted 4 bytes (dword),
1997                            <1>       ;        from the tail to the head, if the PTE address
1998                            <1>       ;        which is pointed in head of the queue is marked
1999                            <1>       ;        as "accessed" or it is marked as "non present".
2000                            <1>       ;        (If "accessed" flag of the PTE -which is pointed
2001                            <1>       ;        in the head- is set -to 1-, it will be reset
2002                            <1>       ;        -to 0- and then, the queue will be rotated
2003                            <1>       ;        -without dropping pointer of the PTE from
2004                            <1>       ;        the queue- for 4 bytes on head to tail direction.
2005                            <1>       ;        Pointer in the head will be moved into the tail,
2006                            <1>       ;        other PTEs will be shifted on head direction.)
2007                            <1>       ;
2008                            <1>       ;        Swap queue will be shifted up to the first
2009                            <1>       ;        'present' or 'non accessed' page will be found
2010                            <1>       ;        (as pointed) on the queue head (then it will be
2011                            <1>         ;          removed/dropped from the queue).
2012                            <1>       ;
2013                            <1>       ;        EAX (> 0) = PTE value of the page which is
2014                            <1>       ;             (it's pointer -virtual address-) dropped
2015                            <1>       ;             (removed) from swap queue.
2016                            <1>       ;        EBX = PTE address of the page (if EAX > 0)
2017                            <1>       ;              which is (it's pointer -virtual address-)
2018                            <1>       ;             dropped (removed) from swap queue.
2019                            <1>       ;
2020                            <1>       ;        EAX = 0 -> empty swap queue !
2021                            <1>       ;
2022                            <1>       ; Modified Registers -> EAX, EBX
2023                            <1>       ;
2024 000050DE 0FB705[60050300]  <1>       movzx   eax, word [swpq_count]  ; Max. 1024
2025 000050E5 6621C0            <1>       and    ax, ax
2026 000050E8 7431              <1>       jz     short swpqs_retn
2027 000050EA 57                <1>       push   edi
2028 000050EB 56                <1>       push   esi
2029 000050EC 51                <1>       push   ecx
2030 000050ED BE00E00800        <1>       mov    esi, swap_queue
2031 000050F2 89C1              <1>       mov    ecx, eax
2032 000050F4 09DB              <1>       or     ebx, ebx
2033 000050F6 7424              <1>       jz     short swpqs_7
2034                            <1> swpqs_1:
2035 000050F8 AD                <1>       lodsd
2036 000050F9 39D8              <1>       cmp    eax, ebx
2037 000050FB 7406              <1>       je     short swpqs_2
2038 000050FD E2F9              <1>       loop   swpqs_1
2039                            <1>       ; 10/06/2016
2040 000050FF 29C0              <1>       sub    eax, eax
2041 00005101 EB15              <1>       jmp    short swpqs_6
2042                            <1> swpqs_2:
2043 00005103 89F7              <1>       mov    edi, esi
2044 00005105 83EF04            <1>       sub    edi, 4
2045                            <1> swpqs_3:
2046 00005108 66FF0D[60050300]  <1>       dec    word [swpq_count]
2047 0000510F 7403              <1>       jz     short swpqs_5
2048                            <1> swpqs_4:
2049 00005111 49                <1>       dec    ecx
2050 00005112 F3A5              <1>       rep    movsd  ; shift up (to the head)
2051                            <1> swpqs_5:
2052 00005114 31C0              <1>       xor    eax, eax
2053 00005116 8907              <1>       mov    [edi], eax
2054                            <1> swpqs_6:
2055 00005118 59                <1>       pop    ecx
2056 00005119 5E                <1>       pop    esi
2057 0000511A 5F                <1>       pop    edi
2058                            <1> swpqs_retn:
2059 0000511B C3                <1>       retn
2060                            <1> swpqs_7:
2061 0000511C 89F7              <1>       mov    edi, esi ; head
2062 0000511E AD                <1>       lodsd
2063                            <1>       ; 20/07/2015
2064 0000511F 89C3              <1>       mov    ebx, eax
2065 00005121 81E300F0FFFF      <1>       and    ebx, ~PAGE_OFF ; ~0FFFh
```

```
                                <1>                     ; ebx = virtual address (at page boundary)
00005127 25FF0F0000             <1>       and   eax, PAGE_OFF ; 0FFFh
                                <1>                     ; ax = process number (1 to 4095)
0000512C 3A05[B3030300]         <1>       cmp   al, [u.uno]
                                <1>             ; Max. 16 (nproc) processes for Retro UNIX 386 v1
00005132 7507                   <1>       jne   short swpqs_8
00005134 A1[B8030300]           <1>       mov   eax, [u.pgdir]
00005139 EB28                   <1>       jmp   short swpqs_9
                                <1> swpqs_8:
                                <1>       ; 09/06/2016
0000513B 80B8[AF000300]00       <1>       cmp   byte [eax+p.stat-1], 0
00005142 76C4                   <1>       jna   short swpqs_3     ; free (or terminated) process
00005144 80B8[AF000300]02       <1>       cmp   byte [eax+p.stat-1], 2 ; waiting
0000514B 77BB                   <1>       ja    short swpqs_3         ; zombie (3) or undefined ?
                                <1>
                                <1>       ;shl   ax, 2
0000514D C0E002                 <1>       shl   al, 2
00005150 8B80[BC000300]         <1>       mov   eax, [eax+p.upage-4]
00005156 09C0                   <1>       or    eax, eax
00005158 74AE                   <1>       jz    short swpqs_3 ; invalid upage
0000515A 83C05C                 <1>       add   eax, u.pgdir - user
                                <1>                     ; u.pgdir value for the process
                                <1>                     ; is in [eax]
0000515D 8B00                   <1>       mov   eax, [eax]
0000515F 21C0                   <1>       and   eax, eax
00005161 74A5                   <1>       jz    short swpqs_3 ; invalid page directory
                                <1> swpqs_9:
00005163 52                     <1>       push  edx
                                <1>       ; eax = page directory
                                <1>       ; ebx = virtual address
00005164 E82BFBFFFF             <1>       call  get_pte
00005169 89D3                   <1>       mov   ebx, edx     ; PTE address
0000516B 5A                     <1>       pop   edx
                                <1>       ; 10/06/2016
0000516C 723A                   <1>       jc    short swpqs_13 ; empty PDE
                                <1>       ; EAX = PTE value
0000516E A801                   <1>       test  al, PTE_A_PRESENT ; bit 0 = 1
00005170 7436                   <1>       jz    short swpqs_13  ; Drop non-present page
                                <1>                             ; from the queue (head)
00005172 A802                   <1>       test  al, PTE_A_WRITE     ; bit 1 = 0 (read only)
00005174 7432                   <1>       jz    short swpqs_13  ; Drop read only page
                                <1>                             ; from the queue (head)
                                <1>       ;test  al, PTE_A_ACCESS ; bit 5 = 1 (Accessed)
                                <1>       ;jnz   short swpqs_11  ; present
                                <1>                             ; accessed page
00005176 0FBAF005               <1>         btr     eax, PTE_A_ACCESS_BIT ; reset 'accessed' bit
0000517A 7210                   <1>       jc    short swpqs_11  ; accessed page
                                <1>
0000517C 49                     <1>       dec   ecx
0000517D 66890D[60050300]       <1>       mov   [swpq_count], cx
00005184 7402                   <1>         jz      short swpqs_10
                                <1>             ; esi = head + 4
                                <1>             ; edi = head
00005186 F3A5                   <1>       rep   movsd   ; n = 1 to k-1, [n - 1] = [n]
                                <1> swpqs_10:
00005188 890F                   <1>       mov   [edi], ecx ; 0
0000518A EB8C                   <1>       jmp   short swpqs_6 ; 26/03/2017
                                <1>
                                <1> swpqs_11:
0000518C 8903                   <1>       mov   [ebx], eax     ; save changed attribute
                                <1>       ; Rotation (head -> tail)
0000518E 49                     <1>       dec   ecx     ; entry count -> last entry number
0000518F 74F7                   <1>       jz    short swpqs_10
                                <1>             ; esi = head + 4
                                <1>             ; edi = head
00005191 8B07                   <1>       mov   eax, [edi] ; 20/07/2015
00005193 F3A5                   <1>       rep   movsd   ; n = 1 to k-1, [n - 1] = [n]
00005195 8907                   <1>       mov   [edi], eax ; head -> tail ; [k] = [1]
                                <1>
00005197 668B0D[60050300]       <1>       mov   cx, [swpq_count]
                                <1>
                                <1> swpqs_12:
0000519E BE00E00800             <1>       mov   esi, swap_queue ; head
000051A3 E974FFFFFF             <1>         jmp     swpqs_7
                                <1>
                                <1> swpqs_13:
000051A8 49                     <1>       dec   ecx
000051A9 66890D[60050300]       <1>       mov   [swpq_count], cx
000051B0 0F845EFFFFFF           <1>         jz      swpqs_5
000051B6 EBE6                   <1>       jmp   short swpqs_12
                                <1>
                                <1> add_to_swap_queue:
                                <1> ; temporary - 16/09/2015
000051B8 C3                     <1> retn
                                <1>       ; 20/02/2017
                                <1>       ; 20/07/2015
                                <1>       ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
                                <1>       ;
                                <1>       ; Adds new page to swap queue
                                <1>       ; (page directories and page tables must not be added
                                <1>       ; to swap queue)
                                <1>       ;
                                <1>       ; INPUT ->
                                <1>       ;     EBX = Linear (Virtual) addr for current process
                                <1>       ;     [u.uno]
                                <1>       ;     20/02/2017
                                <1>       ;     (Linear address = CORE + user's virtual address)
                                <1>       ;
                                <1>       ; OUTPUT ->
                                <1>       ;     EAX = [swpq_count]
                                <1>       ;           (after the PTE has been added)
                                <1>       ;     EAX = 0 -> Swap queue is full, (1024 entries)
```

```
2168                                    <1>       ;           the PTE could not be added.
2169                                    <1>       ;
2170                                    <1>       ; Modified Registers -> EAX
2171                                    <1>       ;
2172 000051B9 53                       <1>       push   ebx
2173 000051BA 6681E300F0                <1>         and     bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
2174 000051BF 8A1D[B3030300]            <1>       mov    bl, [u.uno] ; current process number
2175 000051C5 E814FFFFFF                <1>       call   swap_queue_shift ; drop from the queue if
2176                                    <1>                              ; it is already on the queue
2177                                    <1>                 ; then add it to the tail of the queue
2178 000051CA 0FB705[60050300]          <1>       movzx  eax, word [swpq_count]
2179 000051D1 663D0004                  <1>       cmp    ax, 1024
2180 000051D5 7205                      <1>       jb     short atsq_1
2181 000051D7 6629C0                    <1>       sub    ax, ax
2182 000051DA 5B                        <1>       pop    ebx
2183 000051DB C3                        <1>       retn
2184                                    <1> atsq_1:
2185 000051DC 56                        <1>       push   esi
2186 000051DD BE00E00800                <1>       mov    esi, swap_queue
2187 000051E2 6621C0                    <1>       and    ax, ax
2188 000051E5 740A                      <1>       jz     short atsq_2
2189 000051E7 66C1E002                  <1>       shl    ax, 2  ; convert to offset
2190 000051EB 01C6                      <1>       add    esi, eax
2191 000051ED 66C1E802                  <1>       shr    ax, 2
2192                                    <1> atsq_2:
2193 000051F1 6640                      <1>       inc    ax
2194 000051F3 891E                      <1>       mov    [esi], ebx ; Virtual address + [u.uno] combination
2195 000051F5 66A3[60050300]            <1>       mov    [swpq_count], ax
2196 000051FB 5E                        <1>       pop    esi
2197 000051FC 5B                        <1>       pop    ebx
2198 000051FD C3                        <1>       retn
2199                                    <1>
2200                                    <1> unlink_swap_block:
2201                                    <1>       ; 15/09/2015
2202                                    <1>       ; 30/04/2015
2203                                    <1>       ; 18/04/2015
2204                                    <1>       ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
2205                                    <1>       ;
2206                                    <1>       ; INPUT ->
2207                                    <1>       ;     EAX = swap disk/file offset address
2208                                    <1>       ;           (bit 1 to bit 31)
2209                                    <1>       ; OUTPUT ->
2210                                    <1>       ;     [swpd_free] is increased
2211                                    <1>       ;     (corresponding SWAP DISK ALLOC. TABLE bit is SET)
2212                                    <1>       ;
2213                                    <1>       ; Modified Registers -> EAX
2214                                    <1>       ;
2215 000051FE 53                       <1>       push   ebx
2216 000051FF 52                        <1>       push   edx
2217                                    <1>       ;
2218 00005200 C1E804                    <1>       shr    eax, SECTOR_SHIFT+1  ;3+1 ; shift sector address to
2219                                    <1>                                   ; 3 bits right
2220                                    <1>                                   ; to get swap block/page number
2221 00005203 89C2                      <1>       mov    edx, eax
2222                                    <1>       ; 15/09/2015
2223 00005205 C1EA03                    <1>       shr    edx, 3            ; to get offset to S.A.T.
2224                                    <1>                                   ; (1 allocation bit = 1 page)
2225                                    <1>                                   ; (1 allocation bytes = 8 pages)
2226 00005208 80E2FC                    <1>       and    dl, 0FCh          ; clear lower 2 bits
2227                                    <1>                                   ; (to get 32 bit position)
2228                                    <1>       ;
2229 0000520B BB00000D00                <1>       mov    ebx, swap_alloc_table ; Swap Allocation Table address
2230 00005210 01D3                      <1>       add    ebx, edx
2231 00005212 83E01F                    <1>       and    eax, 1Fh          ; lower 5 bits only
2232                                    <1>                                   ; (allocation bit position)
2233 00005215 3B05[6E050300]            <1>       cmp    eax, [swpd_next]     ; is the new free block addr. lower
2234                                    <1>                                   ; than the address in 'swpd_next' ?
2235                                    <1>                                   ; (next/first free block value)
2236 0000521B 7305                      <1>       jnb    short uswpbl_1          ; no
2237 0000521D A3[6E050300]              <1>       mov    [swpd_next], eax     ; yes
2238                                    <1> uswpbl_1:
2239 00005222 0FAB03                    <1>       bts    [ebx], eax        ; unlink/release/deallocate block
2240                                    <1>                                   ; set relevant bit to 1.
2241                                    <1>                                   ; set CF to the previous bit value
2242 00005225 F5                        <1>       cmc                      ; complement carry flag
2243 00005226 7206                      <1>       jc     short uswpbl_2          ; do not increase swfd_free count
2244                                    <1>                                   ; if the block is already deallocated
2245                                    <1>                                   ; before.
2246 00005228 FF05[6A050300]            <1>         inc     dword [swpd_free]
2247                                    <1> uswpbl_2:
2248 0000522E 5A                        <1>       pop    edx
2249 0000522F 5B                        <1>       pop    ebx
2250 00005230 C3                        <1>       retn
2251                                    <1>
2252                                    <1> link_swap_block:
2253                                    <1>       ; 01/07/2015
2254                                    <1>       ; 18/04/2015
2255                                    <1>       ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
2256                                    <1>       ;
2257                                    <1>       ; INPUT -> none
2258                                    <1>       ;
2259                                    <1>       ; OUTPUT ->
2260                                    <1>       ;     EAX = OFFSET ADDRESS OF THE ALLOCATED BLOCK (4096 bytes)
2261                                    <1>       ;           in sectors (corresponding
2262                                    <1>       ;           SWAP DISK ALLOCATION TABLE bit is RESET)
2263                                    <1>       ;
2264                                    <1>       ;     CF = 1 and EAX = 0
2265                                    <1>       ;               if there is not a free block to be allocated
2266                                    <1>       ;
2267                                    <1>       ; Modified Registers -> none (except EAX)
2268                                    <1>       ;
2269                                    <1>
```

```
2270                            <1>       ;mov   eax, [swpd_free]
2271                            <1>       ;and   eax, eax
2272                            <1>       ;jz    short out_of_swpspc
2273                            <1>       ;
2274 00005231 53                <1>       push   ebx
2275 00005232 51                <1>       push   ecx
2276                            <1>       ;
2277 00005233 BB00000D00        <1>       mov    ebx, swap_alloc_table ; Swap Allocation Table offset
2278 00005238 89D9              <1>       mov    ecx, ebx
2279 0000523A 031D[6E050300]    <1>       add    ebx, [swpd_next] ; Free block searching starts from here
2280                            <1>                               ; next_free_swap_block >> 5
2281 00005240 030D[72050300]    <1>       add    ecx, [swpd_last] ; Free block searching ends here
2282                            <1>                               ; (total_swap_blocks - 1) >> 5
2283                            <1> lswbl_scan:
2284 00005246 39CB              <1>       cmp    ebx, ecx
2285 00005248 770A              <1>       ja     short lswbl_notfound
2286                            <1>       ;
2287 0000524A 0FBC03            <1>       bsf    eax, [ebx] ; Scans source operand for first bit set (1).
2288                            <1>                         ; Clears ZF if a bit is found set (1) and
2289                            <1>                         ; loads the destination with an index to
2290                            <1>                         ; first set bit. (0 -> 31)
2291                            <1>                         ; Sets ZF to 1 if no bits are found set.
2292                            <1>       ; 01/07/2015
2293 0000524D 751C              <1>       jnz    short lswbl_found ; ZF = 0 -> a free block has been found
2294                            <1>                          ;
2295                            <1>                          ; NOTE:  a Swap Disk Allocation Table bit
2296                            <1>                          ;       with value of 1 means
2297                            <1>                          ;       the corresponding page is free
2298                            <1>                          ;       (Retro UNIX 386 v1 feaure only!)
2299 0000524F 83C304            <1>       add    ebx, 4
2300                            <1>                          ; We return back for searching next page block
2301                            <1>                          ; NOTE: [swpd_free] is not ZERO; so,
2302                            <1>                          ;     we always will find at least 1 free block here.
2303 00005252 EBF2              <1>       jmp          short lswbl_scan
2304                            <1>       ;
2305                            <1> lswbl_notfound:
2306 00005254 81E900000D00      <1>       sub    ecx, swap_alloc_table
2307 0000525A 890D[6E050300]    <1>       mov    [swpd_next], ecx ; next/first free page = last page
2308                            <1>                               ; (unlink_swap_block procedure will change it)
2309 00005260 31C0              <1>       xor    eax, eax
2310 00005262 A3[6A050300]      <1>       mov    [swpd_free], eax
2311 00005267 F9                <1>       stc
2312                            <1> lswbl_ok:
2313 00005268 59                <1>       pop    ecx
2314 00005269 5B                <1>       pop    ebx
2315 0000526A C3                <1>       retn
2316                            <1>       ;
2317                            <1> ;out_of_swpspc:
2318                            <1> ;     stc
2319                            <1> ;     retn
2320                            <1>
2321                            <1> lswbl_found:
2322 0000526B 89D9              <1>       mov    ecx, ebx
2323 0000526D 81E900000D00      <1>       sub    ecx, swap_alloc_table
2324 00005273 890D[6E050300]    <1>       mov    [swpd_next], ecx ; Set first free block searching start
2325                            <1>                               ; address/offset (to the next)
2326 00005279 FF0D[6A050300]    <1>         dec     dword [swpd_free] ; 1 block has been allocated (X = X-1)
2327                            <1>       ;
2328 0000527F 0FB303            <1>       btr    [ebx], eax     ; The destination bit indexed by the source value
2329                            <1>                               ; is copied into the Carry Flag and then cleared
2330                            <1>                               ; in the destination.
2331                            <1>                               ;
2332                            <1>                               ; Reset the bit which is corresponding to the
2333                            <1>                               ; (just) allocated block.
2334 00005282 C1E105              <1>       shl    ecx, 5         ; (block offset * 32) + block index
2335 00005285 01C8              <1>       add    eax, ecx       ; = block number
2336 00005287 C1E003            <1>       shl    eax, SECTOR_SHIFT ; 3, sector (offset) address of the block
2337                            <1>                               ; 1 block =  8 sectors
2338                            <1>       ;
2339                            <1>       ; EAX = offset address of swap disk/file sector (beginning of the block)
2340                            <1>       ;
2341                            <1>       ; NOTE: The relevant page table entry will be updated
2342                            <1>       ;       according to this EAX value...
2343                            <1>       ;
2344 0000528A EBDC              <1>       jmp    short lswbl_ok
2345                            <1>
2346                            <1> logical_disk_read:
2347                            <1>       ; 20/07/2015
2348                            <1>       ; 09/03/2015 (temporary code here)
2349                            <1>       ;
2350                            <1>       ; INPUT ->
2351                            <1>       ;      ESI = Logical disk description table address
2352                            <1>       ;      EBX = Memory page (buffer) address (physical!)
2353                            <1>       ;      EAX = Sector adress (offset address, logical sector number)
2354                            <1>       ;      ECX = Sector count
2355                            <1>       ;
2356                            <1>       ;
2357 0000528C C3                <1>       retn
2358                            <1>
2359                            <1> logical_disk_write:
2360                            <1>       ; 20/07/2015
2361                            <1>       ; 09/03/2015 (temporary code here)
2362                            <1>       ;
2363                            <1>       ; INPUT ->
2364                            <1>       ;      ESI = Logical disk description table address
2365                            <1>       ;      EBX = Memory page (buffer) address (physical!)
2366                            <1>       ;      EAX = Sector adress (offset address, logical sector number)
2367                            <1>       ;      ECX = Sector count
2368                            <1>       ;
2369 0000528D C3                <1>       retn
2370                            <1>
2371                            <1> get_physical_addr:
```

```
                                <1>       ; 26/03/2017
                                <1>       ; 20/02/2017
                                <1>       ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; 18/10/2015
                                <1>       ; 29/07/2015
                                <1>       ; 20/07/2015
                                <1>       ; 04/06/2015
                                <1>       ; 20/05/2015
                                <1>       ; 28/04/2015
                                <1>       ; 18/04/2015
                                <1>       ; Get physical address
                                <1>       ;     (allocates a new page for user if it is not present)
                                <1>       ;
                                <1>       ; (This subroutine is needed for mapping user's virtual
                                <1>       ; (buffer) address to physical address (of the buffer).)
                                <1>       ; ('sys write', 'sys read' system calls...)
                                <1>       ;
                                <1>       ; INPUT ->
                                <1>       ;     EBX = virtual address
                                <1>       ;     u.pgdir = page directory (physical) address
                                <1>       ;
                                <1>       ; OUTPUT ->
                                <1>       ;     EAX = physical address
                                <1>       ;     EBX = linear address
                                <1>       ;     EDX = physical address of the page frame
                                <1>       ;           (with attribute bits)
                                <1>       ;     ECX = byte count within the page frame
                                <1>       ;
                                <1>       ; Modified Registers -> EAX, EBX, ECX, EDX
                                <1>       ;
0000528E 81C300004000           <1>       add   ebx, CORE ; 18/10/2015
                                <1> get_physical_addr_x: ; 27/05/2016
00005294 A1[B8030300]           <1>       mov   eax, [u.pgdir]
00005299 E8F6F9FFFF             <1>       call  get_pte
                                <1>             ; EDX = Page table entry address (if CF=0)
                                <1>               ;       Page directory entry address (if CF=1)
                                <1>             ;       (Bit 0 value is 0 if PT is not present)
                                <1>             ; EAX = Page table entry value (page address)
                                <1>             ;     CF = 1 -> PDE not present or invalid ?
0000529E 731C                   <1>       jnc   short gpa_1
                                <1>       ;
000052A0 E8D4F8FFFF             <1>       call  allocate_page
000052A5 7248                   <1>       jc    short gpa_im_err  ; 'insufficient memory' error
                                <1> gpa_0:
000052A7 E847F9FFFF             <1>       call  clear_page
                                <1>       ; EAX = Physical (base) address of the allocated (new) page
000052AC 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)
000052AE 8902                   <1>       mov   [edx], eax ; Let's put the new page directory entry here !
000052B0 A1[B8030300]           <1>       mov   eax, [u.pgdir]
000052B5 E8DAF9FFFF             <1>       call  get_pte
000052BA 7233                   <1>       jc    short gpa_im_err ; 'insufficient memory' error
                                <1> gpa_1:
                                <1>       ; EAX = PTE value, EDX = PTE address
000052BC A801                   <1>       test  al, PTE_A_PRESENT
000052BE 751F                   <1>       jnz   short gpa_3 ; 26/03/2017
000052C0 09C0                   <1>       or    eax, eax
000052C2 7456                   <1>       jz    short gpa_7  ; Allocate a new page
                                <1>       ; 20/07/2015
000052C4 55                     <1>       push  ebp
000052C5 89DD                   <1>       mov   ebp, ebx ; virtual (linear) address
                                <1>       ; reload swapped page
000052C7 E878000000             <1>       call  reload_page ; 28/04/2015
000052CC 5D                     <1>       pop   ebp
000052CD 724A                   <1>       jc    short gpa_retn
                                <1> gpa_2:
                                <1>       ; 26/03/2017
                                <1>       ; 20/02/2017
                                <1>       ; If a page will contain a Signal Response Byte
                                <1>       ; it must not be swapped out, because
                                <1>       ; timer service or irq callback service
                                <1>       ; will write a signal return/response byte
                                <1>       ; directly by using physical address of Signal
                                <1>       ; Response Byte.(Even if process is not running,
                                <1>       ; or it is running with swapped out pages.)
                                <1>       ;
                                <1>       ; 'no_page_swap' will be set by 'systimer' or
                                <1>       ; 'syscalbac' sistem functions/calls. (*)
                                <1>       ;
000052CF 803D[D6640100]00       <1>       cmp   byte [no_page_swap], 0
000052D6 761D                   <1>       jna   short gpa_4 ; this page can be swapped out
                                <1>       ; this page must not be swapped out
                                <1>       ; but 'no_page_swap' must be reset here
                                <1>       ; imediately for other callers (*)
                                <1>       ; (otherwise, swap queue would not be long enough)
000052D8 E84B000000             <1>       call  gpa_8 ; 26/03/2017
000052DD EB1D                   <1>       jmp   short gpa_5
                                <1> gpa_3:
                                <1>       ; 26/03/2017
000052DF 803D[D6640100]00       <1>       cmp   byte [no_page_swap], 0
000052E6 7618                   <1>       jna   short gpa_6 ; this page can be swapped out
000052E8 E83B000000             <1>       call  gpa_8
000052ED EB11                   <1>       jmp   short gpa_6
                                <1>
                                <1> gpa_im_err:
000052EF B804000000             <1>       mov   eax, ERR_MINOR_IM ; Insufficient memory (minor) error!
                                <1>                               ; Major error = 0 (No protection fault)
000052F4 C3                     <1>       retn
                                <1> gpa_4:
                                <1>       ; 20/07/2015
                                <1>       ; 20/05/2015
```

```
2474                                    <1>       ; add this page to swap queue
2475 000052F5 50                       <1>       push   eax
2476                                    <1>       ; EBX = Linear (CORE+virtual) address ; 20/02/2017
2477 000052F6 E8BDFEFFFF               <1>       call   add_to_swap_queue
2478 000052FB 58                       <1>       pop    eax
2479                                    <1> gpa_5:
2480                                    <1>              ; PTE address in EDX
2481                                    <1>              ; virtual address in EBX
2482                                    <1>       ; EAX = memory page address
2483 000052FC 0C07                     <1>       or     al, PTE_A_PRESENT + PTE_A_USER + PTE_A_WRITE
2484                                    <1>                               ; present flag, bit 0 = 1
2485                                    <1>                               ; user flag, bit 2 = 1
2486                                    <1>                               ; writable flag, bit 1 = 1
2487 000052FE 8902                     <1>       mov    [edx], eax  ; Update PTE value
2488                                    <1> gpa_6:
2489                                    <1>       ; 18/10/2015
2490 00005300 89D9                     <1>       mov    ecx, ebx
2491 00005302 81E1FF0F0000             <1>       and    ecx, PAGE_OFF
2492 00005308 89C2                     <1>       mov    edx, eax
2493 0000530A 662500F0                 <1>       and    ax, PTE_A_CLEAR
2494 0000530E 01C8                     <1>       add    eax, ecx
2495 00005310 F7D9                     <1>       neg    ecx ; 1 -> -1 (0FFFFFFFFh), 4095 (0FFFh) -> -4095
2496 00005312 81C100100000             <1>       add    ecx, PAGE_SIZE
2497 00005318 F8                       <1>       clc
2498                                    <1> gpa_retn:
2499 00005319 C3                       <1>       retn
2500                                    <1> gpa_7:
2501 0000531A E85AF8FFFF               <1>       call   allocate_page
2502 0000531F 72CE                     <1>       jc     short gpa_im_err ; 'insufficient memory' error
2503 00005321 E8CDF8FFFF               <1>       call   clear_page
2504 00005326 EBA7                     <1>       jmp    short gpa_2
2505                                    <1>
2506                                    <1> gpa_8: ; 26/03/2017
2507 00005328 C605[D6640100]00         <1>       mov    byte [no_page_swap], 0
2508 0000532F 53                       <1>       push   ebx
2509 00005330 50                       <1>       push   eax ; 26/03/2017
2510 00005331 6681E300F0               <1>         and     bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
2511 00005336 8A1D[B3030300]           <1>       mov    bl, [u.uno] ; current process number
2512 0000533C E89DFDFFFF               <1>       call   swap_queue_shift ; drop from the queue if
2513                                    <1>                               ; it is already on the queue
2514 00005341 58                       <1>       pop    eax ; 26/03/2017
2515 00005342 5B                       <1>       pop    ebx
2516 00005343 C3                       <1>       retn
2517                                    <1>
2518                                    <1> reload_page:
2519                                    <1>       ; 20/07/2015
2520                                    <1>       ; 28/04/2015 (Retro UNIX 386 v1 - beginning)
2521                                    <1>       ;
2522                                    <1>       ; Reload (Restore) swapped page at memory
2523                                    <1>       ;
2524                                    <1>       ; INPUT ->
2525                                    <1>       ;      EBP = Virtual (linear) memory address
2526                                    <1>       ;      EAX = PTE value (swap disk sector address)
2527                                    <1>       ;      (Swap disk sector address = bit 1 to bit 31 of EAX)
2528                                    <1>       ; OUTPUT ->
2529                                    <1>       ;      EAX = PHYSICAL (real/flat) ADDRESS OF RELOADED PAGE
2530                                    <1>       ;
2531                                    <1>       ;      CF = 1 and EAX = error code
2532                                    <1>       ;
2533                                    <1>       ; Modified Registers -> none (except EAX)
2534                                    <1>       ;
2535 00005344 D1E8                     <1>       shr    eax, 1   ; Convert PTE value to swap disk address
2536 00005346 53                       <1>       push   ebx      ;
2537 00005347 89C3                     <1>       mov    ebx, eax ; Swap disk (offset) address
2538 00005349 E82BF8FFFF               <1>       call   allocate_page
2539 0000534E 720C                     <1>       jc     short rlp_im_err
2540 00005350 93                       <1>       xchg   eax, ebx
2541                                    <1>       ; EBX = Physical memory (page) address
2542                                    <1>       ; EAX = Swap disk (offset) address
2543                                    <1>       ; EBP = Virtual (linear) memory address
2544 00005351 E862FCFFFF               <1>       call   swap_in
2545 00005356 720B                     <1>       jc     short rlp_swp_err  ; (swap disk/file read error)
2546 00005358 89D8                     <1>       mov    eax, ebx
2547                                    <1> rlp_retn:
2548 0000535A 5B                       <1>       pop    ebx
2549 0000535B C3                       <1>       retn
2550                                    <1>
2551                                    <1> rlp_im_err:
2552 0000535C B804000000               <1>       mov    eax, ERR_MINOR_IM ; Insufficient memory (minor) error!
2553                                    <1>                              ; Major error = 0 (No protection fault)
2554 00005361 EBF7                     <1>       jmp    short rlp_retn
2555                                    <1>
2556                                    <1> rlp_swp_err:
2557 00005363 B828000000               <1>       mov    eax, SWP_DISK_READ_ERR ; Swap disk read error !
2558 00005368 EBF0                     <1>       jmp    short rlp_retn
2559                                    <1>
2560                                    <1>
2561                                    <1> copy_page_dir:
2562                                    <1>       ; 19/09/2015
2563                                    <1>       ; temporary - 07/09/2015
2564                                    <1>       ; 07/09/2015 (Retro UNIX 386 v1 - beginning)
2565                                    <1>       ;
2566                                    <1>       ; INPUT ->
2567                                    <1>       ;      [u.pgdir] = PHYSICAL (real/flat) ADDRESS of the parent's
2568                                    <1>       ;                  page directory.
2569                                    <1>       ; OUTPUT ->
2570                                    <1>       ;      EAX =  PHYSICAL (real/flat) ADDRESS of the child's
2571                                    <1>       ;             page directory.
2572                                    <1>       ;      (New page directory with new page table entries.)
2573                                    <1>       ;      (New page tables with read only copies of the parent's
2574                                    <1>       ;      pages.)
2575                                    <1>       ;      EAX = 0 -> Error (CF = 1)
```

```
2576                                    <1>       ;
2577                                    <1>       ; Modified Registers -> none (except EAX)
2578                                    <1>       ;
2579 0000536A E80AF8FFFF                <1>       call   allocate_page
2580 0000536F 723E                      <1>       jc     short cpd_err
2581                                    <1>       ;
2582 00005371 55                        <1>       push   ebp ; 20/07/2015
2583 00005372 56                        <1>       push   esi
2584 00005373 57                        <1>       push   edi
2585 00005374 53                        <1>       push   ebx
2586 00005375 51                        <1>       push   ecx
2587 00005376 8B35[B8030300]            <1>       mov    esi, [u.pgdir]
2588 0000537C 89C7                      <1>       mov    edi, eax
2589 0000537E 50                        <1>       push   eax ; save child's page directory address
2590                                    <1>       ; copy PDE 0 from the parent's page dir to the child's page dir
2591                                    <1>       ; (use same system space for all user page tables)
2592 0000537F A5                        <1>       movsd
2593 00005380 BD00004000                <1>       mov    ebp, 1024*4096 ; pass the 1st 4MB (system space)
2594 00005385 B9FF030000                <1>       mov    ecx, (PAGE_SIZE / 4) - 1 ; 1023
2595                                    <1> cpd_0:
2596 0000538A AD                        <1>       lodsd
2597                                    <1>       ;or    eax, eax
2598                                    <1>         ;jnz     short cpd_1
2599 0000538B A801                      <1>       test   al, PDE_A_PRESENT ;  bit 0 =  1
2600 0000538D 7508                      <1>       jnz    short cpd_1
2601                                    <1>       ; (virtual address at the end of the page table)
2602 0000538F 81C500004000              <1>       add    ebp, 1024*4096 ; page size * PTE count
2603 00005395 EB0F                      <1>       jmp    short cpd_2
2604                                    <1> cpd_1:
2605 00005397 662500F0                  <1>       and    ax, PDE_A_CLEAR ; 0F000h ; clear attribute bits
2606 0000539B 89C3                      <1>       mov    ebx, eax
2607                                    <1>       ; EBX = Parent's page table address
2608 0000539D E81F000000                <1>       call   copy_page_table
2609 000053A2 720C                      <1>       jc     short cpd_p_err
2610                                    <1>       ; EAX = Child's page table address
2611 000053A4 0C07                      <1>       or     al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
2612                                    <1>                       ; set bit 0, bit 1 and bit 2 to 1
2613                                    <1>                       ; (present, writable, user)
2614                                    <1> cpd_2:
2615 000053A6 AB                        <1>       stosd
2616 000053A7 E2E1                      <1>       loop   cpd_0
2617                                    <1>       ;
2618 000053A9 58                        <1>       pop    eax  ; restore child's page directory address
2619                                    <1> cpd_3:
2620 000053AA 59                        <1>       pop    ecx
2621 000053AB 5B                        <1>       pop    ebx
2622 000053AC 5F                        <1>       pop    edi
2623 000053AD 5E                        <1>       pop    esi
2624 000053AE 5D                        <1>       pop    ebp
2625                                    <1> cpd_err:
2626 000053AF C3                        <1>       retn
2627                                    <1> cpd_p_err:
2628                                    <1>       ; release the allocated pages missing (recover free space)
2629 000053B0 58                        <1>       pop    eax  ; the new page directory address (physical)
2630 000053B1 8B1D[B8030300]            <1>       mov    ebx, [u.pgdir] ; parent's page directory address
2631 000053B7 E8F6F8FFFF                <1>       call   deallocate_page_dir
2632 000053BC 29C0                      <1>       sub    eax, eax ; 0
2633 000053BE F9                        <1>       stc
2634 000053BF EBE9                      <1>       jmp    short cpd_3
2635                                    <1>
2636                                    <1> copy_page_table:
2637                                    <1>       ; 19/09/2015
2638                                    <1>       ; temporary - 07/09/2015
2639                                    <1>       ; 07/09/2015 (Retro UNIX 386 v1 - beginning)
2640                                    <1>       ;
2641                                    <1>       ; INPUT ->
2642                                    <1>       ;      EBX = PHYSICAL (real/flat) ADDRESS of the parent's page table.
2643                                    <1>       ;      EBP = page table entry index (from 'copy_page_dir')
2644                                    <1>       ; OUTPUT ->
2645                                    <1>       ;      EAX = PHYSICAL (real/flat) ADDRESS of the child's page table.
2646                                    <1>       ;      EBP = (recent) page table index (for 'add_to_swap_queue')
2647                                    <1>       ;      CF = 1 -> error
2648                                    <1>       ;
2649                                    <1>       ; Modified Registers -> EBP (except EAX)
2650                                    <1>       ;
2651 000053C1 E8B3F7FFFF                <1>       call   allocate_page
2652 000053C6 725A                      <1>       jc     short cpt_err
2653                                    <1>       ;
2654 000053C8 50                        <1>       push   eax ; *
2655                                    <1>       ;push  ebx
2656 000053C9 56                        <1>       push   esi
2657 000053CA 57                        <1>       push   edi
2658 000053CB 52                        <1>       push   edx
2659 000053CC 51                        <1>       push   ecx
2660                                    <1>       ;
2661 000053CD 89DE                      <1>       mov    esi, ebx
2662 000053CF 89C7                      <1>       mov    edi, eax
2663 000053D1 89C2                      <1>       mov    edx, eax
2664 000053D3 81C200100000              <1>       add    edx, PAGE_SIZE
2665                                    <1> cpt_0:
2666 000053D9 AD                        <1>       lodsd
2667 000053DA A801                      <1>       test   al, PTE_A_PRESENT ;  bit 0 =  1
2668 000053DC 750B                      <1>       jnz    short cpt_1
2669 000053DE 21C0                      <1>       and    eax, eax
2670 000053E0 7430                      <1>       jz     short cpt_2
2671                                    <1>       ; ebp = virtual (linear) address of the memory page
2672 000053E2 E85DFFFFFF                <1>       call   reload_page ; 28/04/2015
2673 000053E7 7234                      <1>       jc     short cpt_p_err
2674                                    <1> cpt_1:
2675 000053E9 662500F0                  <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
2676 000053ED 89C1                      <1>       mov    ecx, eax
2677                                    <1>       ; Allocate a new page for the child process
```

```
2678 000053EF E885F7FFFF          <1>       call   allocate_page
2679 000053F4 7227                <1>       jc     short cpt_p_err
2680 000053F6 57                  <1>       push   edi
2681 000053F7 56                  <1>       push   esi
2682 000053F8 89CE                <1>       mov    esi, ecx
2683 000053FA 89C7                <1>       mov    edi, eax
2684 000053FC B900040000          <1>       mov    ecx, PAGE_SIZE/4
2685 00005401 F3A5                <1>       rep    movsd  ; copy page (4096 bytes)
2686 00005403 5E                  <1>       pop    esi
2687 00005404 5F                  <1>       pop    edi
2688                              <1>       ;
2689 00005405 53                  <1>       push   ebx
2690 00005406 50                  <1>       push   eax
2691 00005407 89EB                <1>       mov    ebx, ebp
2692                              <1>       ; ebx = virtual address of the memory page
2693 00005409 E8AAFDFFFF          <1>       call   add_to_swap_queue
2694 0000540E 58                  <1>       pop    eax
2695 0000540F 5B                  <1>       pop    ebx
2696                              <1>       ;
2697                              <1>       ;or    ax, PTE_A_USER+PTE_A_PRESENT
2698 00005410 0C07                <1>       or     al, PTE_A_USER+PTE_A_WRITE+PTE_A_PRESENT
2699                              <1> cpt_2:
2700 00005412 AB                  <1>       stosd  ; EDI points to child's PTE
2701                              <1>       ;
2702 00005413 81C500100000        <1>       add    ebp, 4096 ; 20/07/2015 (next page)
2703                              <1>       ;
2704 00005419 39D7                <1>       cmp    edi, edx
2705 0000541B 72BC                <1>       jb     short cpt_0
2706                              <1> cpt_p_err:
2707 0000541D 59                  <1>       pop    ecx
2708 0000541E 5A                  <1>       pop    edx
2709 0000541F 5F                  <1>       pop    edi
2710 00005420 5E                  <1>       pop    esi
2711                              <1>       ;pop   ebx
2712 00005421 58                  <1>       pop    eax ; *
2713                              <1> cpt_err:
2714 00005422 C3                  <1>       retn
2715                              <1>
2716                              <1> allocate_memory_block:
2717                              <1>       ; 01/05/2017
2718                              <1>       ; 28/04/2017
2719                              <1>       ; 25/04/2017
2720                              <1>       ; 01/04/2016, 02/04/2016, 03/04/2016
2721                              <1>       ; 13/03/2016, 14/03/2016
2722                              <1>       ; 12/03/2016 (TRDOS 386 = TRDOS v2.0)
2723                              <1>       ; Allocating contiguous memory pages (in the kernel's memory space)
2724                              <1>       ;
2725                              <1>       ; INPUT ->
2726                              <1>       ;     EAX = Beginning address (physical)
2727                              <1>       ;     EAX = 0 -> Allocate memory block from the first proper aperture
2728                              <1>       ;     ECX = Number of bytes to be allocated
2729                              <1>       ;
2730                              <1>       ; OUTPUT ->
2731                              <1>       ;     1) cf = 0 -> successful
2732                              <1>       ;     EAX = Beginning (physical) address of the allocated memory block
2733                              <1>       ;     ECX = Number of allocated bytes (rounded up to page borders)
2734                              <1>       ;     2) cf = 1 -> unsuccessful
2735                              <1>       ;      2.1) If EAX > 0 ->
2736                              <1>       ;           (Number of requested pages is more than # of free pages
2737                              <1>       ;            but contiguous free pages -the aperture- is not enough!)
2738                              <1>       ;           EAX = Beginning address of available aperture
2739                              <1>       ;               (one of all aperture with max. aperture size/length)
2740                              <1>       ;           ECX = Size of available aperture (memory block) in bytes
2741                              <1>       ;      2.2) If EAX = 0 -> Out of memory error
2742                              <1>       ;                 (number of free pages is less than requested number)
2743                              <1>       ;           ECX = Total number of free bytes (free pages * 4096)
2744                              <1>       ;               (It is not number of contiguous free bytes)
2745                              <1>       ;
2746                              <1>       ; (Modified Registers -> EAX, ECX)
2747                              <1>       ;
2748                              <1>       ; PURPOSE: Loading a file at memory for copying or running etc.
2749                              <1>       ; If this procedure returns with cf is set, ECX contains maximum
2750                              <1>       ; available space and EAX contains the beginning address of it.
2751                              <1>       ; If EAX has zero, ECX contains total number of free bytes.
2752                              <1>       ; If requested block has been successfully allocated (by rounding up to
2753                              <1>       ; the last page border), it must be deallocated later by using
2754                              <1>       ; 'deallocate_memory_block' procedure.
2755                              <1>
2756 00005423 52                  <1>       push   edx ; *
2757 00005424 BAFF0F0000          <1>       mov    edx, PAGE_SIZE - 1   ; 4095
2758 00005429 01D0                <1>       add    eax, edx
2759 0000542B 01D1                <1>       add    ecx, edx
2760 0000542D C1E90C              <1>       shr    ecx, PAGE_SHIFT          ; 12
2761                              <1>
2762                              <1>       ; ECX = number of contiguous pages to be allocated
2763 00005430 8B15[08520100]      <1>       mov    edx, [free_pages]
2764                              <1>       ; 01/05/2017
2765                              <1>       ;or    ecx, ecx
2766                              <1>       ;jz    short amb3
2767                              <1>       ; If ECX=0, set cf to 1 and return with max. available mem block size
2768                              <1>
2769 00005436 39D1                <1>       cmp    ecx, edx
2770 00005438 7760                <1>       ja     short amb_3
2771                              <1>
2772 0000543A C1E80C              <1>       shr    eax, PAGE_SHIFT      ; 12
2773                              <1>
2774 0000543D 89C2                <1>       mov    edx, eax          ; page number
2775 0000543F C1EA03              <1>       shr    edx, 3            ; to get offset to M.A.T.
2776                              <1>                                ; (1 allocation bit = 1 page)
2777                              <1>                                ; (1 allocation bytes = 8 pages)
2778 00005442 80E2FC              <1>       and    dl, 0FCh          ; clear lower 2 bits
2779                              <1>                                ; (to get 32 bit position)
```

```
2780 00005445 53                <1>       push   ebx ; **
2781                            <1> amb_0:
2782 00005446 890D[C05E0100]    <1>       mov    [mem_ipg_count], ecx ; initial (reset) value of page count
2783 0000544C 890D[C45E0100]    <1>       mov    [mem_pg_count], ecx
2784 00005452 31C9              <1>       xor    ecx, ecx ; 0
2785 00005454 890D[C85E0100]    <1>       mov    [mem_aperture], ecx ; 0
2786 0000545A 890D[CC5E0100]    <1>       mov    [mem_max_aperture], ecx ; 0
2787                            <1>
2788 00005460 BB00001000        <1>       mov    ebx, MEM_ALLOC_TBL   ; Memory Allocation Table address.
2789 00005465 3B15[0C520100]    <1>       cmp    edx, [next_page]     ; Is the beginning page address lower
2790                            <1>                                   ; than the address in 'next_page' ?
2791                            <1>                                   ; (the first/next free page of user space)
2792 0000546B 7208              <1>       jb     short amb_1
2793 0000546D 3B15[10520100]    <1>       cmp    edx, [last_page]     ; is the beginning page address higher
2794                            <1>                                   ; than the address in 'last_page' ?
2795                            <1>                                   ; (end of the memory)
2796 00005473 7606              <1>       jna    short amb_2          ; no
2797                            <1> amb_1:
2798 00005475 8B15[0C520100]    <1>       mov    edx, [next_page]     ; M.A.T. offset (1 M.A.T. byte = 8 pages)
2799                            <1> amb_2:
2800 0000547B 01D3              <1>       add    ebx, edx
2801                            <1>
2802                            <1>       ; 28/04/2017
2803                            <1>       ;xor   ecx, ecx
2804 0000547D 0FBC0B            <1>       bsf    ecx, [ebx]           ; 0 to 31
2805 00005480 89D0              <1>       mov    eax, edx
2806 00005482 C1E003            <1>       shl    eax, 3               ; *8
2807 00005485 01C8              <1>       add    eax, ecx             ; beginning page number
2808                            <1>
2809 00005487 A3[D05E0100]      <1>       mov    [mem_pg_pos], eax    ; beginning page no (for curr. mem. aperture)
2810 0000548C A3[D45E0100]      <1>       mov    [mem_max_pg_pos], eax ; beginning page no for max. mem. aperture
2811                            <1>
2812 00005491 83E01F            <1>       and    eax, 1Fh             ; lower 5 bits only (0 to 31)
2813                            <1>                                   ; (allocation bit position)
2814 00005494 750E              <1>       jnz    short amb_4          ; 0
2815 00005496 B120              <1>       mov    cl, 32
2816 00005498 EB4B              <1>       jmp    short amb_10
2817                            <1>
2818                            <1> amb_3:       ; out_of_memory
2819 0000549A 31C0              <1>       xor    eax, eax ; 0
2820 0000549C 89D1              <1>       mov    ecx, edx ; free pages
2821 0000549E C1E10C            <1>       shl    ecx, PAGE_SHIFT
2822 000054A1 5A                <1>       pop    edx ; *
2823 000054A2 F9                <1>       stc
2824 000054A3 C3                <1>       retn
2825                            <1> amb_4:
2826 000054A4 8B13              <1>       mov    edx, [ebx]
2827 000054A6 88C1              <1>       mov    cl, al ; 1 to 31
2828 000054A8 D3EA              <1>       shr    edx, cl
2829 000054AA 89D0              <1>       mov    eax, edx
2830                            <1> amb_5:
2831 000054AC D1E8              <1>       shr    eax, 1 ; (***)
2832 000054AE 7317              <1>       jnc    short amb_7
2833 000054B0 FF05[C85E0100]    <1>       inc    dword [mem_aperture]
2834 000054B6 FF0D[C45E0100]    <1>       dec    dword [mem_pg_count]
2835 000054BC 7470              <1>       jz     short amb_15
2836                            <1> amb_6:
2837                            <1>       ; 28/04/2017
2838 000054BE FEC1              <1>       inc    cl
2839 000054C0 80F920            <1>       cmp    cl, 32
2840 000054C3 730D              <1>       jnb    short amb_9
2841 000054C5 EBE5              <1>       jmp    short amb_5
2842                            <1> amb_7:
2843 000054C7 50                <1>       push   eax ; (***) allocation bits (in shifted status)
2844 000054C8 E81B010000        <1>       call   amb_26 ; set maximum memory aperture (free memory block size)
2845 000054CD 58                <1>       pop    eax ; (***)
2846 000054CE EBEE              <1>       jmp    short amb_6
2847                            <1> amb_8:
2848                            <1>       ; 28/04/2017
2849 000054D0 B120              <1>       mov    cl, 32
2850                            <1> amb_9:
2851 000054D2 89DA              <1>       mov    edx, ebx
2852 000054D4 81EA00001000      <1>       sub    edx, MEM_ALLOC_TBL
2853 000054DA 3B15[10520100]    <1>       cmp    edx, [last_page]
2854 000054E0 7336              <1>       jnb    short amb_14 ; contiguous pages not enough
2855 000054E2 83C304            <1>       add    ebx, 4
2856                            <1> amb_10:
2857 000054E5 8B03              <1>       mov    eax, [ebx]
2858 000054E7 21C0              <1>       and    eax, eax
2859 000054E9 7408              <1>         jz     short amb_11 ; there is not a free page bit in this alloc dword
2860 000054EB 40                <1>       inc    eax ; 0FFFFFFFFh -> 0
2861 000054EC 740C              <1>       jz     short amb_12 ; all of bits are set (32 free pages)
2862 000054EE 48                <1>       dec    eax
2863 000054EF 28C9              <1>       sub    cl, cl ; 0
2864 000054F1 EBB9              <1>       jmp    short amb_5
2865                            <1> amb_11:
2866 000054F3 E8F0000000        <1>       call   amb_26 ; set maximum memory aperture (free memory block size)
2867 000054F8 EBD8              <1>       jmp    short amb_9
2868                            <1> amb_12:
2869 000054FA 390D[C45E0100]    <1>       cmp    [mem_pg_count], ecx ; 32
2870 00005500 7306              <1>       jnb    short amb_13
2871 00005502 8B0D[C45E0100]    <1>       mov    ecx, [mem_pg_count]
2872                            <1> amb_13:
2873 00005508 010D[C85E0100]    <1>       add    [mem_aperture], ecx
2874 0000550E 290D[C45E0100]    <1>       sub    [mem_pg_count], ecx
2875 00005514 7618              <1>       jna    short amb_15
2876 00005516 EBBA              <1>       jmp    short amb_9 ; 01/05/2017
2877                            <1> amb_14:
2878 00005518 E8CB000000        <1>       call   amb_26 ; 28/04/2017
2879 0000551D A1[D45E0100]      <1>       mov    eax, [mem_max_pg_pos] ; begin address of max. mem aperture
2880 00005522 8B0D[CC5E0100]    <1>       mov    ecx, [mem_max_aperture] ; max. (largest) memory aperture
2881 00005528 F9                <1>       stc
```

```
2882 00005529 E9AF000000          <1>         jmp     amb_25
2883                              <1>
2884                              <1> amb_15: ; OK !
2885 0000552E A1[D05E0100]        <1>         mov     eax, [mem_pg_pos]    ; Beginning address as page number
2886 00005533 8B0D[C85E0100]      <1>         mov     ecx, [mem_aperture]  ; Free contiguous page count (>=1)
2887                              <1> amb_16:
2888                              <1>         ; allocate contiguous memory pages (via memory allocation table bits)
2889 00005539 89C2                <1>         mov     edx, eax
2890                              <1>         ; 25/04/2017
2891 0000553B C1EA03              <1>         shr     edx, 3          ; 8 pages in one allocation byte
2892 0000553E 80E2FC              <1>         and     dl, 0FCh        ; clear lower 2 bits
2893                              <1>                                 ; (for dword/32bit positioning)
2894                              <1>
2895 00005541 BB00001000          <1>         mov     ebx, MEM_ALLOC_TBL
2896 00005546 01D3                <1>         add     ebx, edx
2897 00005548 83E01F              <1>         and     eax, 1Fh ; 31
2898                              <1>         ; 03/04/2016
2899 0000554B BA20000000          <1>         mov     edx, 32
2900 00005550 28C2                <1>         sub     dl, al
2901 00005552 39CA                <1>         cmp     edx, ecx        ; ecx >= 1
2902 00005554 7602                <1>         jna     short amb_17
2903 00005556 89CA                <1>         mov     edx, ecx
2904                              <1> amb_17:
2905 00005558 29D1                <1>         sub     ecx, edx
2906 0000555A 51                  <1>         push    ecx ; ***
2907 0000555B 89D1                <1>         mov     ecx, edx
2908                              <1> amb_18:
2909 0000555D 0FB303              <1>         btr     [ebx], eax      ; The destination bit indexed by the source value
2910                              <1>                                 ; is copied into the Carry Flag and then cleared
2911                              <1>                                 ; in the destination.
2912 00005560 FF0D[08520100]      <1>         dec     dword [free_pages] ; 1 page has been allocated (X = X-1)
2913 00005566 49                  <1>         dec     ecx
2914 00005567 7404                <1>         jz      short amb_19
2915 00005569 FEC0                <1>         inc     al
2916 0000556B EBF0                <1>         jmp     short amb_18
2917                              <1> amb_19:
2918 0000556D 59                  <1>         pop     ecx ; ***
2919 0000556E 21C9                <1>         and     ecx, ecx ; 0 ?
2920 00005570 741E                <1>         jz      short amb_22
2921                              <1>         ; 01/04/2016
2922 00005572 B020                <1>         mov     al, 32
2923                              <1> amb_20:
2924 00005574 83C304              <1>         add     ebx, 4
2925 00005577 39C1                <1>         cmp     ecx, eax ; 32
2926 00005579 7305                <1>         jnb     short amb_21
2927                              <1>         ; ECX < 32
2928 0000557B 28C0                <1>         sub     al, al ; 0
2929 0000557D 50                  <1>         push    eax ; 0 ***
2930 0000557E EBDD                <1>         jmp     short amb_18
2931                              <1> amb_21:
2932 00005580 2905[08520100]      <1>         sub     [free_pages], eax   ; [free_pages] = [free_pages] - 32
2933 00005586 C70300000000        <1>         mov     dword [ebx], 0          ; reset 32 bits
2934 0000558C 29C1                <1>         sub     ecx, eax ; 32
2935 0000558E 75E4                <1>         jnz     short amb_20
2936                              <1> amb_22:
2937 00005590 A1[D05E0100]        <1>         mov     eax, [mem_pg_pos]   ; Beginning address as page number
2938 00005595 8B0D[C85E0100]      <1>         mov     ecx, [mem_aperture] ; Free contiguous page count
2939                              <1>         ; [next_page] update
2940 0000559B 89C2                <1>         mov     edx, eax
2941                              <1>         ; 03/04/2016
2942 0000559D C1EA03              <1>         shr     edx, 3            ; to get offset to M.A.T.
2943                              <1>                                   ; (1 allocation bit = 1 page)
2944                              <1>                                   ; (1 allocation bytes = 8 pages)
2945 000055A0 80E2FC              <1>         and     dl, 0FCh          ; clear lower 2 bits
2946                              <1>                                   ; (to get 32 bit position)
2947 000055A3 3B15[0C520100]      <1>         cmp     edx, [next_page] ; first free page pointer offset
2948 000055A9 7732                <1>         ja      short amb_25
2949 000055AB BB00001000          <1>         mov     ebx, MEM_ALLOC_TBL
2950 000055B0 833C1300            <1>         cmp     dword [ebx+edx], 0
2951 000055B4 7721                <1>         ja      short amb_24
2952 000055B6 89C2                <1>         mov     edx, eax
2953 000055B8 01CA                <1>         add     edx, ecx
2954 000055BA C1EA03              <1>         shr     edx, 3
2955 000055BD 80E2FC              <1>         and     dl, 0FCh
2956                              <1> amb_23:
2957 000055C0 833C1300            <1>         cmp     dword [ebx+edx], 0
2958 000055C4 7711                <1>         ja      short amb_24
2959 000055C6 83C204              <1>         add     edx, 4
2960 000055C9 3B15[10520100]      <1>         cmp     edx, [last_page]    ; last page pointer offset
2961 000055CF 76EF                <1>         jna     short amb_23
2962 000055D1 8B15[14520100]      <1>         mov     edx, [first_page]   ; (for) beginning of user's space
2963                              <1> amb_24:
2964 000055D7 8915[0C520100]      <1>         mov     [next_page], edx
2965                              <1> amb_25:
2966 000055DD 9C                  <1>         pushf
2967 000055DE C1E00C              <1>         shl     eax, PAGE_SHIFT         ; convert to phy. address in bytes
2968 000055E1 C1E10C              <1>         shl     ecx, PAGE_SHIFT         ; convert to byte counts
2969 000055E4 9D                  <1>         popf
2970 000055E5 5B                  <1>         pop     ebx ; **
2971 000055E6 5A                  <1>         pop     edx ; *
2972 000055E7 C3                  <1>         retn
2973                              <1>
2974                              <1> amb_26:     ; set maximum free memory aperture (free memory block size)
2975 000055E8 89DA                <1>         mov     edx, ebx ; current address
2976 000055EA 81EA00001000        <1>         sub     edx, MEM_ALLOC_TBL ; MAT beginning address
2977                              <1>         ; 02/04/2016
2978 000055F0 C1E203              <1>         shl     edx, 3 ; MAT byte offset * 8 = page number base
2979 000055F3 01CA                <1>         add     edx, ecx ; current page number (ecx = 0 to 32)
2980                              <1>         ;
2981 000055F5 A1[C85E0100]        <1>         mov     eax, [mem_aperture]
2982 000055FA 21C0                <1>         and     eax, eax
2983 000055FC 7421                <1>           jz      short amb_27
```

```
2984 000055FE C705[C85E0100]0000- <1>         mov     dword [mem_aperture], 0
2984 00005606 0000                <1>
2985 00005608 3B05[CC5E0100]      <1>       cmp    eax, [mem_max_aperture]
2986 0000560E 760F                <1>       jna    short amb_27
2987 00005610 A3[CC5E0100]        <1>       mov    [mem_max_aperture], eax
2988                              <1>       ; 25/04/2017
2989 00005615 A1[D05E0100]        <1>       mov    eax, [mem_pg_pos]
2990                              <1>       ; EAX = Beginning page number of the max. aperture
2991 0000561A A3[D45E0100]        <1>       mov    [mem_max_pg_pos], eax
2992                              <1> amb_27:
2993 0000561F 8915[D05E0100]      <1>       mov    [mem_pg_pos], edx ; current page
2994                              <1>
2995 00005625 A1[C05E0100]        <1>       mov    eax, [mem_ipg_count] ; initial (reset) value of page count
2996 0000562A A3[C45E0100]        <1>       mov    [mem_pg_count], eax
2997                              <1>
2998 0000562F C3                  <1>       retn
2999                              <1>
3000                              <1> deallocate_memory_block:
3001                              <1>       ; 03/04/2016
3002                              <1>       ; 14/03/2016 (TRDOS 386 = TRDOS v2.0)
3003                              <1>       ; Deallocating contiguous memory pages (in the kernel's memory space)
3004                              <1>       ;
3005                              <1>       ; INPUT ->
3006                              <1>       ;     EAX = Beginning address (physical)
3007                              <1>       ;     ECX = Number of bytes to be deallocated
3008                              <1>       ;
3009                              <1>       ; OUTPUT ->
3010                              <1>       ;     Memory Allocation Table bits will be updated
3011                              <1>       ;     [free_pages] will be changed (increased)
3012                              <1>       ;
3013                              <1>       ; (Modified Registers -> EAX, ECX)
3014                              <1>       ;
3015                              <1>       ; PURPOSE: Unloading/Freeing a file -or an allocated memory block-
3016                              <1>       ; at memory after copying, running, saving, reading, writing etc.
3017                              <1>       ;
3018                              <1>
3019 00005630 52                  <1>       push   edx ; *
3020 00005631 53                  <1>       push   ebx ; **
3021                              <1>
3022 00005632 C1E80C              <1>       shr    eax, PAGE_SHIFT          ; 12
3023 00005635 C1E90C              <1>       shr    ecx, PAGE_SHIFT          ; 12
3024                              <1>
3025                              <1>       ; EAX = Beginning page number
3026                              <1>       ; ECX = Number of contiguous pages to be deallocated
3027                              <1> damb_0:
3028                              <1>       ; deallocate contiguous memory pages (via memory allocation table bits)
3029 00005638 89C2                <1>       mov    edx, eax
3030 0000563A C1EA03              <1>       shr    edx, 3            ; to get offset to M.A.T.
3031                              <1>                                ; (1 allocation bit = 1 page)
3032                              <1>                                ; (1 allocation bytes = 8 pages)
3033 0000563D 80E2FC              <1>       and    dl, 0FCh          ; clear lower 2 bits
3034                              <1>                                ; (to get 32 bit position)
3035 00005640 3B15[0C520100]      <1>       cmp    edx, [next_page] ; next free page
3036 00005646 7306                <1>       jnb    short damb_1
3037 00005648 8915[0C520100]      <1>       mov    [next_page], edx
3038                              <1> damb_1:
3039 0000564E BB00001000          <1>       mov    ebx, MEM_ALLOC_TBL
3040 00005653 01D3                <1>       add    ebx, edx
3041 00005655 83E01F              <1>       and    eax, 1Fh ; 31
3042                              <1>
3043                              <1>       ; 03/04/2016
3044 00005658 BA20000000          <1>       mov    edx, 32
3045 0000565D 28C2                <1>       sub    dl, al
3046 0000565F 39CA                <1>       cmp    edx, ecx
3047 00005661 7602                <1>       jna    short damb_2
3048 00005663 89CA                <1>       mov    edx, ecx
3049                              <1> damb_2:
3050 00005665 29D1                <1>       sub    ecx, edx
3051 00005667 51                  <1>       push   ecx ; ***
3052 00005668 89D1                <1>       mov    ecx, edx
3053                              <1> damb_3:
3054 0000566A 0FAB03              <1>       bts    [ebx], eax        ; unlink/release/deallocate page
3055                              <1>                                ; set relevant bit to 1.
3056                              <1>                                ; set CF to the previous bit value
3057 0000566D FF05[08520100]      <1>       inc     dword [free_pages]   ; 1 page has been deallocated (X = X+1)
3058 00005673 49                  <1>       dec    ecx
3059 00005674 7404                <1>       jz     short damb_4
3060 00005676 FEC0                <1>       inc    al
3061 00005678 EBF0                <1>       jmp    short damb_3
3062                              <1> damb_4:
3063 0000567A 59                  <1>       pop    ecx ; ***
3064 0000567B 21C9                <1>       and    ecx, ecx ; 0 ?
3065 0000567D 741E                <1>       jz     short damb_7
3066                              <1>       ; 03/04/2016
3067 0000567F B020                <1>       mov    al, 32
3068                              <1> damb_5:
3069 00005681 83C304              <1>       add    ebx, 4
3070 00005684 39C1                <1>       cmp    ecx, eax ; 32
3071 00005686 7305                <1>       jnb    short damb_6
3072                              <1>       ; ECX < 32
3073 00005688 28C0                <1>       sub    al, al ; 0
3074 0000568A 50                  <1>       push   eax ; 0 ***
3075 0000568B EBDD                <1>       jmp    short damb_3
3076                              <1> damb_6:
3077 0000568D 0105[08520100]      <1>       add    [free_pages], eax ; [free_pages] = [free_pages] + 32
3078 00005693 C703FFFFFFFF        <1>       mov    dword [ebx], 0FFFFFFFFh ; set 32 bits
3079 00005699 29C1                <1>       sub    ecx, eax ; 32
3080 0000569B 75E4                <1>       jnz    short damb_5
3081                              <1> damb_7:
3082 0000569D 5B                  <1>       pop    ebx ; **
3083 0000569E 5A                  <1>       pop    edx ; *
3084 0000569F 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
000056A0 662500F0                       <1>       and    ax, PTE_A_CLEAR ; clear page offset
000056A4 50                             <1>       push   eax
                                        <1>       ;and   ecx, ecx ; page count
                                        <1>       ;jz    dmem_acc_7  ; 'insufficient memory' error
000056A5 89C5                           <1>       mov    ebp, eax
000056A7 81C300004000                   <1>       add    ebx, CORE ; 12/05/2017
                                        <1> dmem_acc_0:
000056AD 891D[C0690100]                 <1>       mov    [base_addr], ebx ; 12/05/2017
000056B3 A1[B8030300]                   <1>       mov    eax, [u.pgdir] ; page dir address (physical)
000056B8 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 ?
000056BD 7324                           <1>       jnc    short dmem_acc_1
                                        <1>       ;
000056BF E8B5F4FFFF                     <1>       call   allocate_page
000056C4 0F82AB000000                   <1>         jc     dmem_acc_7  ; 'insufficient memory' error
                                        <1>       ;
000056CA E824F5FFFF                     <1>       call   clear_page
                                        <1>       ; EAX = Physical (base) address of the allocated (new) page
000056CF 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)
000056D1 8902                           <1>       mov    [edx], eax ; Let's put the new page directory entry here !
000056D3 A1[B8030300]                   <1>       mov    eax, [u.pgdir]
000056D8 E8B7F5FFFF                     <1>       call   get_pte
000056DD 0F8292000000                   <1>         jc     dmem_acc_7 ; 'insufficient memory' error
                                        <1> dmem_acc_1:
                                        <1>       ; EAX = PTE value, EDX = PTE address
000056E3 A801                           <1>       test   al, PTE_A_PRESENT
000056E5 750D                           <1>       jnz    short dmem_acc_2
000056E7 09C0                           <1>       or     eax, eax
000056E9 7468                           <1>       jz     short dmem_acc_6   ; Change PTE
000056EB D1E8                           <1>       shr    eax, 1       ; swap disk block (8 sectors) address
                                        <1>       ; unlink swap disk block
000056ED E80CFBFFFF                     <1>       call   unlink_swap_block
000056F2 EB5F                           <1>       jmp    short dmem_acc_6
                                        <1>
                                        <1> dmem_acc_2:
000056F4 A802                           <1>       test   al, PTE_A_WRITE   ; bit 1, writable (r/w) flag
                                        <1>                                ; (must be 1)
000056F6 7550                           <1>       jnz    short dmem_acc_4
                                        <1>       ; Read only -duplicated- page (belongs to a parent or a child)
000056F8 66A90002                       <1>         test    ax, PTE_DUPLICATED ; Was this page duplicated
                                        <1>                                  ; as child's page ?
000056FC 7455                           <1>       jz     short dmem_acc_5 ; Change PTE but don't deallocate the page!
                                        <1>
                                        <1>       ;push  edi
                                        <1>       ;push  esi
                                        <1>
000056FE 51                             <1>       push   ecx
                                        <1>       ;push  ebx
```

```
3187 000056FF 8B1D[BC030300]      <1>       mov    ebx, [u.ppgdir] ; parent's page dir address (physical)
3188                              <1>
3189                              <1>       ; check the parent's PTE value is read only & same page or not..
3190 00005705 89EF                <1>       mov    edi, ebp
3191 00005707 C1EF16              <1>       shr    edi, PAGE_D_SHIFT ; 22
3192                              <1>       ; EDI = page directory entry index (0-1023)
3193 0000570A 89EE                <1>       mov    esi, ebp
3194 0000570C C1EE0C              <1>       shr    esi, PAGE_SHIFT ; 12
3195 0000570F 81E6FF030000        <1>       and    esi, PTE_MASK
3196                              <1>       ; ESI = page table entry index (0-1023)
3197                              <1>
3198 00005715 66C1E702            <1>       shl    di, 2 ; * 4
3199 00005719 01FB                <1>       add    ebx, edi ; PDE offset (for the parent)
3200 0000571B 8B0F                <1>       mov    ecx, [edi]
3201 0000571D F6C101              <1>       test   cl, PDE_A_PRESENT ; present (valid) or not ?
3202 00005720 7425                <1>       jz     short dmem_acc_3    ; parent process does not use this page
3203 00005722 6681E100F0          <1>       and    cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
3204 00005727 66C1E602            <1>       shl    si, 2 ; *4
3205 0000572B 01CE                <1>       add    esi, ecx ; PTE offset (for the parent)
3206 0000572D 8B1E                <1>       mov    ebx, [esi]
3207 0000572F F6C301              <1>       test   bl, PTE_A_PRESENT ; present or not ?
3208 00005732 7413                <1>       jz     short dmem_acc_3    ; parent process does not use this page
3209 00005734 662500F0            <1>       and    ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3210 00005738 6681E300F0          <1>       and    bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3211 0000573D 39D8                <1>       cmp    eax, ebx     ; parent's and child's pages are same ?
3212 0000573F 7506                <1>       jne    short dmem_acc_3    ; not same page
3213                              <1>                              ; deallocate the child's page
3214 00005741 800E02              <1>         or      byte [esi], PTE_A_WRITE ; convert to writable page (parent)
3215                              <1>       ;pop   ebx
3216 00005744 59                  <1>       pop    ecx
3217 00005745 EB0C                <1>       jmp    short dmem_acc_5
3218                              <1> dmem_acc_3:
3219                              <1>       ;pop   ebx
3220 00005747 59                  <1>       pop    ecx
3221                              <1> dmem_acc_4:
3222 00005748 66A90004            <1>       test   ax, PTE_SHARED ; shared or direct memory access indicator
3223 0000574C 7505                <1>       jnz    short dmem_acc_5   ; AVL bit 1 = 1, do not deallocate this page!
3224                              <1>       ;
3225                              <1>       ;and   ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3226 0000574E E804F6FFFF          <1>       call   deallocate_page
3227                              <1> dmem_acc_5:
3228                              <1>       ;pop   esi
3229                              <1>       ;pop   edi
3230                              <1> dmem_acc_6:
3231 00005753 89E8                <1>       mov    eax, ebp ; physical page (offset=0) address
3232                              <1>       ; EAX = memory page address
3233                              <1>       ; EDX = PTE entry address (physical)
3234 00005755 660D0704            <1>       or     ax, PTE_A_PRESENT+PTE_A_USER+PTE_A_WRITE+PTE_SHARED
3235                              <1>                     ; present flag, bit 0 = 1
3236                              <1>                     ; user flag, bit 2 = 1
3237                              <1>                     ; writable flag, bit 1 = 1
3238                              <1>                     ; direct memory access flag, bit 10 = 1
3239                              <1>                     ; (This page must not be deallocated!)
3240 00005759 8902                <1>       mov    [edx], eax  ; Update PTE value
3241 0000575B 49                  <1>       dec    ecx ; remain count of contiguous pages
3242 0000575C 741E                <1>       jz     short dmem_acc_8
3243 0000575E 81C500100000        <1>       add    ebp, PAGE_SIZE ; next physical page address
3244                              <1>       ; 22/07/2017
3245                              <1>       ;mov   eax, ebp
3246                              <1>       ; 12/05/2017
3247 00005764 8B1D[C0690100]      <1>       mov    ebx, [base_addr] ; linear address (virtual+CORE)
3248 0000576A 81C300100000        <1>       add    ebx, PAGE_SIZE      ; next linear address
3249 00005770 E938FFFFFF          <1>         jmp     dmem_acc_0
3250                              <1> dmem_acc_7:  ; ERROR !
3251 00005775 C7042404000000      <1>       mov    dword [esp], ERR_MINOR_IM
3252                              <1>              ; Insufficient memory (minor) error!
3253                              <1>              ; Major error = 0 (No protection fault)
3254                              <1>       ; cf = 1
3255                              <1> dmem_acc_8:
3256 0000577C 58                  <1>       pop    eax
3257                              <1>       ;pop   edx
3258                              <1>       ;pop   ecx
3259                              <1>       ;pop   ebx
3260                              <1>       ;pop   ebp
3261 0000577D C3                  <1>       retn
3262                              <1>
3263                              <1> deallocate_user_pages:
3264                              <1>       ; 20/05/2017
3265                              <1>       ; 15/05/2017
3266                              <1>       ; 20/02/2017
3267                              <1>       ; 19/02/2017 (TRDOS 386 = TRDOS v2.0)
3268                              <1>       ;
3269                              <1>       ; Deallocate virtually contiguous user pages (memory block)
3270                              <1>       ; (caller: 'sysdalloc' system call)
3271                              <1>       ;
3272                              <1>       ; INPUT ->
3273                              <1>       ;     EBX = VIRTUAL ADDRESS (beginning address)
3274                              <1>       ;     ECX = byte count
3275                              <1>       ;     [u.pgdir] = user's page directory
3276                              <1>       ;     [u.ppdir] = parent's page directory
3277                              <1>       ;
3278                              <1>       ; OUTPUT ->
3279                              <1>       ;    If CF = 0
3280                              <1>       ;     EAX = Deallocated memory bytes
3281                              <1>       ;       (Even if shared or read only pages will not be
3282                              <1>       ;        deallocated on M.A.T., this byte count will be
3283                              <1>       ;        returned as virtually deallocated bytes; in fact
3284                              <1>       ;        virtually deallocated user pages * 4096.)
3285                              <1>       ;     EBX = Virtual address (as rounded up)
3286                              <1>       ;    If CF = 1
3287                              <1>       ;     EAX = 0 (there is not any deallocated pages)
3288                              <1>       ;
```

```
3289                            <1>       ; Note: Empty page tables will not be deallocated!!!
3290                            <1>       ;     (they will be deallocated at process termination stage)
3291                            <1>       ;
3292                            <1>       ; Modified Registers -> EAX, EDX, ESI, EDI, EBX, ECX, EBP
3293                            <1>       ;
3294 0000577E 89DE             <1>       mov   esi, ebx
3295 00005780 89F7             <1>       mov   edi, esi
3296 00005782 01CF             <1>       add   edi, ecx
3297 00005784 81C6FF0F0000     <1>       add   esi, PAGE_SIZE - 1  ; 4095 (round up)
3298 0000578A C1EE0C           <1>       shr   esi, PAGE_SHIFT
3299 0000578D C1EF0C           <1>       shr   edi, PAGE_SHIFT
3300 00005790 89F8             <1>       mov   eax, edi ; end page
3301 00005792 29F0             <1>       sub   eax, esi ; end page - start page
3302 00005794 0F86D5000000     <1>       jna   da_u_pd_err  ; < 1
3303 0000579A 89F3             <1>       mov   ebx, esi
3304 0000579C C1E30C           <1>       shl   ebx, PAGE_SHIFT ; virtual address (as rounded up)
3305 0000579F 53               <1>       push  ebx ; *
3306 000057A0 89C1             <1>       mov   ecx, eax ; page count
3307 000057A2 C1E00C           <1>       shl   eax, PAGE_SHIFT ; byte count as adjusted
3308 000057A5 50               <1>       push  eax ; **
3309 000057A6 8B1D[B8030300]   <1>       mov   ebx, [u.pgdir] ; physical addr of user's page dir
3310 000057AC 81C600040000     <1>       add   esi, CORE/PAGE_SIZE
3311 000057B2 89F7             <1>       mov   edi, esi
3312 000057B4 81E7FF030000     <1>       and   edi, PTE_MASK ; PTE entry in the page table
3313 000057BA 57               <1>       push  edi ; *** ; PTE index (of page directory)
3314 000057BB C1EE0A           <1>       shr   esi, PAGE_D_SHIFT - PAGE_SHIFT ; 22-12=10
3315 000057BE 89F2             <1>       mov   edx, esi
3316                            <1>       ; EDX = PDE index
3317 000057C0 C1E602           <1>       shl   esi, 2 ; convert PDE index to dword offset
3318 000057C3 01DE             <1>       add   esi, ebx ; add page directory address
3319                            <1> da_u_pd_1:
3320 000057C5 AD               <1>       lodsd
3321                            <1>       ;
3322 000057C6 89F5             <1>       mov   ebp, esi ; 20/02/2017
3323                            <1>       ; EBP = next PDE address
3324                            <1>       ;
3325 000057C8 A801             <1>       test  al, PDE_A_PRESENT ; bit 0, present flag (must be 1)
3326 000057CA 0F8494000000     <1>       jz    da_u_pd_3 ; 20/05/2017
3327 000057D0 662500F0         <1>       and   ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3328                            <1>       ; EAX = PHYSICAL (flat) ADDRESS OF THE PAGE TABLE
3329 000057D4 8B3C24           <1>       mov   edi, [esp] ; ***
3330                            <1>       ; EDI = PTE index (of complete page directory)
3331                            <1>       ;and  edi, PTE_MASK ; PTE entry in the page table
3332 000057D7 C1E702           <1>       shl   edi, 2 ; convert PTE index to dword offset
3333 000057DA 89FE             <1>       mov   esi, edi ; PTE offset in page table (0-4092)
3334 000057DC 01C6             <1>       add   esi, eax ; now, esi points to requested PTE
3335                            <1> da_u_pt_0:
3336 000057DE AD               <1>       lodsd
3337 000057DF A801             <1>       test  al, PTE_A_PRESENT ; bit 0, present flag (must be 1)
3338 000057E1 743F             <1>       jz    short da_u_pt_1
3339                            <1>       ;
3340 000057E3 A802             <1>       test  al, PTE_A_WRITE   ; bit 1, writable (r/w) flag
3341                            <1>                               ; (must be 1)
3342 000057E5 7549             <1>       jnz   short da_u_pt_3
3343                            <1>       ; Read only -duplicated- page (belongs to a parent or a child)
3344 000057E7 66A90002         <1>         test    ax, PTE_DUPLICATED ; Was this page duplicated
3345                            <1>                                    ; as child's page ?
3346 000057EB 744E             <1>       jz    short da_u_pt_4 ; Clear PTE but don't deallocate the page!
3347                            <1>       ;
3348                            <1>       ; check the parent's PTE value is read only & same page or not..
3349                            <1>       ; EDX = page directory entry index (0-1023)
3350 000057ED 52               <1>       push  edx ; ****
3351                            <1>       ; EDI = page table entry offset (0-4092)
3352 000057EE 8B1D[BC030300]   <1>       mov   ebx, [u.ppgdir] ; page directory of the parent process
3353 000057F4 66C1E202         <1>       shl   dx, 2 ; *4
3354 000057F8 01D3             <1>       add   ebx, edx ; PDE address (for the parent)
3355 000057FA 8B13             <1>       mov   edx, [ebx] ; page table address
3356 000057FC F6C201           <1>       test  dl, PDE_A_PRESENT ; present (valid) or not ?
3357 000057FF 742E             <1>       jz    short da_u_pt_2     ; parent process does not use this page
3358 00005801 6681E200F0       <1>       and   dx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
3359                            <1>       ; EDI = page table entry offset (0-4092)
3360 00005806 01D7             <1>       add   edi, edx     ; PTE address (for the parent)
3361 00005808 8B1F             <1>       mov   ebx, [edi]
3362 0000580A F6C301           <1>       test  bl, PTE_A_PRESENT ; present or not ?
3363 0000580D 7420             <1>       jz    short da_u_pt_2     ; parent process does not use this page
3364 0000580F 662500F0         <1>       and   ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3365 00005813 6681E300F0       <1>       and   bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3366 00005818 39D8             <1>       cmp   eax, ebx     ; parent's and child's pages are same ?
3367 0000581A 7513             <1>       jne   short da_u_pt_2     ; not same page
3368                            <1>                                 ; deallocate the child's page
3369 0000581C 800F02           <1>         or      byte [edi], PTE_A_WRITE ; convert to writable page (parent)
3370 0000581F 5A               <1>       pop   edx ; ****
3371 00005820 EB19             <1>       jmp   short da_u_pt_4
3372                            <1> da_u_pt_1:
3373 00005822 09C0             <1>       or    eax, eax     ; swapped page ?
3374 00005824 741C             <1>       jz    short da_u_pt_5     ; no
3375                            <1>                                 ; yes
3376 00005826 D1E8             <1>       shr   eax, 1
3377 00005828 E8D1F9FFFF       <1>       call  unlink_swap_block ; Deallocate swapped page block
3378                            <1>                               ; on the swap disk (or in file)
3379 0000582D EB13             <1>       jmp   short da_u_pt_5
3380                            <1> da_u_pt_2:
3381 0000582F 5A               <1>       pop   edx ; ****
3382                            <1> da_u_pt_3:
3383 00005830 66A90004         <1>       test  ax, PTE_SHARED      ; shared or direct memory access indicator
3384 00005834 7505             <1>       jnz   short da_u_pt_4     ; AVL bit 1 = 1, do not deallocate this page!
3385                            <1>       ;
3386                            <1>       ;and  ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3387 00005836 E81CF5FFFF       <1>       call  deallocate_page ; set the mem allocation bit of this page
3388                            <1> da_u_pt_4:
3389 0000583B C746FC00000000   <1>       mov   dword [esi-4], 0 ; clear/reset PTE (child, dupl. as parent)
3390                            <1> da_u_pt_5:
```

```
3391                                    <1>       ; 20/05/2017
3392 00005842 58                        <1>       pop    eax ; *** PTE index (of page directory)
3393 00005843 49                        <1>       dec    ecx ; remain page count
3394 00005844 7426                      <1>       jz     short da_u_pd_4
3395 00005846 40                        <1>       inc    eax ; next PTE
3396 00005847 6625FF03                  <1>       and    ax, PTE_MASK ; PTE entry index in the page table
3397 0000584B 50                        <1>       push   eax ; *** (save again)
3398                                    <1>       ;mov   edi, eax
3399                                    <1>       ;and   di, PTE_MASK
3400                                    <1>       ;cmp   edi, PAGE_SIZE / 4 ; 1024
3401                                    <1>       ;jnb   short da_u_pd_2
3402 0000584C 89C7                      <1>       mov    edi, eax
3403 0000584E C1E702                    <1>       shl    edi, 2 ; convert index to dword offset
3404                                    <1>       ;test  ax, PTE_MASK ; 3FFh
3405 00005851 09C0                      <1>       or     eax, eax
3406 00005853 7589                      <1>       jnz    short da_u_pt_0 ; 1-1023
3407                                    <1> da_u_pd_2:
3408 00005855 42                        <1>       inc    edx
3409                                    <1>       ; 20/05/2017
3410 00005856 6681E2FF03                <1>       and    dx, PTE_MASK  ; 3FFh
3411 0000585B 740F                      <1>       jz     short da_u_pd_4  ; 0 (1024)
3412                                    <1>       ;cmp   edx, 1024
3413                                    <1>       ;jnb   short da_u_pd_4
3414 0000585D 89EE                      <1>       mov    esi, ebp ; 20/02/2017
3415 0000585F E961FFFFFF                <1>       jmp    da_u_pd_1
3416                                    <1> da_u_pd_3:
3417                                    <1>       ; 15/05/2017 (empty page directory entry)
3418 00005864 81E900040000              <1>       sub    ecx, 1024
3419 0000586A 77E9                      <1>       ja     short da_u_pd_2 ; 20/05/2017
3420                                    <1> da_u_pd_4:
3421 0000586C 58                        <1>       pop    eax ; **
3422 0000586D 5B                        <1>       pop    ebx ; *
3423 0000586E C3                        <1>       retn
3424                                    <1>
3425                                    <1> da_u_pd_err:
3426 0000586F 31C0                      <1>       xor    eax, eax
3427 00005871 F9                        <1>       stc
3428 00005872 C3                        <1>       retn
3429                                    <1>
3430                                    <1> allocate_user_pages:
3431                                    <1>       ; 20/05/2017
3432                                    <1>       ; 01/05/2017, 02/05/2017, 15/05/2017
3433                                    <1>       ; 04/03/2017
3434                                    <1>       ; 20/02/2017 (TRDOS 386 = TRDOS v2.0)
3435                                    <1>       ;
3436                                    <1>       ; Allocate physically contiguous user pages (memory block)
3437                                    <1>       ; (caller: 'sysalloc' system call)
3438                                    <1>       ;
3439                                    <1>       ; Note: This procedure does not alloc a page's itself
3440                                    <1>       ;     (page bit) on Memory Allocation Table.
3441                                    <1>       ;     (allocate_memory_block is needed before this proc)
3442                                    <1>       ;
3443                                    <1>       ; INPUT ->
3444                                    <1>       ;     EAX = PHYSICAL ADDRESS (beginning address)
3445                                    <1>       ;     EBX = VIRTUAL ADDRESS (beginning address)
3446                                    <1>       ;     ECX = byte count (>=4096)
3447                                    <1>       ;     [u.pgdir] = user's page directory
3448                                    <1>       ;
3449                                    <1>       ;     Note: All addresses are (must be) already adjusted
3450                                    <1>       ;     to page      borders, otherwise, lower 12bits of addresses
3451                                    <1>       ;     and byte count would be truncated.
3452                                    <1>       ;
3453                                    <1>       ; OUTPUT ->
3454                                    <1>       ;     none
3455                                    <1>       ;
3456                                    <1>       ;     CF = 1 -> insufficient memory error
3457                                    <1>       ;
3458                                    <1>       ; Note: All pages will be allocated in physical page order
3459                                    <1>       ;     from the beginning page address.
3460                                    <1>       ;     * A new page table will be added to the page dir
3461                                    <1>       ;       when the requested PDE is invalid.
3462                                    <1>       ;     * Those pages will not be added to swap queue
3463                                    <1>       ;       because main purpose of this allocation is to
3464                                    <1>       ;       set a direct memory access (DMA controller) buffer.
3465                                    <1>       ;      (Swapping out a page in a DMA buffer would be wrong!)
3466                                    <1>       ;     * Previous content of page tables (PTEs) would be
3467                                    <1>       ;       (should be) deallocated before entering this
3468                                    <1>       ;       procedure. So, new page table entries (PTEs)
3469                                    <1>       ;       directly will be written without checking
3470                                    <1>       ;       their previous content.
3471                                    <1>       ;     * Only solution to increase free memory by removing
3472                                    <1>       ;       that non-swappable memory block is to terminate
3473                                    <1>       ;       the process or to wait until the process will
3474                                    <1>       ;       deallocate that memory block as itself. ('sysdalloc')
3475                                    <1>       ;       (No problem, if the process does not grab all of
3476                                    <1>       ;       -very big amount of- free memory by using
3477                                    <1>       ;       'sysalloc' system call!?)
3478                                    <1>       ;       (Even if the process has grabbed all of free memory,
3479                                    <1>       ;       no problem if the process is not running in
3480                                    <1>       ;       multitasking mode. No problem in multitasking
3481                                    <1>       ;       mode if there is not another process which is running
3482                                    <1>       ;       or waiting or sleeping for an event as it's pages
3483                                    <1>       ;       are swapped-out. But a new process can not start to
3484                                    <1>       ;       run if all of free memory has beeen allocated
3485                                    <1>       ;       by running processes. Deallocation -'sysdalloc'-
3486                                    <1>       ;       or terminate a running process is needed
3487                                    <1>       ;       in order to run a new process.)
3488                                    <1>       ;
3489                                    <1>       ; Modified Registers -> EAX, EDX, ESI, EDI, EBX, ECX, EBP
3490                                    <1>       ;
3491                                    <1>
3492                                    <1>       ; 01/05/2017
```

```
3493 00005873 662500F0           <1>       and    ax, ~PAGE_OFF
3494 00005877 6681E300F0         <1>       and    bx, ~PAGE_OFF
3495                             <1>       ; 02/05/2017
3496 0000587C BD00F0FFFF         <1>       mov    ebp, 0FFFFF000h ; 4 Giga Bytes - 4096 Bytes (for Stack)
3497 00005881 C1E90C             <1>       shr    ecx, PAGE_SHIFT ; page count
3498 00005884 83F901             <1>       cmp    ecx, 1
3499 00005887 7251               <1>       jb     short a_u_im_retn
3500 00005889 89C2               <1>       mov    edx, eax
3501 0000588B 01CA               <1>       add    edx, ecx
3502 0000588D 724B               <1>       jc     short a_u_im_retn
3503 0000588F 39D5               <1>       cmp    ebp, edx
3504 00005891 7247               <1>       jb     short a_u_im_retn
3505 00005893 89DA               <1>       mov    edx, ebx
3506 00005895 81C200004000       <1>       add    edx, CORE
3507 0000589B 723D               <1>       jc     short a_u_im_retn
3508 0000589D 01CA               <1>       add    edx, ecx
3509 0000589F 7239               <1>       jc     short a_u_im_retn
3510 000058A1 39D5               <1>       cmp    ebp, edx
3511 000058A3 7235               <1>       jb     short a_u_im_retn
3512                             <1>       ;
3513 000058A5 89C5               <1>       mov    ebp, eax ; physical address
3514 000058A7 89DE               <1>       mov    esi, ebx
3515 000058A9 81C600004000       <1>       add    esi, CORE ; start of user's memory (4M)
3516 000058AF C1EE0C             <1>       shr    esi, PAGE_SHIFT ; higher 20 bits of the linear address
3517                             <1>       ;shr   ecx, PAGE_SHIFT ; page count
3518 000058B2 8B1D[B8030300]     <1>       mov    ebx, [u.pgdir] ; physical addr of user's page dir
3519 000058B8 89F7               <1>       mov    edi, esi
3520 000058BA 81E7FF030000       <1>       and    edi, PTE_MASK ; PTE entry index in the page table
3521 000058C0 57                 <1>       push   edi  ; * ; PTE index (in page directory)
3522 000058C1 C1EE0A             <1>       shr    esi, PAGE_D_SHIFT - PAGE_SHIFT ; 22-12=10
3523 000058C4 89F2               <1>       mov    edx, esi
3524                             <1>       ; EDX = PDE index
3525 000058C6 C1E602             <1>       shl    esi, 2 ; convert PDE index to dword offset
3526 000058C9 01DE               <1>       add    esi, ebx ; add page directory address
3527                             <1> a_u_pd_0:
3528 000058CB AD                 <1>       lodsd
3529                             <1>       ;
3530 000058CC 89F3               <1>       mov    ebx, esi ; next PDE address
3531                             <1>       ;
3532 000058CE A801               <1>       test   al, PDE_A_PRESENT ; bit 0, present flag (must be 1)
3533 000058D0 7513               <1>       jnz    short a_u_pd_2
3534                             <1>       ;
3535                             <1>       ; empty PDE (it does not point to valid page table address)
3536 000058D2 E8A2F2FFFF         <1>       call   allocate_page  ; (allocate a new page table)
3537 000058D7 7302               <1>       jnc    short a_u_pd_1 ; OK... now, we have a new page table.
3538                             <1>       ; cf = 1
3539                             <1>       ; There is not a free memory page to allocate a new page table !!!
3540 000058D9 5E                 <1>       pop    esi ; *
3541                             <1> a_u_im_retn:
3542 000058DA C3                 <1>       retn   ; return to 'sysalloc' with 'insufficient memory' error
3543                             <1>       ;
3544                             <1> a_u_pd_1: ; clear the new page table content
3545                             <1>       ; EAX = Physical (base) address of the new page table
3546 000058DB E813F3FFFF         <1>       call   clear_page ; Clear page content
3547                             <1>       ;
3548 000058E0 0C07               <1>       or     al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
3549                             <1>              ; set bit 0, bit 1 and bit 2 to 1
3550                             <1>              ; (present, writable, user)
3551 000058E2 8946FC             <1>       mov    [esi-4], eax
3552                             <1> a_u_pd_2:
3553 000058E5 662500F0           <1>       and    ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3554                             <1>       ; EAX = PHYSICAL (flat) ADDRESS OF THE PAGE TABLE
3555 000058E9 8B3C24             <1>       mov    edi, [esp] ; *
3556                             <1>       ; EDI = PTE index (of page directory)
3557                             <1>       ;and   edi, PTE_MASK ; PTE entry index in the page table
3558                             <1>       ; EBX = next PDE address
3559 000058EC 89FE               <1>       mov    esi, edi ; PTE index in page table (0-1023)
3560 000058EE C1E702             <1>       shl    edi, 2 ; convert PTE index to dword offset
3561 000058F1 01C7               <1>       add    edi, eax ; now, edi points to requested PTE
3562                             <1> a_u_pt_0:
3563                             <1>       ; 02/05/2017
3564 000058F3 8B07               <1>       mov    eax, [edi]
3565                             <1>       ;
3566 000058F5 A801               <1>       test   al, PTE_A_PRESENT ; bit 0, present flag (must be 1)
3567 000058F7 7445               <1>       jz     short a_u_pt_1
3568                             <1>       ;
3569 000058F9 A802               <1>       test   al, PTE_A_WRITE   ; bit 1, writable (r/w) flag
3570                             <1>                                ; (must be 1)
3571 000058FB 7550               <1>       jnz    short a_u_pt_3
3572                             <1>       ; Read only -duplicated- page (belongs to a parent or a child)
3573 000058FD 66A90002           <1>         test    ax, PTE_DUPLICATED ; Was this page duplicated
3574                             <1>                                    ; as child's page ?
3575 00005901 7455               <1>       jz     short a_u_pt_4      ; Clear PTE but don't deallocate the page!
3576                             <1>       ;
3577                             <1>       ; check the parent's PTE value is read only & same page or not..
3578                             <1>       ; EDX = page directory entry index (0-1023)
3579 00005903 52                 <1>       push   edx ; **
3580 00005904 53                 <1>       push   ebx ; ***
3581                             <1>       ; ESI = page table entry index (0-1023)
3582                             <1>       ;push  esi ; **** ; 20/05/2017
3583 00005905 8B1D[BC030300]     <1>       mov    ebx, [u.ppgdir] ; page directory of the parent process
3584 0000590B 66C1E202           <1>       shl    dx, 2 ; *4
3585 0000590F 01D3               <1>       add    ebx, edx ; PTE address,0 (for the parent)
3586 00005911 8B13               <1>       mov    edx, [ebx] ; page table address
3587 00005913 F6C201             <1>       test   dl, PDE_A_PRESENT ; present (valid) or not ?
3588 00005916 7433               <1>       jz     short a_u_pt_2      ; parent process does not use this page
3589 00005918 6681E200F0         <1>       and    dx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
3590 0000591D 66C1E602           <1>       shl    si, 2 ; *4
3591                             <1>       ; ESI = page table entry offset (0-4092)
3592 00005921 01D6               <1>       add    esi, edx     ; PTE address (for the parent)
3593 00005923 8B1E               <1>       mov    ebx, [esi]
3594 00005925 F6C301             <1>       test   bl, PTE_A_PRESENT ; present or not ?
```

```
3595 00005928 7421              <1>       jz    short a_u_pt_2      ; parent process does not use this page
3596 0000592A 662500F0          <1>       and   ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3597 0000592E 6681E300F0        <1>       and   bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3598 00005933 39D8              <1>       cmp   eax, ebx      ; parent's and child's pages are same ?
3599 00005935 7514              <1>       jne   short a_u_pt_2      ; not same page
3600                            <1>                             ; deallocate the child's page
3601 00005937 800E02            <1>         or      byte [esi], PTE_A_WRITE ; convert to writable page (parent)
3602                            <1>       ;pop  esi ; **** ; 20/05/2017
3603 0000593A 5B                <1>       pop   ebx ; ***
3604 0000593B 5A                <1>       pop   edx ; **
3605 0000593C EB1A              <1>       jmp   short a_u_pt_4
3606                            <1> a_u_pt_1:
3607 0000593E 09C0              <1>       or    eax, eax      ; swapped page ?
3608 00005940 7416              <1>       jz    short a_u_pt_4      ; no
3609                            <1>                             ; yes
3610 00005942 D1E8              <1>       shr   eax, 1
3611 00005944 E8B5F8FFFF        <1>       call  unlink_swap_block ; Deallocate swapped page block
3612                            <1>                               ; on the swap disk (or in file)
3613 00005949 EB0D              <1>       jmp   short a_u_pt_4
3614                            <1> a_u_pt_2:
3615                            <1>       ;pop  esi ; **** ; 20/05/2017
3616 0000594B 5B                <1>       pop   ebx ; ***
3617 0000594C 5A                <1>       pop   edx ; **
3618                            <1> a_u_pt_3:
3619 0000594D 66A90004          <1>       test  ax, PTE_SHARED      ; shared or direct memory access indicator
3620 00005951 7505              <1>       jnz   short a_u_pt_4      ; AVL bit 1 = 1, do not deallocate this page!
3621                            <1>       ;
3622                            <1>       ;and  ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3623 00005953 E8FFF3FFFF        <1>       call  deallocate_page ; set the mem allocation bit of this page
3624                            <1>       ;
3625                            <1> a_u_pt_4:
3626 00005958 89E8              <1>       mov   eax, ebp ; physical address
3627 0000595A 0C07              <1>       or    al, PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER ; 04/03/2017
3628 0000595C AB                <1>       stosd
3629 0000595D 5E                <1>       pop   esi ; * ; 20/05/2017
3630 0000595E 49                <1>       dec   ecx ; remain page count
3631 0000595F 7417              <1>       jz    short a_u_pd_5
3632 00005961 81C500100000      <1>       add   ebp, PAGE_SIZE
3633 00005967 46                <1>       inc   esi ; next PTE (index)
3634                            <1>       ; 20/05/2017
3635                            <1>       ;cmp  esi, PAGE_SIZE/4 ; 1024
3636                            <1>       ;jb   short a_u_pt_0
3637 00005968 6681E6FF03        <1>       and   si, PTE_MASK ; 3FFh (0 to 1023)
3638 0000596D 56                <1>       push  esi ; *
3639 0000596E 7583              <1>       jnz   short a_u_pt_0 ; > 0 (<1024)
3640                            <1> a_u_pd_3:
3641 00005970 42                <1>       inc   edx
3642                            <1> ;     cmp   edx, 1024
3643                            <1> ;     jnb   short a_u_pd_4 ; 02/05/2017 (error!, ecx > 0)
3644 00005971 89DE              <1>       mov   esi, ebx ; the next PDE address
3645 00005973 E953FFFFFF        <1>       jmp   a_u_pd_0
3646                            <1> a_u_pd_4:
3647                            <1>       ; 02/05/2017
3648                            <1> ;     stc
3649                            <1> a_u_pd_5:
3650                            <1>       ; 20/05/2017
3651                            <1>       ;pop  edi ; *
3652 00005978 C3                <1>       retn
3653                            <1>
3654                            <1>
3655                            <1> ; /// End Of MEMORY MANAGEMENT FUNCTIONS ///
3656                            <1>
3657                            <1> ;; Data:
3658                            <1>
3659                            <1> ; 09/03/2015
3660                            <1> ;swpq_count: dw 0 ; count of pages on the swap que
3661                            <1> ;swp_drv:    dd 0 ; logical drive description table address of the swap drive/disk
3662                            <1> ;swpd_size:  dd 0 ; size of swap drive/disk (volume) in sectors (512 bytes).

3663                            <1> ;swpd_free:  dd 0 ; free page blocks (4096 bytes) on swap disk/drive (logical)
3664                            <1> ;swpd_next:  dd 0 ; next free page block
3665                            <1> ;swpd_last:  dd 0 ; last swap page block
2159                                %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> ; ----------------------------------------------------------------------------
   6                            <1> ; Beginning: 17/01/2016
   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> ;
  15                            <1> ; Derived from 'IBM PC-AT' BIOS source code (1985)
  16                            <1> ; ****************************************************************************
  17                            <1>
  18                            <1> ; TRDOS 386  (TRDOS v2.0) Kernel - TIMER & REAL TIME CLOCK (BIOS) FUNCTIONS
  19                            <1>
  20                            <1> ; IBM PC-AT BIOS Source Code ('BIOS2.ASM')
  21                            <1> ; TITLE BIOS2 ---- 06/10/85 BIOS INTERRUPT ROUTINES
  22                            <1>
  23                            <1> ;
  24                            <1> ; ///////// TIMER (& REAL TIME CLOCK) FUNCTIONS //////////////
  25                            <1>
  26                            <1> int1Ah:
  27                            <1>       ; 29/01/2016
  28                            <1>       ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
  29 00005979 9C                <1>       pushfd
```

```
30 0000597A 0E                  <1>       push   cs
31 0000597B E801000000          <1>       call   TIME_OF_DAY_1
32 00005980 C3                  <1>       retn
33                              <1>
34                              <1> ;--- INT  1A H -- (TIME OF DAY) ------------------------------------------------
35                              <1> ;       THIS BIOS ROUTINE ALLOWS THE CLOCKS TO BE SET OR READ                 :
36                              <1> ;                                                                             :
37                              <1> ; PARAMETERS:                                                                 :
38                              <1> ;     (AH) = 00H  READ THE CURRENT SETTING AND RETURN WITH,                   :
39                              <1> ;                       (CX) = HIGH PORTION OF COUNT                          :
40                              <1> ;                       (DX) = LOW PORTION OF COUNT                           :
41                              <1> ;                       (AL) = 0 TIMER HAS NOT PASSED 24 HOURS SINCE LAST READ :
42                              <1> ;                              1 IF ON ANOTHER DAY. (RESET TO ZERO AFTER READ) :
43                              <1> ;                                                                             :
44                              <1> ;     (AH) = 01H  SET THE CURRENT CLOCK USING,                                :
45                              <1> ;                 (CX) = HIGH PORTION OF COUNT                                :
46                              <1> ;                 (DX) = LOW PORTION OF COUNT.                                :
47                              <1> ;                                                                             :
48                              <1> ;                NOTE: COUNTS OCCUR AT THE RATE OF 1193180/65536 COUNTS/SECOND :
49                              <1> ;                           (OR ABOUT 18.2 PER SECOND -- SEE EQUATES)         :
50                              <1> ;                                                                             :
51                              <1> ;     (AH) = 02H  READ THE REAL TIME CLOCK AND RETURN WITH,                   :
52                              <1> ;                       (CH) = HOURS IN BCD (00-23)                           :
53                              <1> ;                       (CL) = MINUTES IN BCD (00-59)                         :
54                              <1> ;                       (DH) = SECONDS IN BCD (00-59)                         :
55                              <1> ;                       (DL) = DAYLIGHT SAVINGS ENABLE (00-01)                :
56                              <1> ;                                                                             :
57                              <1> ;     (AH) = 03H  SET THE REAL TIME CLOCK USING,                              :
58                              <1> ;                       (CH) = HOURS IN BCD (00-23)                           :
59                              <1> ;                       (CL) = MINUTES IN BCD (00-59)                         :
60                              <1> ;                       (DH) = SECONDS IN BCD (00-59)                         :
61                              <1> ;                       (DL) = 01 IF DAYLIGHT SAVINGS ENABLE OPTION, ELSE 00.  :
62                              <1> ;                                                                             :
63                              <1> ;             NOTE: (DL) = 00 IF DAYLIGHT SAVINGS TIME ENABLE IS NOT ENABLED. :
64                              <1> ;                   (DL) = 01 ENABLES TWO SPECIAL UPDATES THE LAST SUNDAY IN  :
65                              <1> ;                APRIL   (1:59:59 --> 3:00:00 AM) AND THE LAST SUNDAY IN      :
66                              <1> ;                   OCTOBER (1:59:59 --> 1:00:00 AM) THE FIRST TIME.          :
67                              <1> ;                                                                             :
68                              <1> ;     (AH) = 04H  READ THE DATE FROM THE REAL TIME CLOCK AND RETURN WITH,     :
69                              <1> ;                       (CH) = CENTURY IN BCD (19 OR 20)                      :
70                              <1> ;                       (CL) = YEAR IN BCD (00-99)                            :
71                              <1> ;                       (DH) = MONTH IN BCD (01-12)                           :
72                              <1> ;                       (DL) = DAY IN BCD (01-31).                            :
73                              <1> ;                                                                             :
74                              <1> ;     (AH) = 05H  SET THE DATE INTO THE REAL TIME CLOCK USING,                :
75                              <1> ;                       (CH) = CENTURY IN BCD (19 OR 20)                      :
76                              <1> ;                       (CL) = YEAR IN BCD (00-99)                            :
77                              <1> ;                       (DH) = MONTH IN BCD (01-12)                           :
78                              <1> ;                       (DL) = DAY IN BCD (01-31).                            :
79                              <1> ;                                                                             :
80                              <1> ;     (AH) = 06H  SET THE ALARM TO INTERRUPT AT SPECIFIED TIME,               :
81                              <1> ;                       (CH) = HOURS IN BCD (00-23 (OR FFH))                  :
82                              <1> ;                       (CL) = MINUTES IN BCD (00-59 (OR FFH))                :
83                              <1> ;                       (DH) = SECONDS IN BCD (00-59 (OR FFH))                :
84                              <1> ;                                                                             :
85                              <1> ;     (AH) = 07H  RESET THE ALARM INTERRUPT FUNCTION.                         :
86                              <1> ;                                                                             :
87                              <1> ; NOTES: FOR ALL RETURNS CY= 0 FOR SUCCESSFUL OPERATION.                      :
88                              <1> ;        FOR (AH)= 2, 4, 6 - CARRY FLAG SET IF REAL TIME CLOCK NOT OPERATING. :
89                              <1> ;        FOR (AH)= 6 - CARRY FLAG SET IF ALARM ALREADY ENABLED.               :
90                              <1> ;        FOR THE ALARM FUNCTION (AH = 6) THE USER MUST SUPPLY A ROUTINE AND   :
91                              <1> ;         INTERCEPT THE CORRECT ADDRESS IN THE VECTOR TABLE FOR INTERRUPT 4AH.       :
92                              <1> ;         USE 0FFH FOR ANY "DO NOT CARE" POSITION FOR INTERVAL INTERRUPTS.    :
93                              <1> ;        INTERRUPTS ARE DISABLED DURING DATA MODIFICATION.                    :
94                              <1> ;        AH & AL ARE RETURNED MODIFIED AND NOT DEFINED EXCEPT WHERE INDICATED.      :
95                              <1> ;--------------------------------------------------------------------------------
96                              <1>
97                              <1> ; 15/01/2017
98                              <1> ; 14/01/2017
99                              <1> ; 07/01/2017
100                             <1> ; 02/01/2017
101                             <1> ; 29/05/2016
102                             <1> ; 29/01/2016
103                             <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
104                             <1>
105                             <1> ; 29/05/2016
106                             <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
107                             <1> int35h:  ; Date/Time functions
108                             <1>
109                             <1> TIME_OF_DAY_1:
110                             <1>       ;sti                      ; INTERRUPTS BACK ON
111                             <1>       ; 29/05/2016
112 00005981 80642408FE        <1>       and    byte [esp+8], 11111110b   ; clear carry bit of eflags register
113                             <1>       ;
114 00005986 80FC08            <1>       cmp    ah, (RTC_TBE-RTC_TB)/4    ; CHECK IF COMMAND IN VALID RANGE (0-7)
115 00005989 F5                <1>       cmc                       ; COMPLEMENT CARRY FOR ERROR EXIT
116                             <1>       ; (*) jc short TIME_9           ; EXIT WITH CARRY = 1 IF NOT VALID
117 0000598A 721A              <1>       jc     short _TIME_9 ; 29/05/2016
118                             <1>
119 0000598C 1E                <1>       push   ds
120 0000598D 56                <1>       push   esi
121 0000598E 66BE1000          <1>       mov    si, KDATA          ; kernel data segment
122 00005992 8EDE              <1>       mov    ds, si
123                             <1>
124                             <1>       ;;15/01/2017
125                             <1>       ; 14/01/2017
126                             <1>       ; 02/01/2017
127                             <1>       ;;mov  byte [intflg], 35h  ; date & time interrupt
128                             <1>       ;sti
129                             <1>       ;
130 00005994 C0E402            <1>       shl    ah, 2              ; convert function to dword offset
131 00005997 0FB6F4            <1>       movzx  esi, ah                  ; PLACE INTO ADDRESSING REGISTER
```

```
                                <1>       ;cli                          ; NO INTERRUPTS DURING TIME FUNCTIONS
000059A A FF96[AC590000]        <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
000059A0 B400                   <1>       mov    ah, 0                  ; CLEAR (AH) TO ZERO
000059A2 5E                     <1>       pop    esi                    ; RECOVER USERS REGISTER
000059A3 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
000059A4 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
000059A6 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:
000059AB 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:
000059AC [CC590000]             <1>       dd     RTC_00                 ; 0 = READ CURRENT CLOCK COUNT
000059B0 [DF590000]             <1>       dd     RTC_10                 ; 1 = SET CLOCK COUNT
000059B4 [ED590000]             <1>       dd     RTC_20                 ; 2 = READ THE REAL TIME CLOCK TIME
000059B8 [1C5A0000]             <1>       dd     RTC_30                 ; 3 = SET REAL TIME CLOCK TIME
000059BC [5E5A0000]             <1>       dd     RTC_40                 ; 4 = READ THE REAL TIME CLOCK DATE
000059C0 [8B5A0000]             <1>       dd     RTC_50                 ; 5 = SET REAL TIME CLOCK DATE
000059C4 [D85A0000]             <1>       dd     RTC_60                 ; 6 = SET THE REAL TIME CLOCK ALARM
000059C8 [2B5B0000]             <1>       dd     RTC_70                 ; 7 = RESET ALARM
                                <1>
                                <1> RTC_TBE     equ    $
                                <1>
                                <1> RTC_00:                             ; READ TIME COUNT
000059CC A0[84520100]           <1>       mov    al, [TIMER_OFL]              ; GET THE OVERFLOW FLAG
000059D1 C605[84520100]00       <1>       mov    byte [TIMER_OFL], 0 ; AND THEN RESET THE OVERFLOW FLAG
000059D8 8B0D[80520100]         <1>         mov     ecx, [TIMER_LH]          ; GET COUNT OF TIME
000059DE C3                     <1>       retn
                                <1>
                                <1> RTC_10:                             ; SET TIME COUNT
000059DF 890D[80520100]         <1>         mov     [TIMER_LH], ecx          ; SET TIME COUNT
000059E5 C605[84520100]00       <1>       mov    byte [TIMER_OFL], 0 ; RESET OVERFLOW FLAG
000059EC C3                     <1>       retn                           ; RETURN WITH NO CARRY
                                <1>
                                <1> RTC_20:                             ; GET RTC TIME
000059ED E8EB010000             <1>       call   UPD_IPR                       ; CHECK FOR UPDATE IN PROCESS
000059F2 7227                   <1>       jc     short RTC_29            ; EXIT IF ERROR (CY= 1)
                                <1>
000059F4 B000                   <1>       mov    al, CMOS_SECONDS        ; SET ADDRESS OF SECONDS
000059F6 E8FD010000             <1>       call   CMOS_READ               ; GET SECONDS
000059FB 88C6                   <1>       mov    dh, al                  ; SAVE
000059FD B00B                   <1>       mov    al, CMOS_REG_B               ; ADDRESS ALARM REGISTER
000059FF E8F4010000             <1>       call   CMOS_READ               ; READ CURRENT VALUE OF DSE BIT
00005A04 2401                   <1>       and    al, 00000001b           ; MASK FOR VALID DSE BIT
00005A06 88C2                   <1>       mov    dl, al                  ; SET [DL] TO ZERO FOR NO DSE BIT
00005A08 B002                   <1>       mov    al, CMOS_MINUTES        ; SET ADDRESS OF MINUTES
00005A0A E8E9010000             <1>       call   CMOS_READ               ; GET MINUTES
00005A0F 88C1                   <1>       mov    cl, al                  ; SAVE
00005A11 B004                   <1>         mov     al, CMOS_HOURS           ; SET ADDRESS OF HOURS
00005A13 E8E0010000             <1>       call   CMOS_READ               ; GET HOURS
00005A18 88C5                   <1>       mov    ch, al                  ; SAVE
00005A1A F8                     <1>       clc                            ; SET CY= 0
                                <1> RTC_29:
00005A1B C3                     <1>       retn                           ; RETURN WITH RESULT IN CARRY FLAG
                                <1>
                                <1> RTC_30:                             ; SET RTC TIME
00005A1C E8BC010000             <1>       call   UPD_IPR                       ; CHECK FOR UPDATE IN PROCESS
00005A21 7305                   <1>       jnc    short RTC_35            ; GO AROUND IF CLOCK OPERATING
00005A23 E817010000             <1>       call   RTC_STA                       ; ELSE TRY INITIALIZING CLOCK
                                <1> RTC_35:
00005A28 88F4                   <1>       mov    ah, dh                  ; GET TIME BYTE - SECONDS
00005A2A B000                   <1>       mov    al, CMOS_SECONDS        ; ADDRESS SECONDS
00005A2C E8E0010000             <1>       call   CMOS_WRITE              ; UPDATE SECONDS
00005A31 88CC                   <1>       mov    ah, cl                  ; GET TIME BYTE - MINUTES
00005A33 B002                   <1>       mov    al, CMOS_MINUTES        ; ADDRESS MINUTES
00005A35 E8D7010000             <1>       call   CMOS_WRITE              ; UPDATE MINUTES
```

```
00005A3A 88EC              <1>       mov   ah, ch              ; GET TIME BYTE - HOURS
00005A3C B004              <1>       mov   al, CMOS_HOURS            ; ADDRESS HOURS
00005A3E E8CE010000        <1>       call  CMOS_WRITE          ; UPDATE ADDRESS
                           <1>       ;mov  al, CMOS_REG_B            ; ADDRESS ALARM REGISTER
                           <1>       ;mov  ah, al
00005A43 66B80B0B          <1>       mov   ax, CMOS_REG_B * 257
00005A47 E8AC010000        <1>       call  CMOS_READ           ; READ CURRENT TIME
00005A4C 2462              <1>       and   al, 01100010b       ; MASK FOR VALID BIT POSITIONS
00005A4E 0C02              <1>       or    al, 00000010b       ; TURN ON 24 HOUR MODE
00005A50 80E201            <1>       and   dl, 00000001b       ; USE ONLY THE DSE BIT
00005A53 08D0              <1>       or    al, dl              ; GET DAY LIGHT SAVINGS TIME BIT (OSE)
00005A55 86E0              <1>       xchg  ah, al              ; PLACE IN WORK REGISTER AND GET ADDRESS
00005A57 E8B5010000        <1>       call  CMOS_WRITE          ; SET NEW ALARM SITS
00005A5C F8                <1>       clc                       ; SET CY= 0
00005A5D C3                <1>       retn                      ; RETURN WITH CY= 0
                           <1>
                           <1> RTC_40:                         ; GET RTC DATE
00005A5E E87A010000        <1>       call  UPD_IPR                   ; CHECK FOR UPDATE IN PROCESS
00005A63 7225              <1>       jc    short RTC_49        ; EXIT IF ERROR (CY= 1)
                           <1>
00005A65 B007              <1>       mov   al, CMOS_DAY_MONTH  ; ADDRESS DAY OF MONTH
00005A67 E88C010000        <1>       call  CMOS_READ           ; READ DAY OF MONTH
00005A6C 88C2              <1>       mov   dl, al              ; SAVE
00005A6E B008              <1>       mov   al, CMOS_MONTH            ; ADDRESS MONTH
00005A70 E883010000        <1>       call  CMOS_READ           ; READ MONTH
00005A75 88C6              <1>       mov   dh, al              ; SAVE
00005A77 B009              <1>       mov   al, CMOS_YEAR       ; ADDRESS YEAR
00005A79 E87A010000        <1>       call  CMOS_READ           ; READ YEAR
00005A7E 88C1              <1>       mov   cl, al              ; SAVE
00005A80 B032              <1>       mov   al, CMOS_CENTURY    ; ADDRESS CENTURY LOCATION
00005A82 E871010000        <1>       call  CMOS_READ           ; GET CENTURY BYTE
00005A87 88C5              <1>       mov   ch, al              ; SAVE
00005A89 F8                <1>       clc                       ; SET CY=0
                           <1> RTC_49:
00005A8A C3                <1>       retn                      ; RETURN WITH RESULTS IN CARRY FLAG
                           <1>
                           <1> RTC_50:                         ; SET RTC DATE
00005A8B E84D010000        <1>       call  UPD_IPR                   ; CHECK FOR UPDATE IN PROCESS
00005A90 7305              <1>       jnc   short RTC_55        ; GO AROUND IF NO ERROR
00005A92 E8A8000000        <1>       call  RTC_STA                   ; ELSE INITIALIZE CLOCK
                           <1> RTC_55:
00005A97 66B80600          <1>       mov   ax, CMOS_DAY_WEEK   ; ADDRESS OF DAY OF WEEK BYTE
00005A9B E871010000        <1>       call  CMOS_WRITE          ; LOAD ZEROS TO DAY OF WEEK
00005AA0 88D4              <1>       mov   ah, dl              ; GET DAY OF MONTH BYTE
00005AA2 B007              <1>       mov   al, CMOS_DAY_MONTH  ; ADDRESS DAY OF MONTH BYTE
00005AA4 E868010000        <1>       call  CMOS_WRITE          ; WRITE OF DAY OF MONTH REGISTER
00005AA9 88F4              <1>       mov   ah, dh              ; GET MONTH
00005AAB B008              <1>       mov   al, CMOS_MONTH            ; ADDRESS MONTH BYTE
00005AAD E85F010000        <1>       call  CMOS_WRITE          ; WRITE MONTH REGISTER
00005AB2 88CC              <1>       mov   ah, cl              ; GET YEAR BYTE
00005AB4 B009              <1>       mov   al, CMOS_YEAR       ; ADDRESS YEAR REGISTER
00005AB6 E856010000        <1>       call  CMOS_WRITE          ; WRITE YEAR REGISTER
00005ABB 88EC              <1>       mov   ah, ch              ; GET CENTURY BYTE
00005ABD B032              <1>       mov   al, CMOS_CENTURY    ; ADDRESS CENTURY BYTE
00005ABF E84D010000        <1>       call  CMOS_WRITE          ; WRITE CENTURY LOCATION
                           <1>       ;mov  al, CMOS_REG_B            ; ADDRESS ALARM REGISTER
                           <1>       ;mov  ah, al
00005AC4 66B80B0B          <1>       mov   ax, CMOS_REG_B * 257
00005AC8 E82B010000        <1>       call  CMOS_READ           ; READ WIRRENT SETTINGS
00005ACD 247F              <1>       and   al, 07Fh            ; CLEAR 'SET BIT'
00005ACF 86E0              <1>       xchg  ah, al              ; MOVE TO WORK REGISTER
00005AD1 E83B010000        <1>       call  CMOS_WRITE          ; AND START CLOCK UPDATING
00005AD6 F8                <1>       clc                       ; SET CY= 0
00005AD7 C3                <1>       retn                      ; RETURN CY=0
                           <1>
                           <1> RTC_60:                         ; SET RTC ALARM
00005AD8 B00B              <1>       mov   al, CMOS_REG_B            ; ADDRESS ALARM
00005ADA E819010000        <1>       call  CMOS_READ           ; READ ALARM REGISTER
00005ADF A820              <1>       test  al, 20h                   ; CHECK FOR ALARM ALREADY ENABLED
00005AE1 F9                <1>       stc                       ; SET CARRY IN CASE OF ERROR
00005AE2 7542              <1>       jnz   short RTC_69        ; ERROR EXIT IF ALARM SET
00005AE4 E8F4000000        <1>       call  UPD_IPR                   ; CHECK FOR UPDATE IN PROCESS
00005AE9 7305              <1>       jnc   short RTC_65        ; SKIP INITIALIZATION IF NO ERROR
00005AEB E84F000000        <1>       call  RTC_STA                   ; ELSE INITIALIZE CLOCK
                           <1> RTC_65:
00005AF0 88F4              <1>       mov   ah, dh              ; GET SECONDS BYTE
00005AF2 B001              <1>       mov   al, CMOS_SEC_ALARM  ; ADDRESS THE SECONDS ALARM REGISTER
00005AF4 E818010000        <1>       call  CMOS_WRITE          ; INSERT SECONDS
00005AF9 88CC              <1>       mov   ah, cl              ; GET MINUTES PARAMETER
00005AFB B003              <1>       mov   al, CMOS_MIN_ALARM  ; ADDRESS MINUTES ALARM REGISTER
00005AFD E80F010000        <1>       call  CMOS_WRITE          ; INSERT MINUTES
00005B02 88EC              <1>       mov   ah, ch              ; GET HOURS PARAMETER
00005B04 B005              <1>       mov   al, CMOS_HR_ALARM   ; ADDRESS HOUR ALARM REGISTER
00005B06 E806010000        <1>       call  CMOS_WRITE          ; INSERT HOURS
00005B0B E4A1              <1>       in    al, INTB01          ; READ SECOND INTERRUPT MASK REGISTER
00005B0D 24FE              <1>       and   al, 0FEh            ; ENABLE ALARM TIMER BIT (CY= 0)
00005B0F E6A1              <1>       out   INTB01, al          ; WRITE UPDATED MASK
                           <1>       ;mov  al, CMOS_REG_B            ; ADDRESS ALARM REGISTER
                           <1>       ;mov  ah, al
00005B11 66B80B0B          <1>       mov   ax, CMOS_REG_B * 257
00005B15 E8DE000000        <1>       call  CMOS_READ           ; READ CURRENT ALARM REGISTER
00005B1A 247F              <1>       and   al, 07Fh            ; ENSURE SET BIT TURNED OFF
00005B1C 0C20              <1>       or    al, 20h                   ; TURN ON ALARM ENABLE
00005B1E 86E0              <1>       xchg  ah, al              ; MOVE MASK TO OUTPUT REGISTER
00005B20 E8EC000000        <1>       call  CMOS_WRITE          ; WRITE NEW ALARM MASK
00005B25 F8                <1>       clc                       ; SET CY= 0
                           <1> RTC_69:
00005B26 66B80000          <1>       mov   ax, 0               ; CLEAR AX REGISTER
00005B2A 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
00005B2B 66B80B0B               <1>       mov    ax, CMOS_REG_B * 257      ; ADDRESS ALARM REGISTER (TO BOTH AH,AL)
00005B2F E8C4000000             <1>       call   CMOS_READ           ; READ ALARM REGISTER
00005B34 2457                   <1>       and    al, 57h                   ; TURN OFF ALARM ENABLE
00005B36 86E0                   <1>       xchg   ah, al             ; SAVE DATA AND RECOVER ADDRESS
00005B38 E8D4000000             <1>       call   CMOS_WRITE          ; RESTORE NEW VALUE
00005B3D F8                     <1>       clc                       ; SET CY= 0
00005B3E 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
00005B3F 66B80A26               <1>       mov    ax, (26h*100h)+CMOS_REG_A
00005B43 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
00005B48 66B80B82               <1>       mov    ax, (82h*100h)+CMOS_REG_B
00005B4C E8C0000000             <1>       call   CMOS_WRITE          ; AND 24 HOUR MODE TO REGISTER B
00005B51 B00C                   <1>       mov    al, CMOS_REG_C             ; ADDRESS REGISTER C
00005B53 E8A0000000             <1>       call   CMOS_READ           ; READ REGISTER C TO INITIALIZE
00005B58 B00D                   <1>       mov    al, CMOS_REG_D             ; ADDRESS REGISTER D
00005B5A E899000000             <1>       call   CMOS_READ           ; READ REGISTER D TO INITIALIZE
00005B5F 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
00005B60 1E                     <1>       push   ds                 ; LEAVE INTERRUPTS DISABLED
00005B61 50                     <1>       push   eax                ; SAVE REGISTERS
00005B62 57                     <1>       push   edi
                                <1> RTC_I_1:                          ; CHECK FOR SECOND INTERRUPT
00005B63 66B88C8B               <1>       mov    ax, 256*(CMOS_REG_B+NMI)+CMOS_REG_C+NMI ; ALARM AND STATUS
00005B67 E670                   <1>       out    CMOS_PORT, al      ; WRITE ALARM FLAG MASK ADDRESS
00005B69 90                     <1>       nop                       ; I/O DELAY
00005B6A EB00                   <1>       jmp    short $+2
00005B6C E471                   <1>       in     al, CMOS_DATA      ; READ AND RESET INTERRUPT REQUEST FLAGS
00005B6E A860                   <1>       test   al, 01100000b      ; CHECK FOR EITHER INTERRUPT PENDING
00005B70 745D                   <1>       jz     short  RTC_I_9           ; EXIT IF NOT A VALID RTC INTERRUPT
                                <1>
00005B72 86E0                   <1>       xchg   ah, al             ; SAVE FLAGS AND GET ENABLE ADDRESS
00005B74 E670                   <1>       out    CMOS_PORT, al      ; WRITE ALARM ENABLE MASK ADDRESS
00005B76 90                     <1>       nop                       ; I/O DELAY
00005B77 EB00                   <1>       jmp    short $+2
00005B79 E471                   <1>       in     al, CMOS_DATA      ; READ CURRENT ALARM ENABLE MASK
00005B7B 20E0                   <1>       and    al, ah             ; ALLOW ONLY SOURCES THAT ARE ENABLED
00005B7D A840                   <1>       test   al, 01000000b      ; CHECK FOR PERIODIC INTERRUPT
00005B7F 743B                   <1>       jz     short RTC_I_5      ; SKIP IF NOT A PERIODIC INTERRUPT
                                <1>
                                <1> ;-----      DECREMENT WAIT COUNT BY INTERRUPT INTERVAL
                                <1>
00005B81 66BF1000               <1>       mov    di, KDATA          ; kernel data segment
00005B85 8EDF                   <1>       mov    ds, di
                                <1>
00005B87 812D[78520100]D003-    <1>       sub    dword [RTC_LH], 976 ; DECREMENT COUNT BY 1/1024
00005B8F 0000                   <1>
00005B91 7329                   <1>       jnc    short RTC_I_5      ; SKIP TILL 32 BIT WORD LESS THAN ZERO
                                <1>
                                <1> ;-----      TURN OFF PERIODIC INTERRUPT ENABLE
                                <1>
00005B93 6650                   <1>       push   ax                 ; SAVE INTERRUPT FLAG MASK
00005B95 66B88B8B               <1>       mov    ax, 257*(CMOS_REG_B+NMI) ; INTERRUPT ENABLE REGISTER
00005B99 E670                   <1>       out    CMOS_PORT, al      ; WRITE ADDRESS TO CMOS CLOCK
00005B9B 90                     <1>       nop                       ; I/O DELAY
00005B9C EB00                   <1>       jmp    short $+2
00005B9E E471                   <1>       in     al, CMOS_DATA      ; READ CURRENT ENABLES
00005BA0 24BF                   <1>       and    al, 0BFh           ; TURN OFF PIE
00005BA2 86C4                   <1>       xchg   al, ah             ; GET CMOS ADDRESS AND SAVE VALUE
00005BA4 E670                   <1>       out    CMOS_PORT, al      ; ADDRESS REGISTER B
00005BA6 86C4                   <1>       xchg   al, ah             ; GET NEW INTERRUPT ENABLE MASK
00005BA8 E671                   <1>       out    CMOS_DATA, al      ; SET MASK IN INTERRUPT ENABLE REGISTER
00005BAA C605[7C520100]00       <1>       mov    byte [RTC_WAIT_FLAG], 0   ; SET FUNCTION ACTIVE FLAG OFF
00005BB1 8B3D[7D520100]         <1>       mov    edi, [USER_FLAG]   ; SET UP (DS:DI) TO POINT TO USER FLAG
00005BB7 C60780                 <1>       mov    byte [edi], 80h           ; TURN ON USERS FLAG
00005BBA 6658                   <1>       pop    ax                 ; GET INTERRUPT SOURCE BACK
                                <1> RTC_I_5:
00005BBC A820                   <1>       test   al, 00100000b      ; TEST FOR ALARM INTERRUPT
00005BBE 740D                   <1>       jz     short RTC_I_7      ; SKIP USER INTERRUPT CALL IF NOT ALARM
                                <1>
00005BC0 B00D                   <1>       mov    al, CMOS_REG_D             ; POINT TO DEFAULT READ ONLY REGISTER
00005BC2 E670                   <1>       out    CMOS_PORT, al      ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
00005BC4 FB                     <1>       sti                       ; INTERRUPTS BACK ON NOW
00005BC5 52                     <1>       push   edx
00005BC6 E8E4970000             <1>       call   INT4Ah             ; TRANSFER TO USER ROUTINE
00005BCB 5A                     <1>       pop    edx
00005BCC FA                     <1>       cli                       ; BLOCK INTERRUPT FOR RETRY
                                <1> RTC_I_7:                          ; RESTART ROUTINE TO HANDLE DELAYED
```

```
00005BCD EB94               <1>       jmp    short RTC_I_1       ;  ENTRY AND SECOND EVENT BEFORE DONE
                            <1>
                            <1> RTC_I_9:                         ; EXIT - NO PENDING INTERRUPTS
00005BCF B00D               <1>       mov    al, CMOS_REG_D            ; POINT TO DEFAULT READ ONLY REGISTER
00005BD1 E670               <1>       out    CMOS_PORT, al       ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
00005BD3 B020               <1>       mov    al, EOI                   ; END OF INTERRUPT MASK TO 8259 - 2
00005BD5 E6A0               <1>       out    INTB00, al          ; TO 8259 - 2
00005BD7 E620               <1>       out    INTA00,      al           ; TO 8259 - 1
00005BD9 5F                 <1>       pop    edi                 ; RESTORE REGISTERS
00005BDA 58                 <1>       pop    eax
00005BDB 1F                 <1>       pop    ds
00005BDC 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
00005BDD 51                 <1>       push   ecx
                            <1>
                            <1>       ; 29/05/2016
00005BDE 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:
00005BE3 B00A               <1>       mov    al, CMOS_REG_A            ; ADDRESS STATUS REGISTER A
00005BE5 FA                 <1>       cli                        ; NO TIMER INTERRUPTS DURING UPDATES
00005BE6 E80D000000         <1>       call   CMOS_READ           ; READ UPDATE IN PROCESS FLAG
00005BEB A880               <1>       test   al, 80h                   ; IF UIP BIT IS ON ( CANNOT READ TIME )
00005BED 7406               <1>       jz     short UPD_90        ; EXIT WITH CY= 0 IF CAN READ CLOCK NOW
00005BEF FB                 <1>       sti                        ; ALLOW INTERRUPTS WHILE WAITING
00005BF0 E2F1               <1>       loop   UPD_10              ; LOOP TILL READY OR TIMEOUT
00005BF2 31C0               <1>       xor    eax, eax            ; CLEAR RESULTS IF ERROR
                            <1>             ; xor ax, ax
00005BF4 F9                 <1>       stc                        ; SET CARRY FOR ERROR
                            <1> UPD_90:
00005BF5 59                 <1>       pop    ecx                 ; RESTORE CALLERS REGISTER
00005BF6 FA                 <1>       cli                        ; INTERRUPTS OFF DURING SET
00005BF7 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:
00005BF8 9C                 <1>       pushf                      ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
00005BF9 D0C0               <1>       rol    al, 1               ; MOVE NMI BIT TO LOW POSITION
00005BFB F9                 <1>       stc                        ; FORCE NMI BIT ON IN CARRY FLAG
00005BFC D0D8               <1>       rcr    al, 1               ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
00005BFE FA                 <1>       cli                        ; DISABLE INTERRUPTS
00005BFF E670               <1>       out    CMOS_PORT, al       ; ADDRESS LOCATION AND DISABLE NMI
                            <1>       ; 29/05/2016
                            <1>       ;nop                       ; I/O DELAY
00005C01 E6EB               <1>       out    0ebh,al      ; NEWIODELAY ; AWARD BIOS 1999, ATIME.ASM
                            <1>       ;
00005C03 E471               <1>       in     al, CMOS_DATA       ; READ THE REQUESTED CMOS LOCATION
00005C05 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)
00005C07 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
00005C09 D0D8               <1>       rcr    al, 1               ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
00005C0B E670               <1>       out    CMOS_PORT, al       ; SET DEFAULT TO READ ONLY REGISTER
00005C0D 6658               <1>       pop    ax                  ; RESTORE (AH) AND (AL), CMOS BYTE
00005C0F 9D                 <1>       popf
00005C10 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.      :
```

```
 539                            <1> ;           ONLY THE CMOS LOCATION AND THE NMI STATE IS CHANGED.       :
 540                            <1> ;--------------------------------------------------------------------------------
 541                            <1>
 542                            <1> CMOS_WRITE:                     ; WRITE (AH) TO LOCATION (AL)
 543 00005C11 9C                <1>       pushf                     ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
 544 00005C12 6650              <1>       push   ax                 ; SAVE WORK REGISTER VALUES
 545 00005C14 D0C0              <1>       rol    al, 1              ; MOVE NMI BIT TO LOW POSITION
 546 00005C16 F9                <1>       stc                       ; FORCE NMI BIT ON IN CARRY FLAG
 547 00005C17 D0D8              <1>       rcr    al, 1              ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
 548 00005C19 FA                <1>       cli                       ; DISABLE INTERRUPTS
 549 00005C1A E670              <1>       out    CMOS_PORT, al      ; ADDRESS LOCATION AND DISABLE NMI
 550 00005C1C 88E0              <1>       mov    al, ah             ; GET THE DATA BYTE TO WRITE
 551 00005C1E E671              <1>       out    CMOS_DATA, al      ; PLACE IN REQUESTED CMOS LOCATION
 552 00005C20 B01E              <1>       mov    al, CMOS_SHUT_DOWN*2      ; GET ADDRESS OF DEFAULT LOCATION
 553                            <1>       ;mov   al, CMOS_REG_D*2   ; GET ADDRESS OF DEFAULT LOCATION
 554 00005C22 D0D8              <1>       rcr    al, 1              ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
 555 00005C24 E670              <1>       out    CMOS_PORT, al      ; SET DEFAULT TO READ ONLY REGISTER
 556 00005C26 90                <1>       nop                       ; I/O DELAY
 557 00005C27 E471              <1>       in     al, CMOS_DATA      ; OPEN STANDBY LATCH
 558 00005C29 6658              <1>       pop    ax                 ; RESTORE WORK REGISTERS
 559 00005C2B 9D                <1>       popf
 560 00005C2C C3                <1>       retn
 561                            <1>
 562                            <1> ; /// End Of TIMER FUNCTIONS ///
2160
2161 00005C2D 90<rept>                  Align 16
2162
2163                                    gdt:  ; Global Descriptor Table
2164                                          ; (30/07/2015, conforming cs)
2165                                          ; (26/03/2015)
2166                                          ; (24/03/2015, tss)
2167                                          ; (19/03/2015)
2168                                          ; (29/12/2013)
2169                                          ;
2170 00005C30 0000000000000000               dw 0, 0, 0, 0       ; NULL descriptor
2171                                          ; 18/08/2014
2172                                                      ; 8h kernel code segment, base = 00000000h
2173                                          ;dw 0FFFFh, 0, 9E00h, 00CFh      ; KCODE  ; 30/12/2016
2174 00005C38 FFFF0000009ACF00               dw 0FFFFh, 0, 9A00h, 00CFh; KCODE
2175                                                      ; 10h kernel data segment, base = 00000000h
2176 00005C40 FFFF00000092CF00               dw 0FFFFh, 0, 9200h, 00CFh; KDATA
2177                                                      ; 1Bh user code segment, base address = 400000h ; CORE
2178                                          ;dw 0FBFFh, 0, 0FE40h, 00CFh     ; UCODE  ; 30/12/2016
2179 00005C48 FFFB000040FACF00               dw 0FBFFh, 0, 0FA40h, 00CFh      ; UCODE
2180                                                      ; 23h user data segment, base address = 400000h ; CORE
2181 00005C50 FFFB000040F2CF00               dw 0FBFFh, 0, 0F240h, 00CFh      ; UDATA
2182                                                      ; Task State Segment
2183 00005C58 6700                           dw 0067h ; Limit = 103 ; (104-1, tss size = 104 byte,
2184                                                               ;  no IO permission in ring 3)
2185                                    gdt_tss0:
2186 00005C5A 0000                           dw 0  ; TSS base address, bits 0-15
2187                                    gdt_tss1:
2188 00005C5C 00                             db 0  ; TSS base address, bits 16-23
2189                                                      ; 49h
2190 00005C5D E9                             db 11101001b ; E9h => P=1/DPL=11/0/1/0/B/1 --> B = Task is busy (1)
2191 00005C5E 00                             db 0 ; G/0/0/AVL/LIMIT=0000 ; (Limit bits 16-19 = 0000) (G=0, 1 byte)
2192                                    gdt_tss2:
2193 00005C5F 00                             db 0  ; TSS base address, bits 24-31
2194
2195                                    gdt_end:
2196                                          ;; 9Eh = 1001 1110b (GDT byte 5) P=1/DPL=00/1/TYPE=1110,
2197                                                                  ;; Type= 1 (code)/C=1/R=1/A=0
2198                                                ; P= Present, DPL=0=ring 0,  1= user (0= system)
2199                                                ; 1= Code C= Conforming, R= Readable, A = Accessed
2200
2201                                          ;; 9Ah = 1001 1010b (GDT byte 5) P=1/DPL=00/1/TYPE=1010,
2202                                                                  ;; Type= 1 (code)/C=0/R=1/A=0
2203                                                ; P= Present, DPL=0=ring 0,  1= user (0= system)
2204                                                ; 1= Code C= non-Conforming, R= Readable, A = Accessed
2205
2206                                          ;; 92h = 1001 0010b (GDT byte 5) P=1/DPL=00/1/TYPE=1010,
2207                                                                  ;; Type= 0 (data)/E=0/W=1/A=0
2208                                                ; P= Present, DPL=0=ring 0,  1= user (0= system)
2209                                                ; 0= Data E= Expansion direction (1= down, 0= up)
2210                                                ; W= Writeable, A= Accessed
2211
2212                                                ;; FEh = 1111 1110b (GDT byte 5) P=1/DPL=11/1/TYPE=1110,
2213                                                                  ;; Type= 1 (code)/C=1/R=1/A=0
2214                                                ; P= Present, DPL=3=ring 3,  1= user (0= system)
2215                                                ; 1= Code C= Conforming, R= Readable, A = Accessed
2216
2217                                                ;; FAh = 1111 1010b (GDT byte 5) P=1/DPL=11/1/TYPE=1010,
2218                                                                  ;; Type= 1 (code)/C=0/R=1/A=0
2219                                                ; P= Present, DPL=3=ring 3,  1= user (0= system)
2220                                                ; 1= Code C= non-Conforming, R= Readable, A = Accessed
2221
2222                                                ;; F2h = 1111 0010b (GDT byte 5) P=1/DPL=11/1/TYPE=0010,
2223                                                                  ;; Type= 0 (data)/E=0/W=1/A=0
2224                                                ; P= Present, DPL=3=ring 3,  1= user (0= system)
2225                                                ; 0= Data E= Expansion direction (1= down, 0= up)
2226
2227                                                ;; CFh = 1100 1111b (GDT byte 6) G=1/B=1/0/AVL=0, Limit=1111b (3)
2228
2229                                                ;; Limit = FFFFFh (=> FFFFFh+1= 100000h) // bits 0-15, 48-51 //
2230                                                ;       = 100000h * 1000h (G=1) = 4GB
2231                                                ;; Limit = FFBFFh (=> FFBFFh+1= FFC00h) // bits 0-15, 48-51 //
2232                                                ;       = FFC00h * 1000h (G=1) = 4GB - 4MB
2233                                                ; G= Granularity (1= 4KB), B= Big (32 bit),
2234                                                ; AVL= Available to programmers
2235
2236                                    gdtd:
2237 00005C60 2F00                             dw gdt_end - gdt - 1    ; Limit (size)
```

```
2238 00005C62 [305C0000]                    dd gdt                  ; Address of the GDT
2239
2240                                      ; 20/08/2014
2241                                idtd:
2242 00005C66 7F02                          dw idt_end - idt - 1    ; Limit (size)
2243 00005C68 [184F0100]                    dd idt                  ; Address of the IDT
2244
2245                                ; 20/02/2017
2246                                ;;; 11/03/2015
2247                                %include 'diskdata.s'     ; DISK (BIOS) DATA (initialized)
   1                            <1> ; ****************************************************************************
   2                            <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - diskdata.s
   3                            <1> ; ----------------------------------------------------------------------------
   4                            <1> ; Last Update: 24/01/2016
   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> ; diskdata.inc (11/03/2015)
  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 - DISKDATA.INC
  20                            <1> ; Last Modification: 11/03/2015
  21                            <1> ;     (Initialized Disk Parameters Data section for 'DISKIO.INC')
  22                            <1> ;
  23                            <1>
  24                            <1> ;---------------------------------------
  25                            <1> ;     80286 INTERRUPT LOCATIONS :
  26                            <1> ;     REFERENCED BY POST & BIOS :
  27                            <1> ;---------------------------------------
  28                            <1>
  29 00005C6C [CF5C0000]        <1> DISK_POINTER:      dd     MD_TBL6            ; Pointer to Diskette Parameter Table
  30                            <1>
  31                            <1> ; IBM PC-XT Model 286 source code ORGS.ASM (06/10/85) - 14/12/2014
  32                            <1> ;---------------------------------------------------------------
  33                            <1> ; DISK_BASE                                   :
  34                            <1> ;     THIS IS THE SET OF PARAMETERS REQUIRED FOR          :
  35                            <1> ;     DISKETTE OPERATION. THEY ARE POINTED AT BY THE             :
  36                            <1> ;     DATA VARIABLE @DISK_POINTER. TO MODIFY THE PARAMETERS,     :
  37                            <1> ;     BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT       :
  38                            <1> ;---------------------------------------------------------------
  39                            <1>
  40                            <1> ;DISK_BASE:
  41                            <1> ;     DB     11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
  42                            <1> ;     DB     2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
  43                            <1> ;     DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
  44                            <1> ;     DB     2            ; 512 BYTES/SECTOR
  45                            <1> ;     ;DB    15           ; EOT (LAST SECTOR ON TRACK)
  46                            <1> ;     db     18           ; (EOT for 1.44MB diskette)
  47                            <1> ;     DB     01BH         ; GAP LENGTH
  48                            <1> ;     DB     0FFH         ; DTL
  49                            <1> ;     ;DB    054H         ; GAP LENGTH FOR FORMAT
  50                            <1> ;     db     06ch         ; (for 1.44MB dsikette)
  51                            <1> ;     DB     0F6H         ; FILL BYTE FOR FORMAT
  52                            <1> ;     DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
  53                            <1> ;     DB     8            ; MOTOR START TIME (1/8 SECONDS)
  54                            <1>
  55                            <1> ;---------------------------------------
  56                            <1> ;     ROM BIOS DATA AREAS       :
  57                            <1> ;---------------------------------------
  58                            <1>
  59                            <1> ;DATA       SEGMENT AT 40H            ; ADDRESS= 0040:0000
  60                            <1>
  61                            <1> ;@EQUIP_FLAG DW    ?           ; INSTALLED HARDWARE FLAGS
  62                            <1>
  63                            <1> ;---------------------------------------
  64                            <1> ;     DISKETTE DATA AREAS       :
  65                            <1> ;---------------------------------------
  66                            <1>
  67                            <1> ;@SEEK_STATUS      DB     ?            ; DRIVE RECALIBRATION STATUS
  68                            <1> ;                                 ; BIT 3-0 = DRIVE 3-0 RECALIBRATION
  69                            <1> ;                                 ; BEFORE NEXT SEEK IF BIT IS = 0
  70                            <1> ;@MOTOR_STATUS     DB     ?            ; MOTOR STATUS
  71                            <1> ;                                 ; BIT 3-0 = DRIVE 3-0 CURRENTLY RUNNING
  72                            <1> ;                                 ; BIT 7 = CURRENT OPERATION IS A WRITE
  73                            <1> ;@MOTOR_COUNT      DB     ?            ; TIME OUT COUNTER FOR MOTOR(S) TURN OFF
  74                            <1> ;@DSKETTE_STATUS DB       ?            ; RETURN CODE STATUS BYTE
  75                            <1> ;                                 ; CMD_BLOCK  IN STACK FOR DISK OPERATION
  76                            <1> ;@NEC_STATUS DB     7 DUP(?)      ; STATUS BYTES FROM DISKETTE OPERATION
  77                            <1>
  78                            <1> ;---------------------------------------
  79                            <1> ;     POST AND BIOS WORK DATA AREA     :
  80                            <1> ;---------------------------------------
  81                            <1>
  82                            <1> ;@INTR_FLAG  DB     ?            ; FLAG INDICATING AN INTERRUPT HAPPENED
  83                            <1>
  84                            <1> ;---------------------------------------
  85                            <1> ;     TIMER DATA AREA           :
  86                            <1> ;---------------------------------------
  87                            <1>
  88                            <1> ; 17/12/2014  (IRQ 0 - INT 08H)
  89                            <1> ;TIMER_LOW   equ    46Ch          ; Timer ticks (counter)  @ 40h:006Ch
  90                            <1> ;TIMER_HIGH  equ    46Eh          ; (18.2 timer ticks per second)
  91                            <1> ;TIMER_OFL   equ    470h          ; Timer - 24 hours flag  @ 40h:0070h
  92                            <1>
```

```
                        <1> ;----------------------------------------
                        <1> ;     ADDITIONAL MEDIA DATA            :
                        <1> ;----------------------------------------
                        <1>
                        <1> ;@LASTRATE   DB     ?            ; LAST DISKETTE DATA RATE SELECTED
                        <1> ;@DSK_STATE  DB     ?            ; DRIVE 0 MEDIA STATE
                        <1> ;            DB     ?            ; DRIVE 1 MEDIA STATE
                        <1> ;            DB     ?            ; DRIVE 0 OPERATION START STATE
                        <1> ;            DB     ?            ; DRIVE 1 OPERATION START STATE
                        <1> ;@DSK_TRK    DB     ?            ; DRIVE 0 PRESENT CYLINDER
                        <1> ;            DB     ?            ; DRIVE 1 PRESENT CYLINDER
                        <1>
                        <1> ;DATA        ENDS                ; END OF BIOS DATA SEGMENT
                        <1>
                        <1> ;--------------------------------------------------------
                        <1> ;     DRIVE TYPE TABLE                        :
                        <1> ;--------------------------------------------------------
                        <1>              ; 16/02/2015 (unix386.s, 32 bit modifications)
                        <1> DR_TYPE:
00005C70 01             <1>              DB     01           ;DRIVE TYPE, MEDIA TABLE
                        <1>                 ;DW      MD_TBL1
00005C71 [8E5C0000]     <1>              dd     MD_TBL1
00005C75 82             <1>              DB     02+BIT7ON
                        <1>              ;DW      MD_TBL2
00005C76 [9B5C0000]     <1>                 dd      MD_TBL2
00005C7A 02             <1> DR_DEFAULT:  DB     02
                        <1>                 ;DW      MD_TBL3
00005C7B [A85C0000]     <1>              dd      MD_TBL3
00005C7F 03             <1>              DB     03
                        <1>                 ;DW      MD_TBL4
00005C80 [B55C0000]     <1>              dd      MD_TBL4
00005C84 84             <1>              DB     04+BIT7ON
                        <1>                 ;DW      MD_TBL5
00005C85 [C25C0000]     <1>              dd      MD_TBL5
00005C89 04             <1>              DB     04
                        <1>                 ;DW      MD_TBL6
00005C8A [CF5C0000]     <1>              dd      MD_TBL6
                        <1> DR_TYPE_E       equ $                   ; END OF TABLE
                        <1> ;DR_CNT            EQU    (DR_TYPE_E-DR_TYPE)/3
                        <1> DR_CNT             equ    (DR_TYPE_E-DR_TYPE)/5
                        <1> ;--------------------------------------------------------
                        <1> ;     MEDIA/DRIVE PARAMETER TABLES               :
                        <1> ;--------------------------------------------------------
                        <1> ;--------------------------------------------------------
                        <1> ;     360 KB MEDIA IN 360 KB DRIVE               :
                        <1> ;--------------------------------------------------------
                        <1> MD_TBL1:
00005C8E DF             <1>              DB     11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00005C8F 02             <1>              DB     2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00005C90 25             <1>              DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00005C91 02             <1>              DB     2            ; 512 BYTES/SECTOR
00005C92 09             <1>              DB     09           ; EOT (LAST SECTOR ON TRACK)
00005C93 2A             <1>              DB     02AH         ; GAP LENGTH
00005C94 FF             <1>              DB     0FFH         ; DTL
00005C95 50             <1>              DB     050H         ; GAP LENGTH FOR FORMAT
00005C96 F6             <1>              DB     0F6H         ; FILL BYTE FOR FORMAT
00005C97 0F             <1>              DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
00005C98 08             <1>              DB     8            ; MOTOR START TIME (1/8 SECONDS)
00005C99 27             <1>              DB     39           ; MAX. TRACK NUMBER
00005C9A 80             <1>              DB     RATE_250     ; DATA TRANSFER RATE
                        <1> ;--------------------------------------------------------
                        <1> ;     360 KB MEDIA IN 1.2 MB DRIVE               :
                        <1> ;--------------------------------------------------------
                        <1> MD_TBL2:
00005C9B DF             <1>              DB     11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00005C9C 02             <1>              DB     2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00005C9D 25             <1>              DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00005C9E 02             <1>              DB     2            ; 512 BYTES/SECTOR
00005C9F 09             <1>              DB     09           ; EOT (LAST SECTOR ON TRACK)
00005CA0 2A             <1>              DB     02AH         ; GAP LENGTH
00005CA1 FF             <1>              DB     0FFH         ; DTL
00005CA2 50             <1>              DB     050H         ; GAP LENGTH FOR FORMAT
00005CA3 F6             <1>              DB     0F6H         ; FILL BYTE FOR FORMAT
00005CA4 0F             <1>              DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
00005CA5 08             <1>              DB     8            ; MOTOR START TIME (1/8 SECONDS)
00005CA6 27             <1>              DB     39           ; MAX. TRACK NUMBER
00005CA7 40             <1>              DB     RATE_300     ; DATA TRANSFER RATE
                        <1> ;--------------------------------------------------------
                        <1> ;     1.2 MB MEDIA IN 1.2 MB DRIVE               :
                        <1> ;--------------------------------------------------------
                        <1> MD_TBL3:
00005CA8 DF             <1>              DB     11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00005CA9 02             <1>              DB     2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00005CAA 25             <1>              DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00005CAB 02             <1>              DB     2            ; 512 BYTES/SECTOR
00005CAC 0F             <1>              DB     15           ; EOT (LAST SECTOR ON TRACK)
00005CAD 1B             <1>              DB     01BH         ; GAP LENGTH
00005CAE FF             <1>              DB     0FFH         ; DTL
00005CAF 54             <1>              DB     054H         ; GAP LENGTH FOR FORMAT
00005CB0 F6             <1>              DB     0F6H         ; FILL BYTE FOR FORMAT
00005CB1 0F             <1>              DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
00005CB2 08             <1>              DB     8            ; MOTOR START TIME (1/8 SECONDS)
00005CB3 4F             <1>              DB     79           ; MAX. TRACK NUMBER
00005CB4 00             <1>              DB     RATE_500     ; DATA TRANSFER RATE
                        <1> ;--------------------------------------------------------
                        <1> ;     720 KB MEDIA IN 720 KB DRIVE               :
                        <1> ;--------------------------------------------------------
                        <1> MD_TBL4:
00005CB5 DF             <1>              DB     11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00005CB6 02             <1>              DB     2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00005CB7 25             <1>              DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00005CB8 02             <1>              DB     2            ; 512 BYTES/SECTOR
```

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

```
 297                                    <1>
 298                                    <1> ; 05/01/2015
 299 00005CEC 00                        <1> hf_m_s:         db      0     ; (0 = Master, 1 = Slave)
 300                                    <1>
 301                                    <1> ; ****************************************************************************
2248
2249 00005CED 90                            Align 2
2250
2251                                        ; 04/11/2014 (Retro UNIX 386 v1)
2252 00005CEE 0000                          mem_1m_1k:   dw 0  ; Number of contiguous KB between
2253                                                           ; 1 and 16 MB, max. 3C00h = 15 MB.
2254 00005CF0 0000                          mem_16m_64k: dw 0  ; Number of contiguous 64 KB blocks
2255                                                     ; between 16 MB and 4 GB.
2256
2257                                        ; 12/11/2014 (Retro UNIX 386 v1)
2258 00005CF2 00                            boot_drv:    db 0 ; boot drive number (physical)
2259                                        ; 24/11/2014
2260 00005CF3 00                            drv:       db 0
2261 00005CF4 00                            last_drv:    db 0 ; last hdd
2262 00005CF5 00                            hdc:         db 0  ; number of hard disk drives
2263                                                         ; (present/detected)
2264
2265                                        ; 24/11/2014 (Retro UNIX 386 v1)
2266                                        ; Physical drive type & flags
2267 00005CF6 00                            fd0_type:    db 0  ; floppy drive type
2268 00005CF7 00                            fd1_type:    db 0  ; 4 = 1.44 Mb, 80 track, 3.5" (18 spt)
2269                                                           ; 6 = 2.88 Mb, 80 track, 3.5" (36 spt)
2270                                                           ; 3 = 720 Kb, 80 track, 3.5" (9 spt)
2271                                                           ; 2 = 1.2 Mb, 80 track, 5.25" (15 spt)
2272                                                           ; 1 = 360 Kb, 40 track, 5.25" (9 spt)
2273 00005CF8 00                            hd0_type:    db 0  ; EDD status for hd0 (bit 7 = present flag)
2274 00005CF9 00                            hd1_type:    db 0  ; EDD status for hd1 (bit 7 = present flag)
2275 00005CFA 00                            hd2_type:    db 0  ; EDD status for hd2 (bit 7 = present flag)
2276 00005CFB 00                            hd3_type:    db 0  ; EDD status for hd3 (bit 7 = present flag)
2277                                                           ; bit 0 - Fixed disk access subset supported
2278                                                           ; bit 1 - Drive locking and ejecting
2279                                                           ; bit 2 - Enhanced disk drive support
2280                                                           ; bit 3 = Reserved (64 bit EDD support)
2281                                                           ; (If bit 0 is '1' Retro UNIX 386 v1
2282                                                           ; will interpret it as 'LBA ready'!)
2283
2284                                        ; 11/03/2015 - 10/07/2015
2285 00005CFC 00000000000000000-            drv.cylinders: dw 0,0,0,0,0,0,0,0
2285 00005D05 0000000000
2286 00005D0A 00000000000000000-            drv.heads:     dw 0,0,0,0,0,0,0,0
2286 00005D13 0000000000
2287 00005D18 00000000000000000-            drv.spt:       dw 0,0,0,0,0,0,0,0
2287 00005D21 0000000000
2288 00005D26 00000000000000000-            drv.size:      dd 0,0,0,0,0,0,0,0
2288 00005D2F 00000000000000000-
2288 00005D38 00000000000000000-
2288 00005D41 00
2289 00005D42 00000000000000                drv.status:    db 0,0,0,0,0,0,0,0
2290 00005D49 00000000000000                drv.error:     db 0,0,0,0,0,0,0,0
2291
2292                                        Align 2
2293
2294                                        ;;; 11/03/2015
2295                                        %include 'kybdata.s'      ; KEYBOARD (BIOS) DATA
   1                                    <1> ; ****************************************************************************
   2                                    <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - kybdata.s
   3                                    <1> ; ----------------------------------------------------------------------------
   4                                    <1> ; Last Update: 17/01/2016
   5                                    <1> ; ----------------------------------------------------------------------------
   6                                    <1> ; Beginning: 17/01/2016
   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> ; kybdata.inc (11/03/2015)
  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 - KYBDATA.INC
  20                                    <1> ; Last Modification: 11/03/2015
  21                                    <1> ;             (Data Section for 'KEYBOARD.INC')
  22                                    <1> ;
  23                                    <1> ; //////// KEYBOARD DATA ///////////////
  24                                    <1>
  25                                    <1> ; 05/12/2014
  26                                    <1> ; 04/12/2014 (derived from pc-xt-286 bios source code -1986-)
  27                                    <1> ; 03/06/86  KEYBOARD BIOS
  28                                    <1>
  29                                    <1> ;--------------------------------------------------------------------------------
  30                                    <1> ;     KEY IDENTIFICATION SCAN TABLES
  31                                    <1> ;--------------------------------------------------------------------------------
  32                                    <1>
  33                                    <1> ;-----       TABLES FOR ALT CASE ------------
  34                                    <1> ;-----       ALT-INPUT-TABLE
  35 00005D50 524F50514B                <1> K30:  db     82,79,80,81,75
  36 00005D55 4C4D474849                <1>       db     76,77,71,72,73            ; 10 NUMBER ON KEYPAD
  37                                    <1> ;-----       SUPER-SHIFT-TABLE
  38 00005D5A 101112131415              <1>       db     16,17,18,19,20,21   ; A-Z TYPEWRITER CHARS
  39 00005D60 161718191E1F              <1>       db     22,23,24,25,30,31
  40 00005D66 202122232425              <1>       db     32,33,34,35,36,37
  41 00005D6C 262C2D2E2F30              <1>       db     38,44,45,46,47,48
  42 00005D72 3132                      <1>       db     49,50
  43                                    <1>
```

```
                            <1> ;-----      TABLE OF SHIFT KEYS AND MASK VALUES
                            <1> ;-----      KEY_TABLE
00005D74 52                 <1> _K6:    db      INS_KEY                 ; INSERT KEY
00005D75 3A4546381D         <1>         db  CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
00005D7A 2A36               <1>         db      LEFT_KEY,RIGHT_KEY
                            <1> _K6L    equ     $-_K6
                            <1>
                            <1> ;-----      MASK_TABLE
00005D7C 80                 <1> _K7:    db      INS_SHIFT               ; INSERT MODE SHIFT
00005D7D 4020100804         <1>         db  CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
00005D82 0201               <1>         db  LEFT_SHIFT,RIGHT_SHIFT
                            <1>
                            <1> ;-----      TABLES FOR CTRL CASE            ;---- CHARACTERS ------
00005D84 1BFF00FFFFFF       <1> _K8:  db    27,-1,0,-1,-1,-1    ; Esc, 1, 2, 3, 4, 5
00005D8A 1EFFFFFFFF1F       <1>       db    30,-1,-1,-1,-1,31   ; 6, 7, 8, 9, 0, -
00005D90 FF7FFF111705       <1>       db    -1,127,-1,17,23,5   ; =, Bksp, Tab, Q, W, E
00005D96 12141915090F       <1>       db    18,20,25,21,9,15    ; R, T, Y, U, I, O
00005D9C 101B1D0AFF01       <1>       db    16,27,29,10,-1,1    ; P, [, ], Enter, Ctrl, A
00005DA2 13040607080A       <1>       db    19,4,6,7,8,10       ; S, D, F, G, H, J
00005DA8 0B0CFFFFFFFF       <1>       db    11,12,-1,-1,-1,-1   ; K, L, :, ', `, LShift
00005DAE 1C1A18031602       <1>       db    28,26,24,3,22,2           ; Bkslash, Z, X, C, V, B
00005DB4 0E0DFFFFFFFF       <1>       db    14,13,-1,-1,-1,-1   ; N, M, ,, ., /, RShift
00005DBA 96FF20FF           <1>       db    150,-1,' ',-1       ; *, ALT, Spc, CL
                            <1>       ;                         ;----- FUNCTIONS ------
00005DBE 5E5F60616263       <1>       db    94,95,96,97,98,99   ; F1 - F6
00005DC4 64656667FFFF       <1>       db    100,101,102,103,-1,-1     ; F7 - F10, NL, SL
00005DCA 778D848E738F       <1>       db    119,141,132,142,115,143   ; Home, Up, PgUp, -, Left, Pad5
00005DD0 749075917692       <1>       db    116,144,117,145,118,146 ; Right, +, End, Down, PgDn, Ins
00005DD6 93FFFFFF898A       <1>       db    147,-1,-1,-1,137,138      ; Del, SysReq, Undef, WT, F11, F12
                            <1>
                            <1> ;-----      TABLES FOR LOWER CASE ----------
00005DDC 1B3132333435363738- <1> K10:  db    27,'1234567890-=',8,9
00005DE5 39302D3D0809       <1>
00005DEB 71776572747975696F- <1>       db    'qwertyuiop[]',13,-1,'asdfghjkl;',39
00005DF4 705B5D0DFF61736466- <1>
00005DFD 67686A6B6C3B27     <1>
00005E04 60FF5C7A786376626E- <1>       db    96,-1,92,'zxcvbnm,./',-1,'*',-1,' ',-1
00005E0D 6D2C2E2FFF2AFF20FF  <1>
                            <1> ;-----      LC TABLE SCAN
00005E16 3B3C3D3E3F         <1>       db    59,60,61,62,63            ; BASE STATE OF F1 - F10
00005E1B 4041424344         <1>       db    64,65,66,67,68
00005E20 FFFF               <1>       db    -1,-1               ; NL, SL
                            <1>
                            <1> ;-----      KEYPAD TABLE
00005E22 474849FF4BFF       <1> K15:  db    71,72,73,-1,75,-1   ; BASE STATE OF KEYPAD KEYS
00005E28 4DFF4F50515253     <1>       db    77,-1,79,80,81,82,83
00005E2F FFFF5C8586         <1>       db    -1,-1,92,133,134    ; SysRq, Undef, WT, F11, F12
                            <1>
                            <1> ;-----      TABLES FOR UPPER CASE ----------
00005E34 1B21402324255E262A- <1> K11:  db    27,'!@#$%',94,'&*()_+',8,0
00005E3D 28295F2B0800       <1>
00005E43 51574552545955494F- <1>       db    'QWERTYUIOP{}',13,-1,'ASDFGHJKL:"'
00005E4C 507B7D0DFF41534446- <1>
00005E55 47484A4B4C3A22     <1>
00005E5C 7EFF7C5A584356424E- <1>       db    126,-1,'|ZXCVBNM<>?',-1,'*',-1,' ',-1
00005E65 4D3C3E3FFF2AFF20FF  <1>
                            <1> ;-----      UC TABLE SCAN
00005E6E 5455565758         <1> K12:  db    84,85,86,87,88            ; SHIFTED STATE OF F1 - F10
00005E73 595A5B5C5D         <1>       db    89,90,91,92,93
00005E78 FFFF               <1>       db    -1,-1               ; NL, SL
                            <1>
                            <1> ;-----      NUM STATE TABLE
00005E7A 3738392D3435362B31- <1> K14:  db    '789-456+1230.'           ; NUMLOCK STATE OF KEYPAD KEYS
00005E83 3233302E           <1>
                            <1>       ;
00005E87 FFFF7C8788         <1>       db    -1,-1,124,135,136   ; SysRq, Undef, WT, F11, F12
                            <1>
                            <1> ; 26/08/2014
                            <1> ; Retro UNIX 8086 v1 - UNIX.ASM (03/03/2014)
                            <1> ; Derived from IBM "pc-at"
                            <1> ; rombios source code (06/10/1985)
                            <1> ; 'dseg.inc'
                            <1>
                            <1> ;--------------------------------------;
                            <1> ;     SYSTEM DATA AREA          ;
                            <1> ;--------------------------------------
00005E8C 00                 <1> BIOS_BREAK   db     0            ; BIT 7=1 IF BREAK KEY HAS BEEN PRESSED
                            <1>
                            <1> ;--------------------------------------
                            <1> ;     KEYBOARD DATA AREAS       ;
                            <1> ;--------------------------------------
                            <1>
00005E8D 00                 <1> KB_FLAG             db     0            ; KEYBOARD SHIFT STATE AND STATUS FLAGS
00005E8E 00                 <1> KB_FLAG_1    db     0            ; SECOND BYTE OF KEYBOARD STATUS
00005E8F 00                 <1> KB_FLAG_2    db     0            ; KEYBOARD LED FLAGS
00005E90 00                 <1> KB_FLAG_3    db     0            ; KEYBOARD MODE STATE AND TYPE FLAGS
00005E91 00                 <1> ALT_INPUT    db     0            ; STORAGE FOR ALTERNATE KEY PAD ENTRY
00005E92 [A25E0000]         <1> BUFFER_START dd     KB_BUFFER    ; OFFSET OF KEYBOARD BUFFER START
00005E96 [C25E0000]         <1> BUFFER_END   dd     KB_BUFFER + 32      ; OFFSET OF END OF BUFFER
00005E9A [A25E0000]         <1> BUFFER_HEAD  dd     KB_BUFFER    ; POINTER TO HEAD OF KEYBOARD BUFFER
00005E9E [A25E0000]         <1> BUFFER_TAIL  dd     KB_BUFFER    ; POINTER TO TAIL OF KEYBOARD BUFFER
                            <1> ; ------     HEAD = TAIL  INDICATES THAT THE BUFFER IS EMPTY
00005EA2 0000<rept>         <1> KB_BUFFER    times  16 dw 0             ; ROOM FOR 16 SCAN CODE ENTRIES
                            <1>
                            <1> ; /// End Of KEYBOARD DATA ///
                                %include 'vidata.s'       ; VIDEO (BIOS) DATA
                            <1> ; ********************************************************************************
                            <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - vidata.s
                            <1> ; --------------------------------------------------------------------------------
                            <1> ; Last Update: 31/07/2016
                            <1> ; --------------------------------------------------------------------------------
                            <1> ; Beginning: 16/01/2016
```

```
                        <1> ; ----------------------------------------------------------------------------
                        <1> ; Assembler: NASM version 2.11 (trdos386.s)
                        <1> ; ----------------------------------------------------------------------------
                        <1> ; Turkish Rational DOS
                        <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
                        <1> ;
                        <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
                        <1> ; vidata.inc (11/03/2015)
                        <1> ;
                        <1> ; Derived from 'IBM PC-AT' BIOS source code (1985)
                        <1> ; ****************************************************************************
                        <1>
                        <1> ; Retro UNIX 386 v1 Kernel - VIDATA.S
                        <1> ; Last Modification: 11/03/2015
                        <1> ;               (Data section for 'VIDEO.INC')
                        <1> ;
                        <1> ; ///////// VIDEO DATA ///////////////
                        <1>
                        <1> ;----------------------------------------
                        <1> ;     VIDEO DISPLAY DATA AREA          ;
                        <1> ;----------------------------------------
00005EC2 03             <1> CRT_MODE:    db     3      ; CURRENT DISPLAY MODE (TYPE)
00005EC3 29             <1> CRT_MODE_SET:      db     29h    ; CURRENT SETTING OF THE 3X8 REGISTER
                        <1>                            ; (29h default setting for video mode 3)
                        <1>                            ; Mode Select register Bits
                        <1>                            ;   BIT 0 - 80x25 (1), 40x25 (0)
                        <1>                            ;   BIT 1 - ALPHA (0), 320x200 GRAPHICS (1)
                        <1>                            ;   BIT 2 - COLOR (0), BW (1)
                        <1>                            ;   BIT 3 - Video Sig. ENABLE (1), DISABLE (0)
                        <1>                            ;   BIT 4 - 640x200 B&W Graphics Mode (1)
                        <1>                            ;   BIT 5 - ALPHA mode BLINKING (1)
                        <1>                            ;   BIT 6, 7 - Not Used
                        <1>
                        <1> ; Mode 0 - 2Ch = 101100b  ; 40x25 text, 16 gray colors
                        <1> ; Mode 1 - 28h = 101000b  ; 40x25 text, 16 fore colors, 8 back colors
                        <1> ; Mode 2 - 2Dh = 101101b  ; 80x25 text, 16 gray colors
                        <1> ; Mode 3 - 29h = 101001b  ; 80x25 text, 16 fore color, 8 back color
                        <1> ; Mode 4 - 2Ah = 101010b  ; 320x200 graphics, 4 colors
                        <1> ; Mode 5 - 2Eh = 101110b  ; 320x200 graphics, 4 gray colors
                        <1> ; Mode 6 - 1Eh = 011110b  ; 640x200 graphics, 2 colors
                        <1> ; Mode 7 - 29h = 101001b  ; 80x25 text, black & white colors
                        <1> ; Mode & 37h = Video signal OFF
                        <1>
                        <1> ; 24/06/2016
00005EC4 50             <1> CRT_COLS:    db     80     ; Number of columns
                        <1>
                        <1> ; 01/07/2016
00005EC5 00             <1> CRT_PALETTE: db     0      ; Current palette setting
                        <1>
                        <1> ; 03/07/2016
00005EC6 10             <1> CHAR_HEIGHT: db     16     ; Default character height
00005EC7 60             <1> VGA_VIDEO_CTL:     db     60h    ; ROM BIOS DATA AREA Offset 87h
00005EC8 F9             <1> VGA_SWITCHES:      db     0F9h   ; Feature Bit Switches (the basic screen)
00005EC9 51             <1> VGA_MODESET_CTL: db        051h   ; Basic mode set options (VGA video flags)
                        <1>                            ; ROM BIOS DATA AREA Offset 89h
                        <1>                            ; Bit 7, 4 : Mode
                        <1>                            ;       01 : 400-line mode
                        <1>                            ; Bit 6      : Display switch enabled =  1
                        <1>                            ; Bit 5      : Reserved  = 0
                        <1>                            ; Bit 3      : Default palette loading
                        <1>                            ;       disabled = 0
                        <1>                            ; Bit 2 : Color monitor = 0
                        <1>                            ; Bit 1 = Gray scale summing
                        <1>                            ;       disabled = 0
                        <1>                            ; Bit 0 = VGA active = 1
00005ECA 19             <1> VGA_ROWS:    db     25
                        <1>
                        <1> ; 16/01/2016
                        <1> chr_attrib:  ; Character color/attributes for viode pages (0 to 7)
00005ECB 0707070707070707   <1>       db     07h, 07h, 07h, 07h, 07h, 07h, 07h, 07h
                        <1> ; 30/01/2016
                        <1> vmode:
00005ED3 0303030303030303   <1>       db     3,3,3,3,3,3,3,3 ; video modes for pseudo screens
                        <1>
                        <1> CURSOR_MODE: ; cursor start (ch) = 14, cursor end (cl) = 15
00005EDB 0F0E           <1>       db     15, 14 ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
                        <1>
                        <1> ;align 4
                        <1> ;VGA_BASE: ; 26/07/2016
                        <1> ;     dd      0B8000h  ; (Mode < 0Dh) or 0A0000h (mode >= 0Dh)
                        <1>
00005EDD 90             <1> align 2
                        <1>
                        <1> vga_modes:
                        <1>       ; 25/07/2016
                        <1>       ; 09/07/2016
                        <1>       ; 03/07/2016
                        <1>       ; valid (implemented) video modes (>7, extension to IBM PC CGA modes)
00005EDE 0302010007040506   <1>       db     03h, 02h, 01h, 00h, 07h, 04h, 05h, 06h
                        <1> vga_g_modes: ; 31/07/2016
00005EE6 13F0126A0D0E1011   <1>       db     13h, 0F0h, 12h, 6Ah, 0Dh, 0Eh, 10h, 11h
                        <1> vga_mode_count equ $ - vga_modes
                        <1> vga_g_mode_count equ $ - vga_g_modes
                        <1>
                        <1> vga_mode_tbl_ptr:
                        <1>       ; 25/07/2016
00005EEE [4E5F0000]     <1>       dd     vga_mode_03h
00005EF2 [4E5F0000]     <1>       dd     vga_mode_03h ; mode 02h -> mode 03h
00005EF6 [8E5F0000]     <1>       dd     vga_mode_01h
00005EFA [8E5F0000]     <1>       dd     vga_mode_01h ; mode 00h -> mode 01h
                        <1>       ;dd    vga_mode_07h
00005EFE [4E5F0000]     <1>       dd     vga_mode_03h ; mode 07h -> mode 03h
```

```
00005F02 [CE5F0000]          <1>       dd    vga_mode_04h
00005F06 [CE5F0000]          <1>       dd    vga_mode_04h ; mode 05h -> mode 04h
00005F0A [0E600000]          <1>       dd    vga_mode_06h
00005F0E [4E600000]          <1>       dd    vga_mode_13h
00005F12 [8E600000]          <1>       dd    vga_mode_F0h
00005F16 [CE600000]          <1>       dd    vga_mode_12h
00005F1A [0E610000]          <1>       dd    vga_mode_6Ah
00005F1E [4E610000]          <1>       dd    vga_mode_0Dh
00005F22 [8E610000]          <1>       dd    vga_mode_0Eh
00005F26 [CE610000]          <1>       dd    vga_mode_10h
00005F2A [0E620000]          <1>       dd    vga_mode_11h
                             <1>
                             <1> vga_memmodel:
                             <1>       ; 25/07/2016
                             <1>       ; 07/07/2016
                             <1>       CTEXT  equ 0
                             <1>       ;MTEXT equ 1
                             <1>       MTEXT  equ 0 ; mode 07h -> mode 03h
                             <1>       CGA    equ 2
                             <1>       LINEAR8 equ 5
                             <1>       PLANAR4      equ 4
                             <1>       PLANAR1      equ 3
00005F2E 0000000000020202    <1>       db    CTEXT, CTEXT, CTEXT, CTEXT, MTEXT, CGA, CGA, CGA
                             <1> vga_g_memmodel: ; 31/07/2016
00005F36 0504040404040403    <1>       db    LINEAR8, PLANAR4, PLANAR4, PLANAR4, PLANAR4, PLANAR4, PLANAR4, PLANAR1
                             <1> ;vga_pixbits:
                             <1> ;     ; 25/07/2016
                             <1> ;     ; 08/07/2016
                             <1> ;     db    4, 4, 4, 4, 4, 2, 2, 1, 8, 4, 4, 4, 4, 4, 4, 1
                             <1> vga_dac_s:
00005F3E 020202020001010103- <1>       db    2, 2, 2, 2, 0, 1, 1, 1, 3, 3, 2, 2, 1, 1, 2, 2
00005F47 03020201010202      <1>
                             <1>
                             <1> vga_params:
                             <1>       ; 25/07/2016
                             <1>       ; 19/07/2016
                             <1>       ; 03/07/2016
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios Developers Team (2001-2008)
                             <1>       ; 'vgatables.h'
                             <1>       ; Oracle VirtualBox 5.0.24 VGABios Source Code
                             <1>       ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
                             <1>       ;
                             <1> vga_mode_03h:  ; mode 03h, 80*25 text, CGA colors
00005F4E 5018100010          <1>       db    80, 24, 16, 00h, 10h ; tw, th-1, ch, slength (5)
00005F53 00030002            <1>       db    00h, 03h, 00h, 02h ; sequ regs (4)
00005F57 67                  <1>       db    67h    ; misc reg (1)
00005F58 5F4F50825581BF1F    <1>       db    5Fh, 4Fh, 50h, 82h, 55h, 81h, 0BFh, 1Fh
00005F60 004F                <1>       db    00h, 4Fh
                             <1> vga_p_cm_pos equ $ - vga_mode_03h
00005F62 0D0E00000000        <1>       db    0Dh, 0Eh, 00h, 00h, 00h, 00h
00005F68 9C8E8F281F96B9A3    <1>       db    9Ch, 8Eh, 8Fh, 28h, 1Fh, 96h, 0B9h, 0A3h
00005F70 FF                  <1>       db    0FFh   ; crtc_regs (25)
00005F71 0001020304051407    <1>       db    00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
00005F79 38393A3B3C3D3E3F    <1>       db    38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
00005F81 0C000F08            <1>       db    0Ch, 00h, 0Fh, 08h  ; actl regs (20)
00005F85 0000000000100E0FFF  <1>       db    00h, 00h, 00h, 00h, 00h, 10h, 0Eh, 0Fh, 0FFh ; grdc regs (9)
                             <1> vga_mode_01h:      ; mode 01h, 40*25 text, CGA colors
00005F8E 2818100008          <1>       db    40, 24, 16, 00h, 08h ; tw, th-1, ch, slength
00005F93 08030002            <1>       db    08h, 03h, 00h, 02h  ; sequ regs
00005F97 67                  <1>       db    67h    ; misc reg
00005F98 2D2728902BA0BF1F    <1>       db    2Dh, 27h, 28h, 90h, 2Bh, 0A0h, 0BFh, 1Fh
00005FA0 004F0D0E00000000    <1>       db    00h, 4Fh, 0Dh, 0Eh, 00h, 00h, 00h, 00h
00005FA8 9C8E8F141F96B9A3    <1>       db    9Ch, 8Eh, 8Fh, 14h, 1Fh, 96h, 0B9h, 0A3h
00005FB0 FF                  <1>       db    0FFh   ; crtc_regs
00005FB1 0001020304051407    <1>       db    00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
00005FB9 38393A3B3C3D3E3F    <1>       db    38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
00005FC1 0C000F08            <1>       db    0Ch, 00h, 0Fh, 08h  ; actl regs
00005FC5 0000000000100E0FFF  <1>       db    00h, 00h, 00h, 00h, 00h, 10h, 0Eh, 0Fh, 0FFh ; grdc regs
                             <1> ;vga_mode_07h:     ; mode 07h, 80*25 text, mono color
                             <1> ;     db    80, 24, 16, 00h, 10h  ; tw, th-1, ch, slength
                             <1> ;     db    00h, 03h, 00h, 02h ; sequ regs
                             <1> ;     db    66h    ; misc reg
                             <1> ;     db    5Fh, 4Fh, 50h, 82h, 55h, 81h, 0BFh, 1Fh
                             <1> ;     db    00h, 4Fh, 0Dh, 0Eh, 00h, 00h, 00h, 00h
                             <1> ;     db    9Ch, 8Eh, 8Fh, 28h, 0Fh, 96h, 0B9h, 0A3h
                             <1> ;     db    0FFh   ; crtc regs
                             <1> ;     db    00h, 08h, 08h, 08h, 08h, 08h, 08h, 08h
                             <1> ;     db    10h, 18h, 18h, 18h, 18h, 18h, 18h, 18h
                             <1> ;     db    0Eh, 00h, 0Fh, 08h  ; actl regs
                             <1> ;     db    00h, 00h, 00h, 00h, 00h, 10h, 0Ah, 0Fh, 0FFh ; grdc regs
                             <1> vga_mode_04h:      ; 320*200 graphics, 4 colors, CGA
00005FCE 2818080008          <1>       db    40, 24, 8, 00h, 08h   ; tw, th-1, ch, slength
00005FD3 09030002            <1>       db    09h, 03h, 00h, 02h ; sequ regs
00005FD7 63                  <1>       db    63h    ; misc reg
00005FD8 2D2728902B80BF1F    <1>       db    2Dh, 27h, 28h, 90h, 2Bh, 80h, 0BFh, 1Fh
00005FE0 00C1000000000000    <1>       db    00h, 0C1h, 00h, 00h, 00h, 00h, 00h, 00h
00005FE8 9C8E8F140096B9A2    <1>       db    9Ch, 8Eh, 8Fh, 14h, 00h, 96h, 0B9h, 0A2h
00005FF0 FF                  <1>       db    0FFh   ; crtc_regs
00005FF1 0013151702040607    <1>       db    00h, 13h, 15h, 17h, 02h, 04h, 06h, 07h
00005FF9 1011121314151617    <1>       db    10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h
00006001 01000300            <1>       db    01h, 00h, 03h, 00h ; actl regs
00006005 0000000000300F0FFF  <1>       db    00h, 00h, 00h, 00h, 00h, 30h, 0Fh, 0Fh, 0FFh ; grdc regs
                             <1> vga_mode_06h:      ; 640*200 graphics, 2 colors, CGA
0000600E 5018080010          <1>       db    80, 24, 8, 00h, 10h   ;   tw, th-1, ch, slength
00006013 01010006            <1>       db    01h, 01h, 00h, 06h ; sequ regs
00006017 63                  <1>       db    63h    ; misc reg
00006018 5F4F50825480BF1F    <1>       db    5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
00006020 00C1000000000000    <1>       db    00h, 0C1h, 00h, 00h, 00h, 00h, 00h, 00h
00006028 9C8E8F280096B9C2    <1>       db    9Ch, 8Eh, 8Fh, 28h, 00h, 96h, 0B9h, 0C2h
00006030 FF                  <1>       db    0FFh   ; crtc regs
```

```
00006031 0017171717171717    <1>       db    00h, 17h, 17h, 17h, 17h, 17h, 17h, 17h
00006039 1717171717171717    <1>       db    17h, 17h, 17h, 17h, 17h, 17h, 17h, 17h
00006041 01000100             <1>       db    01h, 00h, 01, 00h  ; actl regs
00006045 0000000000000D0FFF   <1>       db    00h, 00h, 00h, 00h, 00h, 00h, 0Dh, 0Fh, 0FFh ; grdc regs
                              <1> vga_mode_13h:  ; mode 13h, 300*200, 256 colors, linear
0000604E 2818080000           <1>       db    40, 24, 8, 0, 0     ; tw, th-1, ch, slength (5)
00006053 010F000E             <1>       db    01h, 0Fh, 00h, 0Eh ; sequ regs (4)
00006057 63                   <1>       db    63h    ; misc reg (1)
00006058 5F4F50825480BF1F     <1>       db    5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
00006060 0041000000000000     <1>       db    00h, 41h, 00h, 00h, 00h, 00h, 00h, 00h
00006068 9C8E8F284096B9A3     <1>       db    9Ch, 8Eh, 8Fh, 28h, 40h, 96h, 0B9h, 0A3h
00006070 FF                   <1>       db    0FFh   ; crtc regs (25)
00006071 0001020304050607     <1>       db    00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
00006079 08090A0B0C0D0E0F     <1>       db    08h, 09h, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh, 0Fh
00006081 41000F00             <1>       db    41h, 00h, 0Fh, 00h  ; actl regs (20)
00006085 000000000040050FFF   <1>       db    00h, 00h, 00h, 00h, 00h, 40h, 05h, 0Fh, 0FFh ; grdc regs (9)
                              <1> vga_mode_setl equ $ - vga_mode_13h  ; = 64
                              <1> vga_mode_F0h:  ; mode X ; 320*240, 256 colors, planar
0000608E 2818080000           <1>       db    40, 24, 8, 0, 0     ; tw, th-1, ch, slength
00006093 010F0006             <1>       db    01h, 0Fh, 00h, 06h ; sequ regs
00006097 E3                   <1>       db    0E3h   ; misc reg
00006098 5F4F508254800D3E     <1>       db    5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Dh, 3Eh
000060A0 0041000000000000     <1>       db    00h, 41h, 00h, 00h, 00h, 00h, 00h, 00h
000060A8 EAACDF2800E706E3     <1>       db    0EAh, 0ACh, 0DFh, 28h, 00h, 0E7h, 06h, 0E3h
000060B0 FF                   <1>       db    0FFh   ; crtc regs (25)
000060B1 0001020304050607     <1>       db    00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
000060B9 08090A0B0C0D0E0F     <1>       db    08h, 09h, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh, 0Fh
000060C1 41000F00             <1>       db    41h, 00h, 0Fh, 00h  ; actl regs
000060C5 000000000040050FFF   <1>       db    00h, 00h, 00h, 00h, 00h, 40h, 05h, 0Fh, 0FFh ; grdc regs
                              <1> vga_mode_12h:  ; mode 12h, 640*480, 16 colors, planar
000060CE 501D100000           <1>       db    80, 29, 16, 0, 0 ; tw, th-1, ch, slength
000060D3 010F0006             <1>       db    01h, 0Fh, 00h, 06h ; sequ regs
000060D7 E3                   <1>       db    0E3h   ; misc reg
000060D8 5F4F508254800B3E     <1>       db    5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Bh, 3Eh
000060E0 0040000000000000     <1>       db    00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h
000060E8 EA8CDF2800E704E3     <1>       db    0EAh, 8Ch, 0DFh, 28h, 00h, 0E7h, 04h, 0E3h
000060F0 FF                   <1>       db    0FFh   ; crtc regs
000060F1 0001020304051407     <1>       db    00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
000060F9 38393A3B3C3D3E3F     <1>       db    38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
00006101 01000F00             <1>       db    01h, 00h, 0Fh, 00h  ; actl regs
00006105 000000000000050FFF   <1>       db    00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
                              <1> vga_mode_6Ah:  ; mode 6Ah, 800*600, 16 colors, planar
0000610E 6424100000           <1>       db    100, 36, 16, 0, 0 ; tw, th-1, ch, slength
00006113 010F0006             <1>       db    01h, 0Fh, 00h, 06h ; sequ regs
00006117 E3                   <1>       db    0E3h   ; misc reg
00006118 7F6363836B1B72F0     <1>       db    7Fh, 63h, 63h, 83h, 6Bh, 1Bh, 72h, 0F0h
00006120 0060000000000000     <1>       db    00h, 60h, 00h, 00h, 00h, 00h, 00h, 00h
00006128 598D5732005773E3     <1>       db    59h, 8Dh, 57h, 32h, 00h, 57h, 73h, 0E3h
00006130 FF                   <1>       db    0FFh   ; crtc regs
00006131 0001020304051407     <1>       db    00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
00006139 38393A3B3C3D3E3F     <1>       db    38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
00006141 01000F00             <1>       db    01h, 00h, 0Fh, 00h  ; actl regs
00006145 000000000000050FFF   <1>       db    00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
                              <1> vga_mode_0Dh:  ; mode 0Dh, 320*200, 16 colors, planar
0000614E 2818080020           <1>       db    40, 24, 8, 0, 20h ; tw, th-1, ch, slength
00006153 090F0006             <1>       db    09h, 0Fh, 00h, 06h ; sequ regs
00006157 63                   <1>       db    63h    ; misc reg
00006158 2D2728902B80BF1F     <1>       db    2Dh, 27h, 28h, 90h, 2Bh, 80h, 0BFh, 1Fh
00006160 00C0000000000000     <1>       db    00h, 0C0h, 00h, 00h, 00h, 00h, 00h, 00h
00006168 9C8E8F140096B9E3     <1>       db    9Ch, 8Eh, 8Fh, 14h, 00h, 96h, 0B9h, 0E3h
00006170 FF                   <1>       db    0FFh   ; crtc regs
00006171 0001020304050607     <1>       db    00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
00006179 1011121314151617     <1>       db    10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h
00006181 01000F00             <1>       db    01h, 00h, 0Fh, 00h  ; actl regs
00006185 000000000000050FFF   <1>       db    00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
                              <1> vga_mode_0Eh:  ; mode 0Eh, 640*200, 16 colors, planar
0000618E 5018080040           <1>       db    80, 24, 8, 0, 40h ; tw, th-1, ch, slength
00006193 010F0006             <1>       db    01h, 0Fh, 00h, 06h ; sequ regs
00006197 63                   <1>       db    63h    ; misc reg
00006198 5F4F50825480BF1F     <1>       db    5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
000061A0 00C0000000000000     <1>       db    00h, 0C0h, 00h, 00h, 00h, 00h, 00h, 00h
000061A8 9C8E8F280096B9E3     <1>       db    9Ch, 8Eh, 8Fh, 28h, 00h, 96h, 0B9h, 0E3h
000061B0 FF                   <1>       db    0FFh   ; crtc regs
000061B1 0001020304050607     <1>       db    00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
000061B9 1011121314151617     <1>       db    10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h
000061C1 01000F00             <1>       db    01h, 00h, 0Fh, 00h  ; actl regs
000061C5 000000000000050FFF   <1>       db    00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
                              <1> vga_mode_10h: ; mode 10h, 640*350, 16 colors, planar
000061CE 50180E0080           <1>       db    80, 24, 14, 0, 80h ; tw, th-1, ch, slength
000061D3 010F0006             <1>       db    01h, 0Fh, 00h, 06h ; sequ regs
000061D7 A3                   <1>       db    0A3h   ; misc reg
000061D8 5F4F50825480BF1F     <1>       db    5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
000061E0 0040000000000000     <1>       db    00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h
000061E8 83855D280F63BAE3     <1>       db    83h, 85h, 5Dh, 28h, 0Fh, 63h, 0BAh, 0E3h
000061F0 FF                   <1>       db    0FFh   ; crtc regs
000061F1 0001020304051407     <1>       db    00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
000061F9 38393A3B3C3D3E3F     <1>       db    38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
00006201 01000F00             <1>       db    01h, 00h, 0Fh, 00h  ; actl regs
00006205 000000000000050FFF   <1>       db    00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
                              <1> vga_mode_11h: ; mode 11h, 640*480, mono color, planar
0000620E 501D100000           <1>       db    80, 29, 16, 0, 0   ; tw, th-1, ch, slength
00006213 010F0006             <1>       db    01h, 0Fh, 00h, 06h ; sequ regs
00006217 E3                   <1>       db    0E3h   ; misc reg
00006218 5F4F508254800B3E     <1>       db    5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Bh, 3Eh
00006220 0040000000000000     <1>       db    00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h
00006228 EA8CDF2800E704E3     <1>       db    0EAh, 8Ch, 0DFh, 28h, 00h, 0E7h, 04h, 0E3h
00006230 FF                   <1>       db    0FFh   ; crtc regs
00006231 003F003F003F003F     <1>       db    00h, 3Fh, 00h, 3Fh, 00h, 3Fh, 00h, 3Fh
00006239 003F003F003F003F     <1>       db    00h, 3Fh, 00h, 3Fh, 00h, 3Fh, 00h, 3Fh
00006241 01000F00             <1>       db    01h, 00h, 0Fh, 00h  ; actl regs
00006245 000000000000050FFF   <1>       db    00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
                              <1> end_of_vga_params:
```

```
 312                                    <1>
 313                                    <1> ; /// End Of VIDEO DATA ///
2297                                        ;%include 'diskdata.s'    ; DISK (BIOS) DATA (initialized)
2298                                        ;;;
2299
2300                                        Align 2
2301
2302                                        %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: 28/08/2017
   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> ; 19/02/2017
  79                                    <1> ; 15/10/2016
  80                                    <1> ; 20/05/2016
  81                                    <1> ; 19/05/2016
  82                                    <1> ; 18/05/2016
  83                                    <1> ; 29/04/2016
  84                                    <1> ; TRDOS 386 (TRDOS v2.0) system calls - temporary List
  85                                    <1> ; 14/07/2013 - 21/09/2015 (Retro UNIX 8086 & 386 system calls)
  86                                    <1> ; UNIX v1 system calls
  87                                    <1> ;_rele      equ 0
  88                                    <1> _ver  equ 0 ; Get TRDOS version (v2.0)
  89                                    <1> _exit       equ 1
  90                                    <1> _fork       equ 2
  91                                    <1> _read       equ 3
  92                                    <1> _write      equ 4
  93                                    <1> _open equ 5
```

```
 94                                     <1> _close      equ 6
 95                                     <1> _wait       equ 7
 96                                     <1> _creat      equ 8
 97                                     <1> _link       equ 9
 98                                     <1> _unlink     equ 10
 99                                     <1> _exec equ 11
100                                     <1> _chdir      equ 12
101                                     <1> _time       equ 13
102                                     <1> _mkdir      equ 14
103                                     <1> _chmod      equ 15
104                                     <1> _chown      equ 16
105                                     <1> _break      equ 17
106                                     <1> _stat equ 18
107                                     <1> _seek equ 19
108                                     <1> _tell       equ 20
109                                     <1> _mount      equ 21
110                                     <1> _umount     equ 22
111                                     <1> _setuid     equ 23
112                                     <1> _getuid     equ 24
113                                     <1> _stime      equ 25
114                                     <1> _quit equ 26
115                                     <1> _intr equ 27
116                                     <1> _fstat      equ 28
117                                     <1> _emt  equ 29
118                                     <1> _mdate      equ 30
119                                     <1> ;_stty      equ 31
120                                     <1> _video  equ 31 ; TRDOS 386 Video Functions (16/05/2016)
121                                     <1> ;_gtty      equ 32
122                                     <1> _audio      equ 32 ; TRDOS 386 Video Functions (16/05/2016)
123                                     <1> ;_ilgins equ 33
124                                     <1> _timer      equ 33 ; TRDOS 386 Timer Functions (18/05/2016)
125                                     <1> _sleep      equ 34 ; Retro UNIX 8086 v1 feature only !
126                                     <1> _msg  equ 35 ; Retro UNIX 386 v1 feature only !
127                                     <1> _geterr     equ 36 ; Retro UNIX 386 v1 feature only !
128                                     <1> _fpsave equ 37 ; TRDOS 386 FPU state option (28/02/2017)
129                                     <1> _pri  equ 38 ; change priority - TRDOS 386 (20/05/2016)
130                                     <1> _rele equ 39 ; TRDOS 386 (19/05/2016)
131                                     <1> _fff  equ 40 ; Find First File - TRDOS 386 (15/10/2016)
132                                     <1> _fnf  equ 41 ; Find Next File - TRDOS 386 (15/10/2016)
133                                     <1> _alloc      equ 42 ; Allocate memory - TRDOS 386 (19/02/2017)
134                                     <1>             ; TRDOS 386 (19/02/2017) DMA buff fuctions
135                                     <1> _dalloc equ 43 ; Deallocate mem - TRDOS 386 (19/02/2017)
136                                     <1> _calbac equ 44 ; Set IRQ callback - TRDOS 386 (20/02/2017)
137                                     <1> _dma  equ 45 ; DMA service - TRDOS 386 (20/08/2017)
138                                     <1>
139                                     <1> %macro sys 1-4
140                                     <1>     ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
141                                     <1>     ; 03/09/2015
142                                     <1>     ; 13/04/2015
143                                     <1>     ; Retro UNIX 386 v1 system call.
144                                     <1>     %if %0 >= 2
145                                     <1>         mov ebx, %2
146                                     <1>         %if %0 >= 3
147                                     <1>             mov ecx, %3
148                                     <1>             %if %0 = 4
149                                     <1>                mov edx, %4
150                                     <1>             %endif
151                                     <1>         %endif
152                                     <1>     %endif
153                                     <1>     mov eax, %1
154                                     <1>     ;int 30h
155                                     <1>     int 40h ; TRDOS 386 (TRDOS v2.0)
156                                     <1> %endmacro
157                                     <1>
158                                     <1> ; TRDOS 386 system calls, interrupt number
159                                     <1> ; 25/12/2016
160                                     <1> SYSCALL_INT_NUM   equ '40' ; '40h'
161                                     <1>
162                                     <1> ; 13/05/2015 - ERROR CODES
163                                     <1> ERR_FILE_NOT_OPEN  equ 10 ; 'file not open !' error
164                                     <1> ERR_FILE_ACCESS    equ 11 ; 'permission denied !' error
165                                     <1> ; 14/05/2015
166                                     <1> ERR_DIR_ACCESS     equ 11 ; 'permission denied !' error
167                                     <1> ERR_FILE_NOT_FOUND equ 12 ; 'file not found !' error
168                                     <1> ERR_TOO_MANY_FILES equ 13 ; 'too many open files !' error
169                                     <1> ERR_DIR_EXISTS     equ 14 ; 'directory already exists !' error
170                                     <1> ; 16/05/2015
171                                     <1> ERR_DRV_NOT_RDY    equ 15 ; 'drive not ready !' error
172                                     <1> ; 18/05/2015
173                                     <1> ERR_DEV_NOT_RDY    equ 15 ; 'device not ready !' error
174                                     <1> ERR_DEV_ACCESS     equ 11 ; 'permission denied !' error
175                                     <1> ERR_DEV_NOT_OPEN   equ 10 ; 'device not open !' error
176                                     <1> ; 07/06/2015
177                                     <1> ERR_FILE_EOF    equ 16 ; 'end of file !' error
178                                     <1> ERR_DEV_VOL_SIZE   equ 16 ; 'out of volume !' error
179                                     <1> ; 09/06/2015
180                                     <1> ERR_DRV_READ    equ 17 ; 'disk read error !'
181                                     <1> ERR_DRV_WRITE         equ 18 ; 'disk write error !'
182                                     <1> ; 16/06/2015
183                                     <1> ERR_NOT_DIR     equ 19 ; 'not a (valid) directory !' error
184                                     <1> ERR_FILE_SIZE         equ 20 ; 'file size error !'
185                                     <1> ; 22/06/2015
186                                     <1> ERR_NOT_SUPERUSER  equ 11 ; 'permission denied !' error
187                                     <1> ERR_NOT_OWNER      equ 11 ; 'permission denied !' error
188                                     <1> ERR_NOT_FILE       equ 11 ; 'permission denied !' error
189                                     <1> ; 23/06/2015
190                                     <1> ERR_FILE_EXISTS    equ 14 ; 'file already exists !' error
191                                     <1> ERR_DRV_NOT_SAME   equ 21 ; 'not same drive !' error
192                                     <1> ERR_DIR_NOT_FOUND  equ 12 ; 'directory not found !' error
193                                     <1> ERR_NOT_EXECUTABLE equ 22 ; 'not executable file !' error
194                                     <1> ; 27/06/2015
195                                     <1> ERR_INV_PARAMETER  equ 23 ; 'invalid parameter !' error
```

```
196                             <1> ERR_INV_DEV_NAME   equ 24 ; 'invalid device name !' error
197                             <1> ; 29/06/2015
198                             <1> ERR_TIME_OUT    equ 25 ; 'time out !' error
199                             <1> ERR_DEV_NOT_RESP   equ 25 ; 'device not responding !' error
200                             <1> ; 10/10/2016
201                             <1> ERR_INV_FILE_NAME  equ 26 ; 'invalid file name !' error
202                             <1> ERR_INV_FLAGS         equ 23 ; 'invalid flags !' error
203                             <1> ; For code compatibility with previous version of TRDOS (2011)
204                             <1> ; (Temporary error codes for current TRDOS 386 -2016- version)
205                             <1> ERR_NO_MORE_FILES  equ 12 ; 'no more files !' error
206                             <1> ERR_PATH_NOT_FOUND equ  3 ; 'path not found !' error
207                             <1>                       ; 'dir not found !' ; TRDOS 8086
208                             <1> ERR_NOT_FOUND:         equ  2 ; 'file not found !' ; TRDOS 8086
209                             <1> ERR_DISK_SPACE         equ 39 ; 'out of volume !' TRDOS 8086
210                             <1>                       ; 'insufficient disk space !' ; 27h
211                             <1> ERR_DISK_WRITE         equ 30 ; 'disk write protected !' ; 16/10/2016
212                             <1> ERR_ACCESS_DENIED  equ  5 ; 'access denied !' ; TRDOS 8086
213                             <1> ; 28/02/2017
214                             <1> ERR_PERM_DENIED        equ 11 ; 'permission denied !' error
215                             <1> ; 18/05/2016
216                             <1> ERR_MISC        equ 27 ; miscellaneous/other errors
217                             <1> ; 15/10/2016
218                             <1> ; TRDOS 8086 -> TRDOS 386 (0Bh -> 28)
219                             <1> ERR_INV_FORMAT         equ 28 ; 'invalid format !' error
220                             <1> ; TRDOS 8086 -> TRDOS 386 (0Dh -> 29)
221                             <1> ERR_INV_DATA    equ 29 ; 'invalid data !' error
222                             <1> ; TRDOS 8086 -> TRDOS 386 (0Eh -> 20)
223                             <1> ERR_ZERO_LENGTH        equ 20  ; 'zero length !' error
224                             <1> ; TRDOS 8086 -> TRDOS 386 (15h -> 17, 1Dh -> 18, 1Eh -> 17)
225                             <1> ERR_DRV_NR_READ        equ 17 ; 'drive not ready or read error !'
226                             <1> ERR_DRV_NR_WRITE   equ 18 ; 'drive not ready or write error !'
227                             <1> ; 15/10/2016
228                             <1> ERR_INV_PATH_NAME  equ 19 ; 'bad path name !' error
229                             <1> ERR_BAD_CMD_ARG        equ  1 ; 'bad command argument !' ; TRDOS 8086
230                             <1> ERR_INV_FNUMBER        equ  1 ; 'invalid function number !' ; TRDOS 8086
231                             <1> ERR_BIG_FILE    equ  8 ; 'big file & out of memory ! ; TRDOS 8086
232                             <1> ERR_BIG_DATA    equ  8 ; 'big data & out of memory ! ; TRDOS 8086
233                             <1> ERR_CLUSTER     equ 35 ; 'cluster not available !' ; TRDOS 8086
234                             <1> ERR_OUT_OF_MEMORY  equ  4 ; 'out of memory !'
235                             <1>                       ; 'insufficient memory !'
236                             <1> ERR_P_VIOLATION        equ 6 ; 'protection violation !'
237                             <1> ERR_PAGE_FAULT         equ 224 ;'page fault !' ;0E0h
238                             <1> ERR_SWP_DISK_READ            equ 40
239                             <1> ERR_SWP_DISK_NOT_PRESENT   equ 41
240                             <1> ERR_SWP_SECTOR_NOT_PRESENT equ 42
241                             <1> ERR_SWP_NO_FREE_SPACE      equ 43
242                             <1> ERR_SWP_DISK_WRITE         equ 44
243                             <1> ERR_SWP_NO_PAGE_TO_SWAP    equ 45
244                             <1> ; 10/04/2017
245                             <1> ERR_BUFFER     equ 46  ; 'buffer error !'
246                             <1> ; 28/08/2017 (20/08/2017)
247                             <1> ERR_DMA                equ -1  ; DMA buffer (allocation/misc.) error!
248                             <1>
249                             <1> ; 26/08/2015
250                             <1> ; 24/07/2015
251                             <1> ; 24/06/2015
252                             <1> MAX_ARG_LEN    equ 256 ; max. length of sys exec arguments
253                             <1> ; 01/07/2015
254                             <1> MAX_MSG_LEN    equ 255 ; max. msg length for 'sysmsg'
255                             <1> ;
256                             <1> ; 06/10/2016
257                             <1> OPENFILES      equ 10  ; max. number of open files (system)
258                             <1> ; 07/10/2016
259                             <1> ;NUMOFDEVICES         equ 20  ; max. num of available devices (sys)
260                             <1>
2303                                %include 'trdosk0.s' ; 04/01/2016
   1                            <1> ; ****************************************************************************
   2                            <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - DEFINITIONS : trdosk0.s
   3                            <1> ; ----------------------------------------------------------------------------
   4                            <1> ; Last Update: 29/02/2016
   5                            <1> ; ----------------------------------------------------------------------------
   6                            <1> ; Beginning: 04/01/2016
   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> ; TRDOS2.ASM (09/11/2011)
  12                            <1> ; ****************************************************************************
  13                            <1> ; TRDOS2.ASM (c) 2004-2011 Erdogan TAN [ 17/01/2004 ] Last Update: 09/11/2011
  14                            <1> ;
  15                            <1> ; Masterboot / Partition Table at Beginning+1BEh
  16                            <1> ptBootable       equ 0
  17                            <1> ptBeginHead      equ 1
  18                            <1> ptBeginSector    equ 2
  19                            <1> ptBeginCylinder  equ 3
  20                            <1> ptFileSystemID   equ 4
  21                            <1> ptEndHead        equ 5
  22                            <1> ptEndSector      equ 6
  23                            <1> ptEndCylinder    equ 7
  24                            <1> ptStartSector    equ 8
  25                            <1> ptSectors        equ 12
  26                            <1>
  27                            <1> ; Boot Sector Parameters at 7C00h
  28                            <1> DataArea1     equ -4
  29                            <1> DataArea2     equ -2
  30                            <1> BootStart     equ 0h
  31                            <1> OemName       equ 03h
  32                            <1> BytesPerSec   equ 0Bh
  33                            <1> SecPerClust   equ 0Dh
  34                            <1> ResSectors    equ 0Eh
  35                            <1> FATs          equ 10h
  36                            <1> RootDirEnts   equ 11h
```

```
                        <1> Sectors       equ 13h
                        <1> Media         equ 15h
                        <1> FATSecs       equ 16h
                        <1> SecPerTrack   equ 18h
                        <1> Heads         equ 1Ah
                        <1> Hidden1       equ 1Ch
                        <1> Hidden2       equ 1Eh
                        <1> HugeSec1      equ 20h
                        <1> HugeSec2      equ 22h
                        <1> DriveNumber   equ 24h
                        <1> Reserved1     equ 25h
                        <1> bootsignature equ 26h
                        <1> VolumeID      equ 27h
                        <1> VolumeLabel   equ 2Bh
                        <1> FileSysType   equ 36h
                        <1> Reserved2     equ 3Eh                       ; Starting cluster of P2000
                        <1>
                        <1> ; FAT32 BPB Structure
                        <1> FAT32_FAT_Size equ 36
                        <1> FAT32_RootFClust equ 44
                        <1> FAT32_FSInfoSec equ 48
                        <1> FAT32_DrvNum equ 64
                        <1> FAT32_BootSig equ 66
                        <1> FAT32_VolID equ 67
                        <1> FAT32_VolLab equ 71
                        <1> FAT32_FilSysType equ 82
                        <1>
                        <1> ; BIOS Disk Parameters
                        <1> DPDiskNumber  equ 0h
                        <1> DPDType       equ 1h
                        <1> DPReturn      equ 2h
                        <1> DPHeads       equ 3h
                        <1> DPCylinders   equ 4h
                        <1> DPSecPerTrack equ 6h
                        <1> DPDisks       equ 7h
                        <1> DPTableOff    equ 8h
                        <1> DPTableSeg    equ 0Ah
                        <1> DPNumOfSecs   equ 0Ch
                        <1>
                        <1> ; BIOS INT 13h Extensions (LBA extensions)
                        <1> ; Just After DP Data (DPDiskNumber+)
                        <1> DAP_PacketSize equ 10h  ; If extensions present, this byte will be >=10h
                        <1> DAP_Reserved1 equ 11h   ; Reserved Byte
                        <1> DAP_NumOfBlocks equ 12h ; Value of this byte must be 0 to 127
                        <1> DAP_Reserved2 equ 13h   ; Reserved Byte
                        <1> DAP_Destination equ 14h ; Address of Transfer Buffer as SEGMENT:OFFSET
                        <1> DAP_LBA_Address equ 18h ; LBA=(C1*H0+H1)*S0+S1-1
                        <1>                         ; C1= Selected Cylinder Number
                        <1>                         ; H0= Number Of Heads (Maximum Head Number + 1)
                        <1>                         ; H1= Selected Head Number
                        <1>                         ; S0= Maximum Sector Number
                        <1>                         ; S1= Selected Sector Number
                        <1>                         ; QUAD WORD
                        <1> ; DAP_Flat_Destination equ 20h ; 64 bit address, if value in 4h is FFFF:FFFFh
                        <1>                               ; QUAD WORD (Also, value in 0h must be 18h)
                        <1>                               ; TR-DOS will not use 64 bit Flat Address
                        <1>
                        <1> ; INT 13h Function 48h "Get Enhanced Disk Drive Parameters"
                        <1> ; Just After DP Data (DPDiskNumber+)
                        <1> GetDParams_48h equ 20h ; Word. Data Length, must be 26 (1Ah) for short data.
                        <1> GDP_48h_InfoFlag equ 22h ; Word
                        <1> ; Bit 1 = 1 -> The geometry returned in bytes 4-15 is valid.
                        <1> GDP_48h_NumOfPCyls equ 24h ; Double Word. Number physical cylinders.
                        <1> GDP_48h_NumOfPHeads equ 28h ; Double Word. Number of physical heads.
                        <1> GDP_48h_NumOfPSpT equ 2Ch ; Double word. Num of physical sectors per track.
                        <1> GDP_48h_LBA_Sectors equ 30h ; 8 bytes. Number of physical/LBA sectors.
                        <1> GDP_48h_BytesPerSec equ 38h ; Word. Number of bytes in a sector.
                        <1>
                        <1> ; TR-DOS Standalone Program Extensions to the DiskParams Block
                        <1> ; Just After DP Data (DPDiskNumber+)
                        <1> TRDP_CurrentSector equ 3Ah  ; DX:AX (LBA)
                        <1> TRDP_SectorCount equ 3Eh    ; CX (or Counter)
                        <1>
                        <1>
                        <1> ; DOS Logical Disks
                        <1> LD_Name equ 0
                        <1> LD_DiskType equ 1
                        <1> LD_PhyDrvNo equ 2
                        <1> LD_FATType equ 3
                        <1> LD_FSType equ 4
                        <1> LD_LBAYes equ 5
                        <1> LD_BPB equ 6
                        <1> LD_FATBegin equ 96
                        <1> LD_ROOTBegin equ 100
                        <1> LD_DATABegin equ 104
                        <1> LD_StartSector equ 108
                        <1> LD_TotalSectors equ 112
                        <1> LD_FreeSectors equ 116
                        <1> LD_Clusters equ 120
                        <1> LD_PartitionEntry equ 124
                        <1> LD_DParamEntry equ 125
                        <1> LD_MediaChanged equ 126
                        <1> LD_CDirLevel equ 127
                        <1> LD_CurrentDirectory equ 128
                        <1>
                        <1> ; Singlix FS Extensions to DOS Logical Disks
                        <1> ; 03/01/2010 (LD_BPB compatibility for CHS r/w)
                        <1>
                        <1> LD_FS_Name equ 0
                        <1> LD_FS_DiskType equ 1
                        <1> LD_FS_PhyDrvNo equ 2
                        <1> LD_FS_FATType equ 3
```

```
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
227                                     <1> BPB_NumFATs equ 16
228                                     <1> BPB_RootEntCnt equ 17
229                                     <1> BPB_TotalSec16 equ 19
230                                     <1> BPB_Media equ 21
231                                     <1> BPB_FATSz16 equ 22
232                                     <1> BPB_SecPerTrk equ 24
233                                     <1> BPB_NumHeads equ 26
234                                     <1> BPB_HiddSec equ 28
235                                     <1> BPB_TotalSec32 equ 32
236                                     <1>
237                                     <1> ; FAT12 and FAT16 only
238                                     <1> BS_DrvNum equ 36
239                                     <1> BS_Reserved1 equ 37
240                                     <1> BS_BootSig equ 38
```

```
241                                 <1> BS_VolID equ 39
242                                 <1> BS_VolLab equ 43
243                                 <1> BS_FilSysType equ 54 ; 8 bytes
244                                 <1> BS_BootCode equ 62
245                                 <1>
246                                 <1> ; FAT32 only
247                                 <1> BPB_FATSz32 equ 36 ; FAT32, 4 bytes
248                                 <1> BPB_ExtFlags equ 40 ; FAT32, 2 bytes
249                                 <1> BPB_FSVer equ 42 ; FAT32, 2 bytes
250                                 <1> BPB_RootClus equ 44 ; FAT32, 4 bytes
251                                 <1> BPB_FSInfo equ 48 ; FAT 32, 2 bytes
252                                 <1> BPB_BkBootSec equ 50 ; FAT32, 2 bytes
253                                 <1> BPB_Reserved equ 52 ; FAT32, 12 bytes
254                                 <1> BS_FAT32_DrvNum equ 64 ; FAT32, 1 byte
255                                 <1> BS_FAT32_Reserved1 equ 65 ; FAT32, 1 byte
256                                 <1> BS_FAT32_BootSig equ 66 ; FAT32, 1 byte
257                                 <1> BS_FAT32_VolID equ 67 ; FAT32, 4 bytes
258                                 <1> BS_FAT32_VolLab equ 71 ; FAT32, 11 bytes
259                                 <1> BS_FAT32_FilSysType equ 82 ; FAT32, 8 bytes
260                                 <1> BS_FAT32_BootCode equ 90
261                                 <1>
262                                 <1> ; 29/02/2016
263                                 <1> ;(FAT32 Free Cluster Count & First Free Cluster values)
264                                 <1> ;[BPB_Reserved] = Free Cluster Count (offset 52)
265                                 <1> ;[BPB_Reserved+4] = First Free Cluster (offset 56)
266                                 <1>
267                                 <1> BS_Validation equ 510
268                                 <1>
269                                 <1> ; 15/02/2016
270                                 <1> ; FILE.ASM - 09/10/2011
271                                 <1> ; Directory Entry Structure
272                                 <1> ; 29/10/2009 (According to Microsoft FAT32 File System Specification)
273                                 <1> DirEntry_Name equ 0
274                                 <1> DirEntry_Attr equ 11
275                                 <1> DirEntry_NTRes equ 12
276                                 <1> DirEntry_CrtTimeTenth equ 13
277                                 <1> DirEntry_CrtTime equ 14
278                                 <1> DirEntry_CrtDate equ 16
279                                 <1> DirEntry_LastAccDate equ 18
280                                 <1> DirEntry_FstClusHI equ 20
281                                 <1> DirEntry_WrtTime equ 22
282                                 <1> DirEntry_WrtDate equ 24
283                                 <1> DirEntry_FstClusLO equ 26
284                                 <1> DirEntry_FileSize equ 28
2304                                    %include 'trdosk1.s' ; 04/01/2016
   1                                <1> ; ****************************************************************************
   2                                <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - SYS INIT : trdosk1.s
   3                                <1> ; ----------------------------------------------------------------------------
   4                                <1> ; Last Update: 23/01/2017
   5                                <1> ; ----------------------------------------------------------------------------
   6                                <1> ; Beginning: 04/01/2016
   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> ; TRDOS2.ASM (09/11/2011)
  12                                <1> ; ****************************************************************************
  13                                <1> ; TRDOS2.ASM (c) 2004-2011 Erdogan TAN [ 17/01/2004 ] Last Update: 09/11/2011
  14                                <1> ;
  15                                <1>
  16                                <1> sys_init:
  17                                <1>       ; 23/01/2017
  18                                <1>       ; 07/05/2016
  19                                <1>       ; 02/05/2016
  20                                <1>       ; 24/04/2016
  21                                <1>       ; 14/04/2016
  22                                <1>       ; 13/04/2016
  23                                <1>       ; 30/03/2016
  24                                <1>       ; 24/01/2016
  25                                <1>       ; 06/01/2016
  26                                <1>       ; 04/01/2016
  27                                <1>
  28                                <1>       ; 23/01/2017 - reset timer frequency (to 18.2Hz)
  29 0000624E B036                  <1>       mov   al, 00110110b ; 36h
  30 00006250 E643                  <1>       out   43h, al
  31 00006252 31C0                  <1>       xor   eax, eax  ; sub     al, al ; 0
  32 00006254 E640                  <1>       out   40h, al ; LB
  33 00006256 E640                  <1>       out   40h, al ; HB
  34                                <1>       ;
  35                                <1>       ; 30/03/2016
  36                                <1>       ; Clear Logical DOS Disk Description Tables Area
  37                                <1>       ;xor   eax, eax
  38 00006258 BF00010900            <1>       mov   edi, Logical_DOSDisks
  39 0000625D B980060000            <1>       mov   ecx, 6656/4 ; 26*256 = 6656 bytes
  40 00006262 F3AB                  <1>       rep   stosd ; 1664 times 4 bytes
  41                                <1>
  42 00006264 B83F3A2F00            <1>       mov   eax, '?:/'
  43 00006269 A3[C7520100]          <1>       mov   [Current_Dir_Drv], eax
  44                                <1>
  45                                <1>       ; Logical DRV INIT (only for hard disks)
  46 0000626E E8B3010000            <1>       call  ldrv_init  ; trdosk2.s
  47                                <1>
  48                                <1>       ; When floppy_drv_init call is disabled
  49                                <1>       ; media changed sign is needed
  50                                <1>       ; for proper drive initialization
  51                                <1>
  52 00006273 BE00010900            <1>       mov   esi, Logical_DOSDisks
  53 00006278 B001                  <1>       mov   al, 1 ; Initialization sign (invalid_fd_parameter)
  54 0000627A 83C67E                <1>       add   esi, LD_MediaChanged ; Media Change Status = 1 (init needed)
  55 0000627D 8806                  <1>       mov   [esi], al ; A:
  56 0000627F 81C600010000          <1>       add   esi, 100h
  57 00006285 8806                  <1>       mov   [esi], al ; B:
```

```
                                <1>
                                <1> _current_drive_bootdisk:
00006287 8A15[F25C0000]         <1>       mov    dl, [boot_drv] ; physical drive number
0000628D 80FAFF                 <1>       cmp    dl, 0FFh
00006290 740A                   <1>       je     short _last_dos_diskno_check
                                <1> _boot_drive_check:
00006292 80FA80                 <1>       cmp    dl, 80h
00006295 7218                   <1>       jb     short _current_drive_a
00006297 80EA7E                 <1>       sub    dl, 7Eh ; C = 2 , D = 3
0000629A EB13                   <1>       jmp    short _current_drive_a
                                <1>
                                <1> _last_dos_diskno_check:
0000629C 8A15[AC060100]         <1>       mov    dl, [Last_DOS_DiskNo]
000062A2 80FA02                 <1>       cmp    dl, 2
000062A5 7706                   <1>       ja     short _current_drive_c
000062A7 7406                   <1>       je     short _current_drive_a
000062A9 30D2                   <1>       xor    dl, dl ; A:
000062AB EB02                   <1>       jmp    short _current_drive_a
                                <1>
                                <1> _current_drive_c:
000062AD B202                   <1>       mov    dl, 2 ; C:
                                <1>
                                <1> _current_drive_a:
000062AF 8815[F35C0000]         <1>       mov    [drv], dl
000062B5 BE[AE060100]           <1>         mov     esi, msg_CRLF_temp
000062BA E89E000000             <1>       call   print_msg
                                <1>
000062BF 8A15[F35C0000]         <1>       mov    dl, [drv]
000062C5 E893090000             <1>       call   change_current_drive
000062CA 730C                   <1>       jnc    short _start_mainprog
                                <1>
                                <1> _drv_not_ready_error:
000062CC BE[69090100]           <1>       mov    esi, msgl_drv_not_ready
000062D1 E887000000             <1>       call   print_msg
000062D6 EB63                   <1>         jmp     _end_of_mainprog
                                <1>
                                <1> _start_mainprog:
                                <1>       ; 07/01/2017
                                <1>       ; 07/05/2016
                                <1>       ; 02/05/2016
                                <1>       ; 24/04/2016
                                <1>       ; Retro UNIX 386 v1, 'sys_init' (u0.s)
                                <1>       ; 23/06/2015
                                <1>
                                <1>       ; 02/05/2016
                                <1>       ; 24/04/2016
000062D8 66B80100               <1>       mov    ax, 1
000062DC A2[B3030300]           <1>       mov    [u.uno], al
000062E1 66A3[4E030300]         <1>       mov    [mpid], ax
000062E7 66A3[20000300]         <1>       mov    [p.pid], ax
000062ED A2[B0000300]           <1>       mov    [p.stat], al
000062F2 C605[A8030300]04       <1>       mov    byte [u.quant], time_count  ; 07/01/2017
                                <1>       ;
000062F9 A1[00520100]           <1>       mov    eax, [k_page_dir]
000062FE A3[B8030300]           <1>       mov    [u.pgdir], eax ; reset
                                <1>       ;
00006303 E871E8FFFF             <1>       call   allocate_page
00006308 0F82A3000000           <1>       jc     panic
0000630E A3[B4030300]           <1>       mov    [u.upage], eax ; user structure page
00006313 A3[C0000300]           <1>       mov    [p.upage], eax
00006318 E8D6E8FFFF             <1>       call   clear_page
                                <1>       ;
                                <1>       ; 24/08/2015
0000631D FE0D[5B030300]         <1>       dec    byte [sysflg] ; FFh = ready for system call
                                <1>                            ; 0 = executing a system call
                                <1>       ; 13/04/2016
                                <1>       ; Clear Environment Variables Page/Area
00006323 BF00300900             <1>       mov    edi, Env_Page ; 93000h
00006328 B980000000             <1>       mov    ecx, Env_Page_Size / 4    ; 512/4  (4096/4)

0000632D 31C0                   <1>       xor    eax, eax
0000632F F3AB                   <1>       rep    stosd
                                <1>
                                <1>       ; 14/04/2016
00006331 E8C3320000             <1>       call   mainprog_startup_configuration
                                <1>
00006336 E8630A0000             <1>         call    dos_prompt
                                <1>
                                <1> _end_of_mainprog:
0000633B BE[AE060100]           <1>         mov     esi, msg_CRLF_temp
00006340 E818000000             <1>       call   print_msg
00006345 BE[B4060100]           <1>       mov    esi, mainprog_Version
0000634A E80E000000             <1>       call   print_msg
                                <1>       ; 24/01/2016
0000634F 28E4                   <1>       sub    ah, ah
00006351 E8C0A8FFFF             <1>       call   int16h ; call getch
00006356 E9A0ADFFFF             <1>       jmp    cpu_reset
                                <1>
0000635B EBFE                   <1> infinitiveloop: jmp short infinitiveloop
                                <1>
                                <1> print_msg:
                                <1>       ; 13/05/2016
                                <1>       ; 04/01/2016
                                <1>       ; 01/07/2015
                                <1>       ; 13/03/2015 (Retro UNIX 386 v1)
                                <1>       ; 07/03/2014 (Retro UNIX 8086 v1)
                                <1>       ; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)
                                <1>       ;
0000635D 8A3D[2E520100]         <1>       mov    bh, [ACTIVE_PAGE] ; 04/01/2016 (ptty)
                                <1>       ;mov   bl, 07h ; Black background, light gray forecolor
                                <1>
00006363 AC                     <1>       lodsb
```

```
                                <1> pmsg1:
00006364 56                     <1>       push   esi
                                <1>       ;mov   bh, [ACTIVE_PAGE] ; 04/01/2016 (ptty)
00006365 B307                   <1>       mov    bl, 07h ; Black background, light gray forecolor
00006367 E846B9FFFF             <1>       call   _write_tty
0000636C 5E                     <1>       pop    esi
0000636D AC                     <1>       lodsb
0000636E 20C0                   <1>       and    al, al
00006370 75F2                   <1>       jnz    short pmsg1
00006372 C3                     <1>       retn
                                <1>
                                <1> clear_screen:
                                <1>       ; 13/05/2016
                                <1>       ; 30/01/2016
                                <1>       ; 24/01/2016
                                <1>       ; 04/01/2016
00006373 0FB61D[2E520100]       <1>       movzx  ebx, byte [ACTIVE_PAGE] ; video page number (0 to 7)
0000637A 8AA3[D35E0000]         <1>       mov    ah, [ebx+vmode] ; default = 03h (80x25 text)
00006380 80FC04                 <1>       cmp    ah, 4
00006383 7205                   <1>       jb     short cls1
00006385 80FC07                 <1>       cmp    ah, 7
00006388 7526                   <1>       jne    short vga_clear
                                <1> cls1:
                                <1>       ;mov   bh, bl
                                <1>       ;mov   bl, 7
0000638A 3A25[C25E0000]         <1>       cmp    ah, [CRT_MODE] ; current video mode ?
                                <1>       ;je    short cls2 ; yes (current video mode = 3)
                                <1>       ;;call set_mode_3 ; set video mode to 3 (& clear screen)
                                <1>       ;;retn
                                <1>       ;jmp   set_mode_3
00006390 0F8526B9FFFF           <1>       jne    set_mode_3
                                <1> cls2:
00006396 88DF                   <1>       mov    bh, bl ; video page (0 to 7)
00006398 B307                   <1>       mov    bl, 07h ; attribute to be used on blanked line
0000639A 28C0                   <1>       sub    al, al ; 0 =  entire window
0000639C 6631C9                 <1>       xor    cx, cx
0000639F 66BA4F18               <1>       mov    dx, 184Fh
000063A3 E862B6FFFF             <1>       call   _scroll_up ; 24/01/2016
                                <1>       ;
                                <1>       ;mov   bh, [ACTIVE_PAGE] ; video page number (0 to 7)
000063A8 6631D2                 <1>       xor    dx, dx
000063AB E898B9FFFF             <1>       call   _set_cpos ; 24/01/2016
                                <1>       ;retn
                                <1> vga_clear:
000063B0 C3                     <1>       retn
                                <1>
                                <1> panic:
                                <1>       ; 13/05/2016 (TRDOS 386 = TRDOS v2)
                                <1>       ; 13/03/2015 (Retro UNIX 386 v1)
                                <1>       ; 07/03/2014 (Retro UNIX 8086 v1)
000063B1 BE[4E130100]           <1>       mov    esi, panic_msg
000063B6 E8A2FFFFFF             <1>       call   print_msg
                                <1> key_to_reboot:
                                <1>         ; 24/01/2016
000063BB 28E4                   <1>         sub     ah, ah
000063BD E854A8FFFF             <1>         call    int16h ; call   getch
                                <1>         ; wait for a character from the current tty
                                <1>       ;
000063C2 B00A                   <1>       mov    al, 0Ah
000063C4 8A3D[2E520100]         <1>       mov    bh, [ptty] ; [ACTIVE_PAGE]
000063CA B307                   <1>       mov    bl, 07h ; Black background,
                                <1>                    ; light gray forecolor
000063CC E8E1B8FFFF             <1>       call   _write_tty
000063D1 E925ADFFFF             <1>       jmp    cpu_reset
                                <1>
                                <1> ctrlbrk:
                                <1>       ; 12/11/2015
                                <1>       ; 13/03/2015 (Retro UNIX 386 v1)
                                <1>       ; 06/12/2013 (Retro UNIX 8086 v1)
                                <1>       ;
                                <1>       ; INT 1Bh (control+break) handler
                                <1>       ;
                                <1>             ; Retro Unix 8086 v1 feature only!
                                <1>             ;
000063D6 66833D[AA030300]00     <1>       cmp    word [u.intr], 0
000063DE 7645                   <1>       jna    short cbrk4
                                <1> cbrk0:
                                <1>       ; 12/11/2015
                                <1>       ; 06/12/2013
000063E0 66833D[AC030300]00     <1>       cmp    word [u.quit], 0
000063E8 743B                   <1>       jz     short cbrk4
                                <1>       ;
                                <1>       ; 20/09/2013
000063EA 6650                   <1>       push   ax
000063EC A0[2E520100]           <1>       mov    al, [ptty]
                                <1>       ;
                                <1>       ; 12/11/2015
                                <1>       ;
                                <1>       ; ctrl+break (EOT, CTRL+D) from serial port
                                <1>       ; or ctrl+break from console (pseudo) tty
                                <1>       ; (!redirection!)
                                <1>       ;
000063F1 3C08                   <1>       cmp    al, 8 ; serial port tty nums > 7
000063F3 7211                   <1>         jb      short cbrk1 ; console (pseudo) tty
                                <1>       ;
                                <1>       ; Serial port interrupt handler sets [ptty]
                                <1>       ; to the port's tty number (as temporary).
                                <1>       ;
                                <1>       ; If active process is using a stdin or
                                <1>       ; stdout redirection (by the shell),
                                <1>         ; console tty keyboard must be available
                                <1>       ; to terminate running process,
```

```
 261                                <1>       ; in order to prevent a deadlock.
 262                                <1>       ;
 263 000063F5 52                    <1>       push   edx
 264 000063F6 0FB615[B3030300]      <1>       movzx  edx, byte [u.uno]
 265 000063FD 3A82[7F000300]        <1>       cmp     al, [edx+p.ttyc-1] ; console tty (rw)
 266 00006403 5A                    <1>       pop    edx
 267 00006404 7412                  <1>       je     short cbrk2
 268                                <1> cbrk1:
 269 00006406 FEC0                  <1>       inc    al  ; [u.ttyp] : 1 based tty number
 270                                <1>       ; 06/12/2013
 271 00006408 3A05[94030300]        <1>       cmp    al, [u.ttyp] ; recent open tty (r)
 272 0000640E 7408                  <1>       je     short cbrk2
 273 00006410 3A05[95030300]        <1>         cmp     al, [u.ttyp+1] ; recent open tty (w)
 274 00006416 750B                  <1>       jne    short cbrk3
 275                                <1> cbrk2:
 276                                <1>       ;; 06/12/2013
 277                                <1>       ;mov   ax, [u.quit]
 278                                <1>       ;and   ax, ax
 279                                <1>       ;jz    short cbrk3
 280                                <1>       ;
 281 00006418 6631C0                <1>       xor    ax, ax ; 0
 282 0000641B 6648                  <1>       dec    ax
 283                                <1>       ; 0FFFFh = 'ctrl+brk' keystroke
 284 0000641D 66A3[AC030300]        <1>       mov    [u.quit], ax
 285                                <1> cbrk3:
 286 00006423 6658                  <1>       pop    ax
 287                                <1> cbrk4:
 288 00006425 C3                    <1>       retn
2305                                    %include 'trdosk2.s' ; 04/01/2016
   1                                <1> ; ****************************************************************************
   2                                <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - DRV INIT : trdosk2.s
   3                                <1> ; ----------------------------------------------------------------------------
   4                                <1> ; Last Update: 06/07/2016
   5                                <1> ; ----------------------------------------------------------------------------
   6                                <1> ; Beginning: 04/01/2016
   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> ; TRDOS2.ASM (09/11/2011)
  12                                <1> ; ****************************************************************************
  13                                <1> ; DRV_INIT.ASM (c) 2009-2011 Erdogan TAN  [26/09/2009] Last Update: 07/08/2011
  14                                <1> ;
  15                                <1>
  16                                <1> ldrv_init: ; Logical Drive Initialization
  17                                <1>       ; 12/02/2016
  18                                <1>       ; 06/01/2016
  19                                <1>       ;     ('diskinit.inc', 'diskio.inc' integration)
  20                                <1>       ; 04/01/2016 (TRDOS 386 = TRDOS v2.0)
  21                                <1>       ; 07/08/2011
  22                                <1>       ; 20/09/2009
  23                                <1>       ; 2005
  24 00006426 0FB60D[9C520100]      <1>       movzx  ecx, byte [HF_NUM] ; number of fixed disks
  25 0000642D 80F901                <1>       cmp    cl, 1
  26 00006430 7301                  <1>       jnb    short load_hd_partition_tables
  27                                <1>       ; No hard disks
  28 00006432 C3                    <1>       retn
  29                                <1> load_hd_partition_tables:
  30 00006433 8B35[A0520100]        <1>       mov    esi, [HDPM_TBL_VEC] ; primary master disk FDPT
  31 00006439 BF[C6560100]          <1>       mov    edi, PTable_hd0
  32 0000643E B280                  <1>       mov    dl, 80h
  33                                <1> load_next_hd_partition_table:
  34 00006440 51                    <1>       push   ecx
  35 00006441 57                    <1>       push   edi
  36 00006442 56                    <1>       push   esi ; FDPT (+ DPTE) address
  37 00006443 8A4614                <1>       mov    al, [esi+20] ; DPTE offset 4
  38 00006446 2440                  <1>       and    al, 40h ;  LBA bit (bit 6)
  39                                <1>       ;shr   al, 6
  40 00006448 A2[C7580100]          <1>       mov    [HD_LBA_yes], al
  41 0000644D E81C040000            <1>       call   load_masterboot
  42 00006452 7275                  <1>       jc     short pass_pt_this_hard_disk
  43                                <1>
  44 00006454 BE[84560100]          <1>       mov    esi, PartitionTable
  45 00006459 89F3                  <1>       mov    ebx, esi
  46                                <1>       ;mov   ecx, 16
  47 0000645B B110                  <1>       mov    cl, 16
  48 0000645D F3A5                  <1>       rep    movsd
  49 0000645F 89DE                  <1>       mov    esi, ebx
  50 00006461 C605[F55C0000]04      <1>       mov    byte [hdc], 4 ; 4 - partition index
  51                                <1> loc_validate_hdp_partition:
  52 00006468 807E0400              <1>       cmp    byte [esi+ptFileSystemID], 0
  53 0000646C 7641                  <1>       jna    short loc_validate_next_hdp_partition2
  54 0000646E 56                    <1>       push   esi ; Masterboot partition table offset
  55 0000646F 52                    <1>       push   edx ; dl = Physical drive number
  56 00006470 FE05[C8580100]        <1>       inc    byte [PP_Counter]
  57 00006476 31FF                  <1>         xor  edi, edi ; 0
  58                                <1>       ; Input -> ESI = PartitionTable offset
  59                                <1>       ; DL = Hard disk drive number
  60                                <1>       ; EDI = 0 -> Primary Partition
  61                                <1>       ; EDI > 0 -> Extended Partition's Start Sector
  62 00006478 E879010000            <1>       call   validate_hd_fat_partition
  63 0000647D 730A                  <1>       jnc    short loc_set_valid_hdp_partition_entry
  64                                <1>       ;pop   edx
  65                                <1>       ;push  edx
  66 0000647F 8B1424                <1>       mov    edx, [esp]
  67 00006482 E8C5020000            <1>       call   validate_hd_fs_partition
  68 00006487 7224                  <1>       jc     short loc_validate_next_hdp_partition1
  69                                <1> loc_set_valid_hdp_partition_entry:
  70 00006489 8A0D[AC060100]        <1>       mov    cl, [Last_DOS_DiskNo]
  71 0000648F 80C141                <1>       add    cl, 'A'
  72                                <1>       ; ESI = Logical dos drive description table address
  73 00006492 880E                  <1>       mov    [esi+LD_Name], cl
```

```
 74 00006494 8A6602              <1>       mov    ah, [esi+LD_PhyDrvNo]
 75 00006497 88E0                <1>       mov    al, ah ; Physical drive number
 76 00006499 2C80                <1>       sub    al, 80h
 77 0000649B C0E002              <1>       shl    al, 2
 78 0000649E 0404                <1>       add    al, 4 ; 0 Based
 79 000064A0 2A05[F55C0000]      <1>       sub    al, [hdc] ; 4 - partition index
 80                              <1>       ; AL = Partition entry/index, 0 based
 81                              <1>       ;  0 -> hd 0, Partition Table offset = 0
 82                              <1>       ; 15 -> hd 3, Partition Table offset = 3
 83                              <1>       ;mov   [esi+LD_PartitionEntry], al
 84 000064A6 80EC7E              <1>       sub    ah, 7Eh
 85                              <1>       ; AH = Physical drive index, zero based
 86                              <1>       ;  0 for drive A:, 2 for drive C:
 87                              <1>       ;mov   [esi+LD_DParamEntry], ah
 88 000064A9 6689467C            <1>       mov    [esi+LD_PartitionEntry], ax
 89                              <1> loc_validate_next_hdp_partition1:
 90 000064AD 5A                  <1>       pop    edx ; dl = Physical drive number
 91 000064AE 5E                  <1>       pop    esi ; Masterboot partition table offset
 92                              <1> loc_validate_next_hdp_partition2:
 93                              <1>       ; ESI = PartitionTable offset
 94                              <1>       ; DL = Hard/Fixed disk drive number
 95 000064AF FE0D[F55C0000]      <1>       dec    byte [hdc] ; 4 - partition index
 96 000064B5 7412                <1>       jz     short pass_pt_this_hard_disk
 97 000064B7 83C610              <1>       add    esi, 16 ; 10h
 98 000064BA EBAC                <1>       jmp    short loc_validate_hdp_partition
 99                              <1> loc_next_hd_partition_table:
100 000064BC FEC2                <1>       inc    dl
101 000064BE 83C620              <1>       add    esi, 32 ; next FDPT address
102 000064C1 83C740              <1>       add    edi, 64 ; next partition table destination
103 000064C4 E977FFFFFF          <1>         jmp     load_next_hd_partition_table
104                              <1> pass_pt_this_hard_disk:
105 000064C9 5E                  <1>       pop    esi ; FDPT (+ DPTE) address
106 000064CA 5F                  <1>       pop    edi ; Ptable_hd?
107 000064CB 59                  <1>       pop    ecx
108 000064CC E2EE                <1>       loop   loc_next_hd_partition_table
109 000064CE 803D[C8580100]01    <1>       cmp    byte [PP_Counter], 1
110 000064D5 7301                <1>       jnb    short load_extended_dos_partitions
111                              <1>       ; Empty partition table
112 000064D7 C3                  <1>       retn
113                              <1> load_extended_dos_partitions:
114 000064D8 BE[C6560100]        <1>       mov    esi, PTable_hd0
115 000064DD BF[C6570100]        <1>       mov    edi, PTable_ep0
116 000064E2 C605[F55C0000]80    <1>       mov    byte [hdc], 80h
117                              <1> next_hd_extd_partition:
118 000064E9 56                  <1>       push   esi ; PTable_hd? offset
119 000064EA 57                  <1>       push   edi ; PTable_ep?
120                              <1>       ;mov   ecx, 4
121 000064EB B104                <1>       mov    cl, 4
122 000064ED 8A15[F55C0000]      <1>       mov    dl, byte [hdc]
123                              <1> hd_check_fs_id_05h:
124 000064F3 8A4604              <1>       mov    al, [esi+ptFileSystemID]
125 000064F6 3C05                <1>       cmp    al, 05h ; Is it an extended dos partition ?
126 000064F8 7404                <1>       je     short loc_set_ep_start_sector
127 000064FA 3C0F                <1>       cmp    al, 0Fh ; Is it an extended win4 (LBA mode) partition ?
128 000064FC 7546                <1>       jne    short continue_to_check_ep
129                              <1> loc_set_ep_start_sector:
130 000064FE FE05[C9580100]      <1>       inc    byte [EP_Counter]
131 00006504 88D4                <1>       mov    ah, dl ; byte [hdc]
132 00006506 86E0                <1>       xchg   ah, al ; al = Drv Number, ah = Partition Identifier
133 00006508 50                  <1>       push   eax
134 00006509 30E4                <1>       xor    ah, ah
135 0000650B 2C80                <1>       sub    al, 80h
136 0000650D 50                  <1>       push   eax
137 0000650E C0E002              <1>       shl    al, 2 ; al = al * 4
138 00006511 0FB6D8              <1>       movzx  ebx, al
139 00006514 81C3[CA580100]      <1>       add    ebx, EP_StartSector
140 0000651A 8B4608              <1>       mov    eax, [esi+ptStartSector]
141                              <1>         ; EAX = Extended partition's start sector
142 0000651D 8903                <1>         mov  [ebx], eax
143 0000651F 58                  <1>       pop    eax ; AL = Drv number - 80h, AH = 0
144 00006520 5A                  <1>       pop    edx ; DL = Drv number, DH = Partition ID
145 00006521 BB[C6540100]        <1>       mov    ebx, MasterBootBuff
146 00006526 803D[C7580100]01    <1>       cmp    byte [HD_LBA_yes], 1 ; LBA ready = Yes
147 0000652D 7240                <1>       jb     short loc_hd_load_ep_05h
148 0000652F 80FE05              <1>       cmp    dh, 05h
149 00006532 743B                <1>       je     short loc_hd_load_ep_05h
150                              <1> loc_hd_load_ep_0Fh:
151                              <1>       ; 04/01/2016
152 00006534 51                  <1>       push   ecx
153 00006535 8B4E08              <1>       mov    ecx, [esi+ptStartSector] ; sector number
154                              <1>       ;mov   ebx, MasterBootBuff ; buffer address
155                              <1>       ; LBA read/write (with private LBA function)
156                              <1>       ;((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))
157                              <1>       ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)
158 00006538 B41B                <1>       mov    ah, 1Bh ; LBA read
159 0000653A B001                <1>       mov    al, 1 ; sector count
160 0000653C E8C5DCFFFF          <1>       call   int13h
161 00006541 59                  <1>       pop    ecx
162 00006542 733F                <1>       jnc    short loc_hd_move_ep_table
163                              <1> continue_to_check_ep:
164 00006544 83C610              <1>       add    esi, 16
165 00006547 E2AA                <1>       loop   hd_check_fs_id_05h
166                              <1> continue_check_ep_next_disk:
167 00006549 5F                  <1>       pop    edi ; PTable_ep?
168 0000654A 5E                  <1>       pop    esi ; PTable_hd?
169 0000654B A0[9C520100]        <1>       mov    al, [HF_NUM] ; number of hard disks
170 00006550 047F                <1>       add    al, 7Fh
171 00006552 3805[F55C0000]      <1>       cmp    [hdc], al
172 00006558 0F8392000000        <1>       jnb    loc_validating_hd_partitions_ok
173 0000655E 83C640              <1>       add    esi, 64
174 00006561 83C740              <1>       add    edi, 64
175 00006564 FE05[F55C0000]      <1>       inc    byte [hdc]
```

```
176 0000656A E97AFFFFFF          <1>       jmp    next_hd_extd_partition
177                              <1> loc_hd_load_ep_05h:
178 0000656F 51                  <1>       push   ecx
179 00006570 8A7601              <1>       mov    dh, [esi+ptBeginHead]
180 00006573 668B4E02            <1>         mov     cx, word [esi+ptBeginSector]
181 00006577 66B80102            <1>       mov    ax, 0201h ; Read 1 sector
182                              <1>       ;mov   ebx, MasterBootBuff
183 0000657B E886DCFFFF          <1>       call   int13h
184 00006580 59                  <1>       pop    ecx
185 00006581 72C1                <1>       jc     short continue_to_check_ep
186                              <1> loc_hd_move_ep_table:
187                              <1>         ;pop edi
188                              <1>       ;push  edi  ; PTable_ep?
189 00006583 8B3C24              <1>       mov    edi, [esp]
190 00006586 BE[84560100]        <1>         mov  esi, PartitionTable ; Extended
191 0000658B 89F3                <1>       mov    ebx, esi
192                              <1>       ;mov   ecx, 16
193 0000658D B110                <1>       mov    cl, 16
194 0000658F F3A5                <1>              rep    movsd
195 00006591 89DE                <1>       mov    esi, ebx
196                              <1> loc_set_hde_sub_partition_count:
197 00006593 C605[C8580100]04    <1>       mov    byte [PP_Counter], 4
198                              <1> loc_validate_hde_partition:
199 0000659A 807E0400            <1>       cmp    byte [esi+ptFileSystemID], 0
200 0000659E 763F                <1>       jna    short loc_validate_next_hde_partition2
201 000065A0 56                  <1>       push   esi ; Extended partition table offset
202 000065A1 8A15[F55C0000]      <1>       mov    dl, byte [hdc]
203 000065A7 0FB6C2              <1>       movzx  eax, dl
204 000065AA 2C80                <1>       sub    al, 80h
205 000065AC C0E002              <1>       shl    al, 2
206                              <1>       ; 06/01/2016
207                              <1>       ; (TRDOS v1.0 had a bug here, in 'DRV_INIT.ASM')
208                              <1>       ; BUGFIX *
209                              <1>       ;mov   ecx, eax
210 000065AF 88C1                <1>       mov    cl, al
211 000065B1 80C104              <1>       add    cl, 4
212 000065B4 2A0D[C8580100]      <1>       sub    cl, [PP_Counter] ; 4 to 1
213                              <1>       ; CL = Partition entry/index, 0 based
214                              <1>         ;  0 -> hd 0, Partition Table offset = 0
215                              <1>         ; 15 -> hd 3, Partition Table offset = 3
216 000065BA 88D5                <1>            mov    ch, dl
217 000065BC 80ED7E              <1>       sub    ch, 7Eh ;
218                              <1>       ; CH = Physical drive index, zero based
219                              <1>       ;  0 for drive A:, 2 for drive C:
220                              <1>       ; BUGFIX *
221 000065BF 51                  <1>       push   ecx ; *
222 000065C0 BF[CA580100]        <1>       mov    edi, EP_StartSector
223 000065C5 01C7                <1>       add    edi, eax
224                              <1>       ; Input -> ESI = PartitionTable offset
225                              <1>       ; DL = Hard disk drive number
226                              <1>       ; EDI = Extended partition start sector pointer
227 000065C7 E82A000000          <1>       call   validate_hd_fat_partition
228 000065CC 59                  <1>       pop    ecx ; *
229 000065CD 720F                <1>       jc     short loc_validate_next_hde_partition1
230                              <1> loc_set_valid_hde_partition_entry:
231                              <1>       ; 06/01/2016 (TRDOS v2.0)
232                              <1>       ; BUGFIX *
233                              <1>       ;mov   [esi+LD_PartitionEntry], cl
234                              <1>       ;mov   [esi+LD_DParamEntry], ch
235 000065CF 66894E7C              <1>       mov    [esi+LD_PartitionEntry], cx
236                              <1>       ;
237 000065D3 8A0D[AC060100]      <1>       mov    cl, [Last_DOS_DiskNo]
238 000065D9 80C141              <1>       add    cl, 'A'
239 000065DC 880E                <1>       mov    [esi+LD_Name], cl
240                              <1> loc_validate_next_hde_partition1:
241 000065DE 5E                  <1>       pop    esi ; Extended partition table offset
242                              <1> loc_validate_next_hde_partition2:
243                              <1>       ; ESI = Extended partition table offset
244                              <1>       ; DL = Hard disk drive number
245 000065DF FE0D[C8580100]      <1>       dec    byte [PP_Counter]
246 000065E5 0F845EFFFFFF        <1>       jz     continue_check_ep_next_disk
247 000065EB 83C610              <1>       add    esi, 16 ; 10h
248 000065EE EBAA                <1>       jmp    short loc_validate_hde_partition
249                              <1> loc_validating_hd_partitions_ok:
250 000065F0 A0[AC060100]        <1>       mov    al, [Last_DOS_DiskNo]
251                              <1> loc_drv_init_retn:
252 000065F5 C3                  <1>       retn
253                              <1>
254                              <1> validate_hd_fat_partition:
255                              <1>       ; 12/02/2016
256                              <1>       ; 07/01/2016 (TRDOS 386 = TRDOS v2.0)
257                              <1>       ; 07/08/2011
258                              <1>       ; 23/07/2011
259                              <1>       ; Input
260                              <1>       ;   DL = Hard/Fixed Disk Drive Number
261                              <1>       ;   ESI = PartitionTable offset
262                              <1>       ;   EDI = Extend. Part. Start Sector Pointer
263                              <1>       ;   EDI = 0 -> Primary Partition
264                              <1>       ;   byte [Last_DOS_DiskNo]
265                              <1>       ; Output
266                              <1>       ;  cf=0 -> Validated
267                              <1>       ;   ESI = Logical dos drv desc. table
268                              <1>       ;   EBX = FAT boot sector buffer
269                              <1>       ;   byte [Last_DOS_DiskNo]
270                              <1>       ;  cf=1 -> Not a valid FAT partition
271                              <1>       ; EAX, EDX, ECX, EDI -> changed
272                              <1>
273                              <1>       ;mov   esi, PartitionTable
274 000065F6 8A6604              <1>       mov    ah, [esi+ptFileSystemID]
275 000065F9 80FC06              <1>       cmp    ah, 06h ; FAT16 CHS partition
276                              <1>       ; 12/02/2016
277                              <1>       ;jb    short loc_not_a_valid_fat_partition2
```

```
278 000065FC 7305                <1>       jnb    short vhdp_FAT16_32
279                              <1>       ;
280 000065FE 80FC04              <1>       cmp    ah, 04h ; FAT16 CHS partition (< 32MB)
281 00006601 7519                <1>       jne    short loc_not_a_valid_fat_partition1
282                              <1> vhdp_FAT16_32:
283 00006603 B002                <1>       mov    al, 2
284 00006605 7417                <1>       je     short loc_set_valid_hd_partition_params
285 00006607 80FC0E              <1>       cmp    ah, 0Eh ; FAT16 LBA partition
286 0000660A 7710                <1>       ja     short loc_not_a_valid_fat_partition1
287 0000660C 7410                <1>       je     short loc_set_valid_hd_partition_params
288                              <1>
289 0000660E FEC0                <1>       inc    al ; 3
290 00006610 80FC0B              <1>       cmp    ah, 0Bh ; FAT32 CHS partition
291 00006613 7409                <1>       je     short loc_set_valid_hd_partition_params
292 00006615 7206                <1>       jb     short loc_not_a_valid_fat_partition2
293 00006617 80FC0C              <1>       cmp    ah, 0Ch ; FAT32 LBA partition
294 0000661A 7402                <1>       je     short loc_set_valid_hd_partition_params
295                              <1> loc_not_a_valid_fat_partition1:
296 0000661C F9                  <1>       stc
297                              <1> loc_not_a_valid_fat_partition2:
298 0000661D C3                  <1>       retn
299                              <1>
300                              <1> loc_set_valid_hd_partition_params:
301 0000661E FE05[AC060100]      <1>       inc    byte [Last_DOS_DiskNo] ; > 1
302                              <1>       ;
303 00006624 31DB                <1>       xor    ebx, ebx
304 00006626 8A3D[AC060100]      <1>       mov    bh, [Last_DOS_DiskNo] ; * 256
305 0000662C 81C300010900        <1>       add    ebx, Logical_DOSDisks
306                              <1>       ;
307 00006632 C6430102            <1>       mov    byte [ebx+LD_DiskType], 2
308 00006636 885302              <1>       mov    byte [ebx+LD_PhyDrvNo], dl
309                              <1>       ;mov   byte [ebx+LD_FATType], al ; 2 or 3
310                              <1>       ;mov   byte [ebx+LD_FSType], ah ; 06h, 0Eh, 0Bh, 0Ch
311 00006639 66894303              <1>       mov    word [ebx+LD_FATType], ax
312                              <1>       ;
313 0000663D 8B4E08              <1>       mov    ecx, [esi+ptStartSector]
314 00006640 09FF                <1>       or     edi, edi
315 00006642 7402                <1>       jz     short pass_hd_FAT_ep_start_sector_adding
316                              <1> loc_add_hd_FAT_ep_start_sector:
317 00006644 030F                <1>       add    ecx, [edi]
318                              <1> pass_hd_FAT_ep_start_sector_adding:
319 00006646 894B6C              <1>       mov    [ebx+LD_StartSector], ecx
320                              <1> loc_hd_FAT_logical_drv_init:
321 00006649 89DD                <1>       mov    ebp, ebx
322                              <1>       ;mov   dl, [ebx+LD_PhyDrvNo]
323 0000664B A0[C7580100]        <1>       mov    al, [HD_LBA_yes] ; 07/01/2016
324 00006650 884305              <1>       mov    [ebx+LD_LBAYes], al
325 00006653 BB[DA580100]        <1>       mov    ebx, DOSBootSectorBuff ; buffer address
326 00006658 08C0                <1>       or     al, al
327 0000665A 740C                <1>       jz     short loc_hd_FAT_drv_init_load_bs_chs
328                              <1> loc_hd_FAT_drv_init_load_bs_lba:
329                              <1>       ; DL = Physical drive number
330                              <1>       ;mov   ecx, [esi+ptStartSector] ; sector number
331                              <1>       ;mov   ebx, DOSBootSectorBuff ; buffer address
332                              <1>       ; LBA read/write (with private LBA function)
333                              <1>       ;((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))
334                              <1>       ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)
335 0000665C B41B                <1>       mov    ah, 1Bh ; LBA read
336 0000665E B001                <1>       mov    al, 1 ; sector count
337 00006660 E8A1DBFFFF          <1>       call   int13h
338 00006665 7313                <1>       jnc    short loc_hd_drv_FAT_boot_validation
339                              <1> loc_not_a_valid_fat_partition3:
340 00006667 C3                  <1>       retn
341                              <1> loc_hd_FAT_drv_init_load_bs_chs:
342 00006668 8A7601              <1>       mov    dh, [esi+ptBeginHead]
343 0000666B 668B4E02            <1>       mov    cx, [esi+ptBeginSector]
344 0000666F 66B80102            <1>       mov    ax, 0201h ; Read 1 sector
345                              <1>       ;mov   ebx, DOSBootSectorBuff
346 00006673 E88EDBFFFF          <1>       call   int13h
347 00006678 72ED                <1>       jc     short loc_not_a_valid_fat_partition3
348                              <1> loc_hd_drv_FAT_boot_validation:
349                              <1>       ;mov   esi, DOSBootSectorBuff
350 0000667A 89DE                <1>       mov    esi, ebx
351 0000667C 6681BEFE01000055AA  <1>       cmp    word [esi+BS_Validation], 0AA55h
352 00006685 751A                <1>       jne    short loc_not_a_valid_fat_partition4
353 00006687 807E15F8            <1>       cmp    byte [esi+BPB_Media], 0F8h
354 0000668B 7514                <1>       jne    short loc_not_a_valid_fat_partition4
355 0000668D 66837E1600          <1>       cmp    word [esi+BPB_FATSz16], 0
356 00006692 770F                <1>       ja     short loc_hd_FAT16_BPB
357 00006694 807E4229            <1>       cmp    byte [esi+BS_FAT32_BootSig], 29h
358 00006698 7507                <1>       jne    short loc_not_a_valid_fat_partition4
359                              <1> loc_hd_FAT32_BPB:
360 0000669A B92D000000          <1>       mov    ecx, 45
361 0000669F EB0D                <1>       jmp    short loc_hd_move_FAT_BPB
362                              <1>       ;
363                              <1> loc_not_a_valid_fat_partition4:
364 000066A1 F9                  <1>       stc
365 000066A2 C3                  <1>       retn
366                              <1>       ;
367                              <1> loc_hd_FAT16_BPB:
368 000066A3 807E2629            <1>       cmp    byte [esi+BS_BootSig], 29h
369 000066A7 75F8                <1>       jne    short loc_not_a_valid_fat_partition4
370 000066A9 B920000000          <1>       mov    ecx, 32
371                              <1> loc_hd_move_FAT_BPB:
372 000066AE 89EF                <1>       mov    edi, ebp
373                              <1>       ;mov   esi, ebx ; Boot sector
374 000066B0 57                  <1>       push   edi
375 000066B1 83C706              <1>       add    edi, LD_BPB
376 000066B4 F366A5              <1>       rep    movsw
377 000066B7 5E                  <1>       pop    esi
378 000066B8 0FB74614            <1>       movzx  eax, word [esi+LD_BPB+BPB_RsvdSecCnt]
379 000066BC 03466C              <1>       add    eax, [esi+LD_StartSector]
```

```
380 000066BF 894660              <1>       mov    [esi+LD_FATBegin], eax
381 000066C2 807E0303            <1>       cmp    byte [esi+LD_FATType], 3
382 000066C6 7224                <1>       jb     short loc_set_FAT16_RootDirLoc
383                              <1> loc_set_FAT32_RootDirLoc:
384 000066C8 8B462A              <1>       mov    eax, [esi+LD_BPB+BPB_FATSz32]
385 000066CB 0FB65E16            <1>         movzx      ebx, byte [esi+LD_BPB+BPB_NumFATs]
386 000066CF F7E3                <1>       mul    ebx
387 000066D1 034660              <1>       add    eax, [esi+LD_FATBegin]
388                              <1> loc_set_FAT32_data_begin:
389 000066D4 894668              <1>       mov    [esi+LD_DATABegin], eax
390 000066D7 894664              <1>       mov    [esi+LD_ROOTBegin], eax
391                              <1>       ; If Root Directory Cluster <> 2 then
392                              <1>       ; change the beginning sector value
393                              <1>       ; of the root dir by adding sector offset.
394 000066DA 8B4632              <1>       mov    eax, [esi+LD_BPB+BPB_RootClus]
395 000066DD 83E802              <1>       sub    eax, 2
396 000066E0 7442                <1>       jz     short short loc_set_32bit_FAT_total_sectors
397                              <1>       ;movzx ebx, byte [esi+LD_BPB+BPB_SecPerClust]
398 000066E2 8A5E13              <1>       mov    bl, byte [esi+LD_BPB+BPB_SecPerClust]
399 000066E5 F7E3                <1>       mul    ebx
400 000066E7 014664              <1>       add    [esi+LD_ROOTBegin], eax
401 000066EA EB38                <1>       jmp    short loc_set_32bit_FAT_total_sectors
402                              <1>       ;
403                              <1> loc_set_FAT16_RootDirLoc:
404 000066EC 0FB64616            <1>       movzx  eax, byte [esi+LD_BPB+BPB_NumFATs]
405 000066F0 0FB7561C            <1>       movzx  edx, word [esi+LD_BPB+BPB_FATSz16]
406 000066F4 F7E2                <1>       mul    edx
407 000066F6 034660              <1>       add    eax, [esi+LD_FATBegin]
408 000066F9 894664              <1>       mov    [esi+LD_ROOTBegin], eax
409                              <1> loc_set_FAT16_data_begin:
410 000066FC 894668              <1>       mov    [esi+LD_DATABegin], eax
411 000066FF B820000000          <1>       mov    eax, 20h  ; Size of a directory entry
412                              <1>       ;movzx edx, word [esi+LD_BPB+BPB_RootEntCnt]
413 00006704 668B5617            <1>         mov     dx, [esi+LD_BPB+BPB_RootEntCnt]
414 00006708 F7E2                <1>         mul  edx
415                              <1>       ;mov   ecx, 511
416 0000670A 66B9FF01            <1>       mov    cx, 511
417 0000670E 01C8                <1>       add    eax, ecx
418 00006710 41                  <1>       inc    ecx ; 512
419 00006711 F7F1                <1>       div    ecx
420 00006713 014668              <1>       add    [esi+LD_DATABegin], eax
421 00006716 0FB74619            <1>       movzx  eax, word [esi+LD_BPB+BPB_TotalSec16]
422 0000671A 6685C0              <1>       test   ax, ax
423 0000671D 7405                <1>       jz     short loc_set_32bit_FAT_total_sectors
424                              <1> loc_set_16bit_FAT_total_sectors:
425 0000671F 894670              <1>       mov    [esi+LD_TotalSectors], eax
426 00006722 EB06                <1>       jmp    short loc_set_hd_FAT_cluster_count
427                              <1> loc_set_32bit_FAT_total_sectors:
428 00006724 8B4626              <1>       mov    eax, [esi+LD_BPB+BPB_TotalSec32]
429 00006727 894670              <1>       mov    [esi+LD_TotalSectors], eax
430                              <1> loc_set_hd_FAT_cluster_count:
431 0000672A 8B5668              <1>       mov    edx, [esi+LD_DATABegin]
432 0000672D 2B566C              <1>       sub    edx, [esi+LD_StartSector]
433 00006730 29D0                <1>       sub    eax, edx
434 00006732 31D2                <1>       xor    edx, edx ; 0
435 00006734 0FB64E13            <1>         movzx   ecx, byte [esi+LD_BPB+BPB_SecPerClust]
436 00006738 F7F1                <1>         div  ecx
437 0000673A 894678              <1>       mov    [esi+LD_Clusters], eax
438                              <1>       ; Maximum Valid Cluster Number= EAX +1
439                              <1>       ; with 2 reserved clusters= EAX +2
440                              <1> loc_set_hd_FAT_fs_free_sectors:
441                              <1>       ;mov   dword [esi+LD_FreeSectors], 0
442 0000673D E859010000            <1>       call   get_free_FAT_sectors
443 00006742 7207                <1>       jc     short loc_validate_hd_FAT_partition_retn
444 00006744 894674              <1>       mov    [esi+LD_FreeSectors], eax
445 00006747 C6467E06            <1>       mov    byte [esi+LD_MediaChanged], 6  ; Volume Name Reset
446                              <1>       ;mov   cl, [Last_DOS_DiskNo]
447                              <1>       ;add   cl, 'A'
448                              <1>       ;mov   [esi+LD_FS_Name], cl
449                              <1>
450                              <1> loc_validate_hd_FAT_partition_retn:
451 0000674B C3                  <1>       retn
452                              <1>
453                              <1> validate_hd_fs_partition:
454                              <1>       ; 13/02/2016
455                              <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
456                              <1>       ; 29/01/2011
457                              <1>       ; 23/07/2011
458                              <1>       ; Input
459                              <1>       ;   DL = Hard/Fixed Disk Drive Number
460                              <1>       ;   ESI = PartitionTable offset
461                              <1>       ;   byte [Last_DOS_DiskNo]
462                              <1>       ; Output
463                              <1>       ;  cf=0 -> Validated
464                              <1>       ;   ESI = Logical dos drv desc. table
465                              <1>       ;   EBX = Singlix FS boot sector buffer
466                              <1>       ;   byte [Last_DOS_DiskNo]
467                              <1>       ;  cf=1 -> Not a valid 'Singlix FS' partition
468                              <1>       ; EAX, EDX, ECX, EDI -> changed
469                              <1>
470                              <1>       ;mov   esi, PartitionTable
471 0000674C 8A6604              <1>       mov    ah, [esi+ptFileSystemID]
472 0000674F 80FCA1              <1>       cmp    ah, 0A1h ; SINGLIX FS1 (trfs1) partition
473 00006752 7549                <1>       jne    short loc_validate_hd_fs_partition_stc_retn
474                              <1> loc_set_valid_hd_fs_partition_params:
475 00006754 FE05[AC060100]      <1>       inc    byte [Last_DOS_DiskNo] ; > 1
476 0000675A 30C0                <1>       xor    al, al ; mov al, 0
477                              <1>       ;mov   [drv], dl
478 0000675C 29DB                <1>       sub    ebx, ebx ; 0
479 0000675E 8A3D[AC060100]      <1>       mov    bh, [Last_DOS_DiskNo]
480 00006764 81C300010900        <1>       add    ebx, Logical_DOSDisks
481 0000676A C6430102            <1>       mov    byte [ebx+LD_DiskType], 2
```

```
482 0000676E 885302              <1>       mov    [ebx+LD_PhyDrvNo], dl
483                              <1>       ;mov   [ebx+LD_FATType], al ; 0
484                              <1>       ;mov   [ebx+LD_FSType], ah
485 00006771 66894303            <1>       mov    [ebx+LD_FATType], ax
486                              <1>       ;mov   eax, [esi+ptStartSector]
487                              <1>       ;mov   [ebx+LD_StartSector], eax
488                              <1> loc_hd_fs_logical_drv_init:
489 00006775 89DD                <1>       mov    ebp, ebx ; 10/01/2016
490                              <1>       ;mov   dl, [ebx+LD_PhyDrvNo]
491 00006777 A0[C7580100]        <1>       mov    al, [HD_LBA_yes] ; 10/01/2016
492 0000677C 884305              <1>       mov    [ebx+LD_LBAYes], al
493 0000677F 89DE                <1>       mov    esi, ebx
494 00006781 BB[DA580100]        <1>       mov    ebx, DOSBootSectorBuff ; buffer address
495 00006786 08C0                <1>       or     al, al
496 00006788 7515                <1>       jnz    short loc_hd_fs_drv_init_load_bs_lba
497                              <1> loc_hd_fs_drv_init_load_bs_chs:
498 0000678A 8A7601              <1>       mov    dh, [esi+ptBeginHead]
499 0000678D 668B4E02            <1>       mov    cx, [esi+ptBeginSector]
500 00006791 66B80102            <1>       mov    ax, 0201h ; Read 1 sector
501                              <1>       ;mov   ebx, DOSBootSectorBuff
502 00006795 E86CDAFFFF          <1>       call   int13h
503 0000679A 7311                <1>       jnc    short loc_hd_drv_fs_boot_validation
504                              <1> loc_validate_hd_fs_partition_err_retn:
505 0000679C C3                  <1>       retn
506                              <1> loc_validate_hd_fs_partition_stc_retn:
507 0000679D F9                  <1>       stc
508 0000679E C3                  <1>       retn
509                              <1> loc_hd_fs_drv_init_load_bs_lba:
510                              <1>       ; DL = Physical drive number
511                              <1>       ;mov   esi, ebx
512 0000679F 8B4E08              <1>       mov    ecx, [esi+ptStartSector] ; sector number
513                              <1>       ;mov   ebx, DOSBootSectorBuff ; buffer address
514                              <1>       ; LBA read/write (with private LBA function)
515                              <1>       ;((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))
516                              <1>       ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)
517 000067A2 B41B                <1>       mov    ah, 1Bh ; LBA read
518 000067A4 B001                <1>       mov    al, 1 ; sector count
519 000067A6 E85BDAFFFF          <1>       call   int13h
520 000067AB 72EF                <1>       jc     short loc_validate_hd_fs_partition_err_retn
521                              <1> loc_hd_drv_fs_boot_validation:
522                              <1>       ;mov   esi, DOSBootSectorBuff
523 000067AD 89DE                <1>       mov    esi, ebx ; Boot sector buffer
524 000067AF 6681BEFE01000055AA  <1>       cmp    word [esi+BS_Validation], 0AA55h
525 000067B8 75E3                <1>       jne    short loc_validate_hd_fs_partition_stc_retn
526                              <1>         ;
527                              <1>       ;Singlix FS Extensions to TR-DOS (7/6/2009)
528 000067BA 66817E035346        <1>       cmp    word [esi+bs_FS_Identifier], 'SF'
529 000067C0 75DB                <1>       jne    short loc_validate_hd_fs_partition_stc_retn
530                              <1>         ;'A1h' check is not necessary
531                              <1>       ;  if 'FS' check is passed as OK/Yes.
532 000067C2 807E09A1            <1>       cmp    byte [esi+bs_FS_PartitionID], 0A1h
533 000067C6 75D5                <1>       jne    short loc_validate_hd_fs_partition_stc_retn
534                              <1>       ;
535 000067C8 89EF                <1>       mov    edi, ebp ; 10/01/2016
536                              <1>       ;
537 000067CA 8A462D              <1>       mov    al, byte [esi+bs_FS_LBA_Ready]
538 000067CD 884705              <1>       mov    [edi+LD_FS_LBAYes], al
539                              <1>       ;
540                              <1>       ; 03/01/2010 CHS -> DOS FAT/BPB compatibility fix
541 000067D0 8A4608              <1>       mov    al, [esi+bs_FS_MediaAttrib]
542 000067D3 884706              <1>       mov    byte [edi+LD_FS_MediaAttrib], al
543                              <1>       ;
544 000067D6 8A460A              <1>       mov    al, [esi+bs_FS_VersionMaj]
545 000067D9 884707              <1>       mov    [edi+LD_FS_VersionMajor], al
546                              <1>       ;
547 000067DC 668B4606            <1>       mov    ax, [esi+bs_FS_BytesPerSec]
548 000067E0 66894711            <1>       mov    [edi+LD_FS_BytesPerSec], ax
549 000067E4 8A462E              <1>       mov    al, [esi+bs_FS_SecPerTrack]
550 000067E7 6698                <1>       cbw
551 000067E9 6689471E            <1>       mov    [edi+LD_FS_SecPerTrack], ax
552 000067ED 8A462F              <1>       mov    al, [esi+bs_FS_Heads]
553                              <1>       ;cbw
554 000067F0 66894720            <1>       mov    [edi+LD_FS_NumHeads], ax
555                              <1>       ;
556 000067F4 8B4628              <1>       mov    eax, [esi+bs_FS_UnDelDirD]
557 000067F7 894722              <1>       mov    [edi+LD_FS_UnDelDirD], eax
558 000067FA 8B5618              <1>       mov    edx, [esi+bs_FS_MATLocation]
559 000067FD 89570C              <1>       mov    [edi+LD_FS_MATLocation], edx
560 00006800 8B461C              <1>       mov    eax, [esi+bs_FS_RootDirD]
561 00006803 894708              <1>       mov    [edi+LD_FS_RootDirD], eax
562 00006806 8B460C              <1>       mov    eax, [esi+bs_FS_BeginSector]
563 00006809 89476C              <1>       mov    [edi+LD_FS_BeginSector], eax
564 0000680C 8B4710              <1>       mov    eax, [edi+bs_FS_VolumeSize]
565 0000680F 894770              <1>       mov    [edi+LD_FS_VolumeSize], eax
566                              <1>       ;
567 00006812 89D0                <1>       mov    eax, edx ; [edi+LD_FS_MATLocation]
568 00006814 03476C              <1>       add    eax, [edi+LD_FS_BeginSector]
569 00006817 89FE                <1>       mov    esi, edi
570                              <1> mread_hd_fs_MAT_sector:
571                              <1>         ;mov  ebx, DOSBootSectorBuff
572 00006819 B901000000          <1>       mov    ecx, 1
573 0000681E E88F890000          <1>       call   disk_read
574 00006823 7248                <1>       jc     short loc_validate_hd_fs_partition_retn
575                              <1>       ; EDI will not be changed
576 00006825 89DE                <1>       mov    esi, ebx
577                              <1> use_hdfs_mat_sector_params:
578 00006827 8B460C              <1>       mov    eax, [esi+FS_MAT_DATLocation]
579 0000682A 894714              <1>       mov    [edi+LD_FS_DATLocation], eax
580 0000682D 8B4610              <1>       mov    eax, [esi+FS_MAT_DATScount]
581 00006830 894718              <1>       mov    [edi+LD_FS_DATSectors], eax
582 00006833 8B4614              <1>       mov    eax, [esi+FS_MAT_FreeSectors]
583 00006836 894774              <1>         mov     [edi+LD_FS_FreeSectors], eax
```

```
584 00006839 8B4618              <1>       mov    eax, [esi+FS_MAT_FirstFreeSector]
585 0000683C 894778              <1>       mov    [edi+LD_FS_FirstFreeSector], eax
586 0000683F 8B4708              <1>       mov    eax, [edi+LD_FS_RootDirD]
587 00006842 03476C              <1>       add    eax, [edi+LD_FS_BeginSector]
588 00006845 89FE                <1>       mov    esi, edi
589                              <1> read_hd_fs_RDT_sector:
590 00006847 BB[DA580100]        <1>       mov    ebx, DOSBootSectorBuff
591                              <1>       ;mov   ecx, 1
592 0000684C B101                <1>       mov    cl, 1
593 0000684E E85F890000          <1>       call   disk_read
594 00006853 7218                <1>       jc     short loc_validate_hd_fs_partition_retn
595                              <1>       ; EDI will not be changed
596 00006855 89DE                <1>       mov    esi, ebx
597                              <1> use_hdfs_RDT_sector_params:
598 00006857 8B461C              <1>       mov    eax, [esi+FS_RDT_VolumeSerialNo]
599 0000685A 894728              <1>       mov    [edi+LD_FS_VolumeSerial], eax
600 0000685D 57                  <1>       push   edi
601                              <1>       ;mov   ecx, 16
602 0000685E B110                <1>       mov    cl, 16
603 00006860 83C640              <1>       add    esi, FS_RDT_VolumeName
604 00006863 83C72C              <1>       add    edi, LD_FS_VolumeName
605 00006866 F3A5                <1>       rep    movsd ; 64 bytes
606 00006868 5E                  <1>       pop    esi
607                              <1>              ; Volume Name Reset
608 00006869 C6467E06            <1>         mov     byte [esi+LD_FS_MediaChanged], 6
609                              <1>       ;
610                              <1>         ;mov cl, [Last_DOS_DiskNo]
611                              <1>       ;add   cl, 'A'
612                              <1>       ;mov   [esi+LD_FS_Name], cl
613                              <1>
614                              <1> loc_validate_hd_fs_partition_retn:
615 0000686D C3                  <1>       retn
616                              <1>
617                              <1> load_masterboot:
618                              <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
619                              <1>       ; 2005 - 2011
620                              <1>       ; input -> DL = drive number
621 0000686E B40D                <1>       mov    ah, 0Dh ; Alternate disk reset
622 00006870 E891D9FFFF          <1>       call   int13h
623 00006875 7301                <1>       jnc    short pass_reset_error
624                              <1> harddisk_error:
625 00006877 C3                  <1>       retn
626                              <1> pass_reset_error:
627 00006878 BB[C6540100]        <1>       mov    ebx, MasterBootBuff
628 0000687D 66B80102            <1>       mov    ax, 0201h
629 00006881 66B90100            <1>       mov    cx, 1
630 00006885 30F6                <1>       xor    dh, dh
631 00006887 E87AD9FFFF          <1>       call   int13h
632 0000688C 72E9                <1>       jc     short harddisk_error
633                              <1>       ;
634 0000688E 66813D[C4560100]55- <1>       cmp    word [MBIDCode], 0AA55h
634 00006896 AA                  <1>
635 00006897 7401                <1>       je     short load_masterboot_ok
636 00006899 F9                  <1>       stc
637                              <1> load_masterboot_ok:
638 0000689A C3                  <1>       retn
639                              <1>
640                              <1> get_free_FAT_sectors:
641                              <1>       ; 29/02/2016
642                              <1>       ; 13/02/2016
643                              <1>       ; 04/02/2016
644                              <1>       ; 07/01/2016 (TRDOS 386 = TRDOS v2.0)
645                              <1>       ; 11/07/2010
646                              <1>       ; 21/06/2009
647                              <1>       ; INPUT: ESI = Logical DOS Drive Description Table address
648                              <1>       ; OUTPUT: STC => Error
649                              <1>         ;    cf = 0 and EAX = Free FAT sectors
650                              <1>       ; Also, related parameters and FAT buffer will be reset and updated
651                              <1>
652 0000689B 31C0                <1>       xor    eax, eax
653                              <1>       ;mov   [esi+LD_FreeSectors], eax ; Reset
654                              <1>
655 0000689D 807E0302            <1>         cmp     byte [esi+LD_FATType], 2
656 000068A1 7650                <1>       jna    short loc_gfc_get_fat_free_clusters
657                              <1>
658                              <1>       ; 29/02/2016
659 000068A3 48                  <1>       dec    eax ; 0FFFFFFFFh
660 000068A4 89463A              <1>       mov    [esi+LD_BPB+BPB_Reserved], eax ; Free cluster count (reset)
661 000068A7 89463E              <1>       mov    [esi+LD_BPB+BPB_Reserved+4], eax ; First Free Cluster (reset)
662 000068AA 40                  <1>       inc    eax ; 0
663                              <1>       ;
664 000068AB 668B4636            <1>       mov    ax, [esi+LD_BPB+BPB_FSInfo]
665 000068AF 03466C              <1>       add    eax, [esi+LD_StartSector]
666                              <1>
667 000068B2 BB[DA580100]        <1>       mov    ebx, DOSBootSectorBuff
668 000068B7 B901000000          <1>       mov    ecx, 1
669 000068BC E8F1880000          <1>       call   disk_read
670 000068C1 7301                <1>       jnc    short loc_gfc_check_fsinfo_signs
671                              <1> retn_gfc_get_fsinfo_sec:
672 000068C3 C3                  <1>       retn
673                              <1>
674                              <1> loc_gfc_check_fsinfo_signs:
675 000068C4 BB[DA580100]        <1>       mov    ebx, DOSBootSectorBuff ; 13/02/2016
676 000068C9 813B52526141        <1>         cmp     dword [ebx], 41615252h
677 000068CF 7520                <1>       jne    short retn_gfc_get_fsinfo_stc
678                              <1>       ;add   ebx, 484
679                              <1>       ;cmp   dword [ebx], 61417272h
680 000068D1 81BBE4010000727241- <1>       cmp    dword [ebx+484], 61417272h
680 000068DA 61                  <1>
681 000068DB 7514                <1>       jne    short retn_gfc_get_fsinfo_stc
682                              <1>       ;add   ebx, 4
683                              <1>       ;mov   eax, [ebx]
```

```
000068DD 8B83E8010000        <1>       mov    eax, [ebx+488]
                             <1>       ; 29/02/2016
000068E3 89463A              <1>       mov    [esi+LD_BPB+BPB_Reserved], eax ; Free cluster count
000068E6 8B93EC010000        <1>       mov    edx,  [ebx+492]
000068EC 89463E              <1>       mov    [esi+LD_BPB+BPB_Reserved+4], eax ; First Free Cluster
                             <1>       ;
000068EF EB12                <1>       jmp    short retn_from_get_free_fat32_clusters
                             <1>
                             <1> retn_gfc_get_fsinfo_stc:
000068F1 F9                  <1>       stc
000068F2 C3                  <1>       retn
                             <1>
                             <1> loc_gfc_get_fat_free_clusters:
                             <1>       ;mov   eax, 2
000068F3 B002                <1>       mov    al, 2
                             <1>       ;mov   [FAT_CurrentCluster], eax
                             <1> loc_gfc_loop_get_next_cluster:
000068F5 E823500000          <1>       call   get_next_cluster
000068FA 730E                <1>       jnc    short loc_gfc_free_fat_clusters_cont
000068FC 21C0                <1>       and    eax, eax
000068FE 7411                <1>       jz     short loc_gfc_pass_inc_free_cluster_count
                             <1>
                             <1> retn_from_get_free_fat_clusters:
00006900 8B4674              <1>       mov    eax, [esi+LD_FreeSectors] ; Free clusters !
                             <1> retn_from_get_free_fat32_clusters:
00006903 0FB65E13            <1>         movzx      ebx, byte [esi+LD_BPB+BPB_SecPerClust]
00006907 F7E3                <1>             mul    ebx
                             <1>       ;mov   [esi+LD_FreeSectors], eax ; Free sectors
                             <1> retn_get_free_sectors_calc:
00006909 C3                  <1>       retn
                             <1>
                             <1> loc_gfc_free_fat_clusters_cont:
0000690A 09C0                <1>       or     eax, eax
0000690C 7503                <1>       jnz    short loc_gfc_pass_inc_free_cluster_count
0000690E FF4674              <1>       inc    dword [esi+LD_FreeSectors] ; Free clusters !
                             <1>
                             <1> loc_gfc_pass_inc_free_cluster_count:
                             <1>       ;mov   eax, [FAT_CurrentCluster]
00006911 89C8                <1>       mov    eax, ecx ; [FAT_CurrentCluster]
00006913 3B4678              <1>       cmp    eax, [esi+LD_Clusters]
00006916 77E8                <1>       ja     short retn_from_get_free_fat_clusters
00006918 40                  <1>       inc    eax
                             <1>       ;mov   [FAT_CurrentCluster], eax
00006919 EBDA                <1>       jmp    short loc_gfc_loop_get_next_cluster
                             <1>
                             <1> floppy_drv_init:
                             <1>       ; 06/07/2016
                             <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 24/07/2011
                             <1>       ; 04/07/2009
                             <1>       ; INPUT ->
                             <1>       ;      DL = Drive number (0,1)
                             <1>       ; OUTPUT ->
                             <1>       ;      BL = drive name
                             <1>       ;      BH = drive number
                             <1>       ;      ESI = Logical DOS drv description table
                             <1>       ;      EAX = Volume serial number
                             <1>
0000691B BE[F65C0000]        <1>       mov    esi, fd0_type ; 10/01/2016
00006920 BF00010900          <1>       mov    edi, Logical_DOSDisks
00006925 08D2                <1>       or     dl, dl
00006927 7407                <1>       jz     short loc_drv_init_fd0_fd1
00006929 81C700010000        <1>       add    edi, 100h
0000692F 46                  <1>       inc    esi ; fd1_type ; 10/01/2016
                             <1> loc_drv_init_fd0_fd1:
00006930 C6477E00            <1>       mov    byte [edi+LD_MediaChanged], 0
00006934 803E01              <1>       cmp    byte [esi], 1 ; type (>0 if it is existing)
                             <1>                            ; 4 = 1.44 MB, 80 track, 3 1/2"
00006937 7221                <1>       jb     short read_fd_boot_sector_retn
00006939 885702              <1>       mov    [edi+LD_PhyDrvNo], dl
                             <1> read_fd_boot_sector:
0000693C 30F6                <1>       xor    dh, dh
0000693E B904000000          <1>       mov    ecx, 4 ; Retry Count
                             <1> read_fd_boot_sector_again:
00006943 51                  <1>       push   ecx
                             <1>       ;mov   cx, 1
00006944 B101                <1>       mov    cl, 1
00006946 66B80102            <1>       mov    ax, 0201h ; Read 1 sector
0000694A BB[DA580100]        <1>       mov    ebx, DOSBootSectorBuff
0000694F E8B2D8FFFF          <1>       call   int13h
00006954 59                  <1>       pop    ecx
00006955 7304                <1>       jnc    short use_fd_boot_sector_params
00006957 E2EA                <1>       loop   read_fd_boot_sector_again
                             <1>
                             <1> read_fd_boot_sector_stc_retn:
00006959 F9                  <1>       stc
                             <1> read_fd_boot_sector_retn:
0000695A C3                  <1>       retn
                             <1>
                             <1> use_fd_boot_sector_params:
                             <1>       ;mov   esi, DOSBootSectorBuff
0000695B 89DE                <1>       mov    esi, ebx
0000695D 6681BEFE01000055AA  <1>       cmp    word [esi+BS_Validation], 0AA55h
00006966 75F1                <1>       jne    short read_fd_boot_sector_stc_retn
00006968 66817E035346        <1>         cmp     word [esi+bs_FS_Identifier], 'SF'
0000696E 0F85A2000000        <1>         jne     use_fd_fatfs_boot_sector_params
                             <1>       ;
00006974 8A462D              <1>       mov    al, [esi+bs_FS_LBA_Ready]
00006977 884705              <1>       mov    [edi+LD_FS_LBAYes], al
                             <1>       ;
                             <1>       ; 03/01/2010 CHS -> DOS FAT/BPB compatibility fix
0000697A 8A4608              <1>       mov    al, [esi+bs_FS_MediaAttrib]
```

```
786 0000697D 884706             <1>       mov   [edi+LD_FS_MediaAttrib], al
787                             <1>       ;
788 00006980 8A460A             <1>         mov  al, [esi+bs_FS_VersionMaj]
789 00006983 884707             <1>       mov   byte [edi+LD_FS_VersionMajor], al
790 00006986 668B4606           <1>       mov   ax, [esi+bs_FS_BytesPerSec]
791 0000698A 66894711           <1>       mov   [edi+LD_FS_BytesPerSec], ax
792 0000698E 8A462E             <1>       mov   al, [esi+bs_FS_SecPerTrack]
793 00006991 6698               <1>       cbw
794 00006993 6689471E           <1>       mov   [edi+LD_FS_SecPerTrack], ax
795 00006997 8A462F             <1>       mov   al, [esi+bs_FS_Heads]
796                             <1>       ;cbw
797 0000699A 66894720           <1>       mov   [edi+LD_FS_NumHeads], ax
798                             <1>       ;
799 0000699E 8B4628             <1>       mov   eax, [esi+bs_FS_UnDelDirD]
800 000069A1 894722             <1>       mov   [edi+LD_FS_UnDelDirD], eax
801 000069A4 8B4618             <1>       mov   eax, [esi+bs_FS_MATLocation]
802 000069A7 89470C             <1>       mov   [edi+LD_FS_MATLocation], eax
803 000069AA 8B461C             <1>       mov   eax, [esi+bs_FS_RootDirD]
804 000069AD 894708             <1>       mov   [edi+LD_FS_RootDirD], eax
805 000069B0 8B460C             <1>       mov   eax, [esi+bs_FS_BeginSector]
806 000069B3 89476C             <1>       mov   [edi+LD_FS_BeginSector], eax
807 000069B6 8B4610             <1>       mov   eax, [esi+bs_FS_VolumeSize]
808 000069B9 894770             <1>       mov   [edi+LD_FS_VolumeSize], eax
809                             <1>       ;
810 000069BC 89FE               <1>       mov   esi, edi
811 000069BE 8B460C             <1>       mov   eax, [esi+LD_FS_MATLocation]
812                             <1>       ;add   eax, [edi+LD_FS_BeginSector]
813                             <1> read_fd_MAT_sector_again:
814                             <1>       ;mov   ebx, DOSBootSectorBuff
815                             <1>       ;mov   ecx, 1
816 000069C1 B101               <1>       mov   cl, 1
817 000069C3 E8F0870000         <1>       call  chs_read
818 000069C8 89DE               <1>       mov   esi, ebx
819 000069CA 7301               <1>       jnc   short use_fdfs_mat_sector_params
820                             <1>       ;jmp  short read_fd_boot_sector_retn
821 000069CC C3                 <1>       retn
822                             <1> use_fdfs_mat_sector_params:
823 000069CD 8B460C             <1>       mov   eax, [esi+FS_MAT_DATLocation]
824 000069D0 894714             <1>       mov   [edi+LD_FS_DATLocation], eax
825 000069D3 8B4610             <1>       mov   eax, [esi+FS_MAT_DATScount]
826 000069D6 894718             <1>       mov   [edi+LD_FS_DATSectors], eax
827 000069D9 8B4714             <1>       mov   eax, [edi+FS_MAT_FreeSectors]
828 000069DC 894774             <1>       mov   [edi+LD_FS_FreeSectors], eax
829 000069DF 8B4618             <1>       mov   eax, [esi+FS_MAT_FirstFreeSector]
830 000069E2 894778             <1>       mov   [edi+LD_FS_FirstFreeSector], eax
831                             <1>       ;
832 000069E5 89FE               <1>       mov   esi, edi
833 000069E7 8B4608             <1>       mov   eax, [esi+LD_FS_RootDirD]
834                             <1> read_fd_RDT_sector_again:
835                             <1>       ;mov   ebx, DOSBootSectorBuff
836                             <1>       ;mov   cx, 1
837 000069EA B101               <1>       mov   cl, 1
838 000069EC E8C7870000         <1>       call  chs_read
839 000069F1 89DE               <1>       mov   esi, ebx
840 000069F3 7220               <1>       jc    short read_fd_RDT_sector_retn
841                             <1> use_fdfs_RDT_sector_params:
842 000069F5 8B461C             <1>       mov   eax, [esi+FS_RDT_VolumeSerialNo]
843 000069F8 894728             <1>       mov   [edi+LD_FS_VolumeSerial], eax
844 000069FB 57                 <1>       push  edi
845                             <1>       ;mov   ecx, 16
846 000069FC B110               <1>       mov   cl, 16
847 000069FE 83C640             <1>       add   esi, FS_RDT_VolumeName
848 00006A01 83C72C             <1>       add   edi, LD_FS_VolumeName
849 00006A04 F3A5               <1>       rep   movsd ; 64 bytes
850 00006A06 5E                 <1>       pop   esi
851 00006A07 C6460300           <1>       mov   byte [esi+LD_FATType], 0
852 00006A0B C64604A1           <1>       mov   byte [esi+LD_FSType], 0A1h
853 00006A0F E9A5000000         <1>         jmp     loc_cont_use_fd_boot_sector_params
854                             <1>
855                             <1> read_fd_RDT_sector_stc_retn:
856 00006A14 F9                 <1>       stc
857                             <1> read_fd_RDT_sector_retn:
858 00006A15 C3                 <1>       retn
859                             <1>
860                             <1> use_fd_fatfs_boot_sector_params:
861 00006A16 807E2629           <1>       cmp   byte [esi+BS_BootSig], 29h
862 00006A1A 75F8               <1>       jne   short read_fd_RDT_sector_stc_retn
863 00006A1C 807E15F0           <1>       cmp   byte [esi+BPB_Media], 0F0h
864 00006A20 72F3               <1>       jb    short read_fd_RDT_sector_retn
865 00006A22 57                 <1>       push  edi
866 00006A23 83C706             <1>       add   edi, LD_BPB
867                             <1>       ;mov   ecx, 16
868 00006A26 B110               <1>       mov   cl, 16
869 00006A28 F3A5               <1>       rep   movsd ; 64 bytes
870 00006A2A 5E                 <1>       pop   esi
871 00006A2B 31C0               <1>       xor   eax, eax
872 00006A2D 89466C             <1>       mov   [esi+LD_StartSector], eax ; 0
873 00006A30 668B461C           <1>       mov   ax, [esi+LD_BPB+BPB_FATSz16]
874 00006A34 8A4E16             <1>       mov   cl, [esi+LD_BPB+BPB_NumFATs]
875 00006A37 F7E1               <1>       mul   ecx
876                             <1>       ; edx = 0 !
877 00006A39 668B5614           <1>       mov   dx, [esi+LD_BPB+BPB_RsvdSecCnt]
878 00006A3D 66895660           <1>       mov   [esi+LD_FATBegin], dx
879                             <1>       ;add   eax, edx
880 00006A41 6601D0             <1>       add   ax, dx
881 00006A44 894664             <1>       mov   [esi+LD_ROOTBegin], eax
882 00006A47 894668             <1>       mov   [esi+LD_DATABegin], eax
883 00006A4A 668B5617           <1>       mov   dx, [esi+LD_BPB+BPB_RootEntCnt]
884                             <1>       ;;shl  edx, 5 ; * 32 (Size of a directory entry)
885                             <1>       ;shl   dx, 5
886                             <1>       ;;add  edx, 511
887                             <1>       ;add   dx, 511
```

```
                        <1>       ;;shr  edx, 9 ; edx = ((edx*32)+511) / 512
                        <1>       ;shr   dx, 9
00006A4E 6683C20F       <1>       add    dx, 15 ; 06/07/2016 (+(512/32)-1)
00006A52 66C1EA04       <1>       shr    dx, 4 ; / 16 (==16 entries per sector)
00006A56 015668         <1>       add    [esi+LD_DATABegin], edx ; + rd sectors
                        <1>       ;movzx eax, word [esi+LD_BPB+BPB_TotalSec16]
00006A59 668B4619       <1>       mov    ax, [esi+LD_BPB+BPB_TotalSec16]
00006A5D 894670         <1>       mov    [esi+LD_TotalSectors], eax
00006A60 2B4668         <1>       sub    eax, [esi+LD_DATABegin]
                        <1>       ;movzx ecx, byte [esi+LD_BPB+BPB_SecPerClust]
00006A63 8A4E13         <1>       mov    cl, [esi+LD_BPB+BPB_SecPerClust]
00006A66 80F901         <1>       cmp    cl, 1
00006A69 7605           <1>       jna    short save_fd_fatfs_cluster_count
                        <1>       ;sub   edx, edx
00006A6B 6629D2         <1>       sub    dx, dx ; 0
                        <1>       ;sub   dl, dl ; 06/07/2016
00006A6E F7F1           <1>       div    ecx
                        <1> save_fd_fatfs_cluster_count:
00006A70 894678         <1>       mov    [esi+LD_Clusters], eax
                        <1>
                        <1>       ; Maximum Valid Cluster Number = EAX +1
                        <1>       ; with 2 reserved clusters= EAX +2
                        <1>
                        <1> reset_FAT_buffer_decriptors:
00006A73 29C0           <1>       sub    eax, eax ; 0
00006A75 A2[DE5A0100]   <1>       mov    [FAT_BuffValidData], al ; 0
00006A7A A2[DF5A0100]   <1>       mov    [FAT_BuffDrvName], al ; 0
00006A7F A3[E25A0100]   <1>       mov    [FAT_BuffSector], eax ; 0
                        <1>
                        <1> read_fd_FAT_sectors:
00006A84 BB001C0900     <1>       mov    ebx, FAT_Buffer
00006A89 668B4614       <1>       mov    ax, [esi+LD_BPB+BPB_RsvdSecCnt]
                        <1>       ;mov   ecx, 3
00006A8D B103           <1>       mov    cl, 3 ; 3 sectors
00006A8F E824870000     <1>       call   chs_read
00006A94 7240           <1>       jc     short read_fd_FAT_sectors_retn
                        <1> use_fd_FAT_sectors:
00006A96 8A4602         <1>       mov    al, [esi+LD_PhyDrvNo]
00006A99 0441           <1>       add    al, 'A'
00006A9B A2[DF5A0100]   <1>       mov    [FAT_BuffDrvName], al
00006AA0 C605[DE5A0100]01 <1>     mov    byte [FAT_BuffValidData], 1
00006AA7 E82B000000     <1>       call   fd_init_calculate_free_clusters
00006AAC 7228           <1>       jc     short read_fd_FAT_sectors_retn
                        <1>
                        <1> loc_use_fd_boot_sector_params_FAT:
00006AAE C6460301       <1>       mov    byte [esi+LD_FATType], 1 ; FAT 12
00006AB2 C6460401       <1>       mov    byte [esi+LD_FSType], 1
00006AB6 8B462D         <1>         mov     eax, [esi+LD_BPB+VolumeID]
                        <1> loc_cont_use_fd_boot_sector_params:
00006AB9 8A7E02         <1>       mov    bh, [esi+LD_PhyDrvNo]
00006ABC 887E7D         <1>       mov    [esi+LD_DParamEntry], bh
00006ABF 88FB           <1>       mov    bl, bh
00006AC1 80C341         <1>       add    bl, 'A'
00006AC4 881E           <1>       mov    byte [esi+LD_Name], bl
00006AC6 C6460101       <1>       mov    byte [esi+LD_DiskType], 1
00006ACA C6460500       <1>       mov    byte [esi+LD_LBAYes], 0
00006ACE C6467C00       <1>       mov    byte [esi+LD_PartitionEntry], 0
00006AD2 C6467E06       <1>       mov    byte [esi+LD_MediaChanged], 6 ; Volume Name Reset
                        <1>
                        <1> read_fd_FAT_sectors_retn:
00006AD6 C3             <1>       retn
                        <1>
                        <1> fd_init_calculate_free_clusters:
                        <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
                        <1>       ; 04/07/2009
                        <1>       ; INPUT ->
                        <1>       ;     ESI = Logical DOS drive description table address
                        <1>       ; OUTPUT ->
                        <1>       ;    [ESI+LD_FreeSectors] will be set
                        <1>
00006AD7 29C0           <1>       sub    eax, eax
00006AD9 894674         <1>       mov    [esi+LD_FreeSectors], eax ; 0
00006ADC B002           <1>       mov    al, 2 ; eax = 2
                        <1>
                        <1> fd_init_loop_get_next_cluster:
00006ADE E830000000     <1>       call   fd_init_get_next_cluster
00006AE3 722D           <1>       jc     short fd_init_calculate_free_clusters_retn
                        <1>
                        <1> fd_init_free_fat_clusters:
                        <1>       ;cmp   eax, 0
                        <1>       ;ja    short fd_init_pass_inc_free_cluster_count
                        <1>       ;and   eax, eax
                        <1>       ;jnz   short fd_init_pass_inc_free_cluster_count
00006AE5 6621C0         <1>       and    ax, ax
00006AE8 7504           <1>       jnz    short fd_init_pass_inc_free_cluster_count
                        <1>       ;inc   dword [esi+LD_FreeSectors]
00006AEA 66FF4674       <1>         inc  word [esi+LD_FreeSectors]
                        <1>
                        <1> fd_init_pass_inc_free_cluster_count:
                        <1>       ;mov   eax, [FAT_CurrentCluster]
00006AEE 66A1[DA5A0100] <1>       mov    ax, [FAT_CurrentCluster]
                        <1>       ;cmp   eax, [esi+LD_Clusters]
00006AF4 663B4678       <1>       cmp    ax, [esi+LD_Clusters]
00006AF8 7704           <1>       ja     short short retn_from_fd_init_calculate_free_clusters
                        <1>       ;inc   eax
00006AFA 6640           <1>       inc    ax
00006AFC EBE0           <1>       jmp    short fd_init_loop_get_next_cluster
                        <1>
                        <1> retn_from_fd_init_calculate_free_clusters:
00006AFE 8A4613         <1>       mov    al, [esi+LD_BPB+BPB_SecPerClust]
00006B01 3C01           <1>       cmp    al, 1
00006B03 760D           <1>       jna    short fd_init_calculate_free_clusters_retn
```

```
 990                                    <1>       ;movzx eax, al
 991 00006B05 6698                      <1>       cbw
 992                                    <1>       ;mov   ecx, [esi+LD_FreeSectors]
 993 00006B07 668B4E74                  <1>       mov    cx, [esi+LD_FreeSectors] ; Count of free clusters
 994                                    <1>       ;mul   ecx
 995 00006B0B 66F7E1                    <1>       mul    cx
 996                                    <1>       ;mov   [esi+LD_FreeSectors], eax
 997 00006B0E 66894674                  <1>       mov    [esi+LD_FreeSectors], ax
 998                                    <1> fd_init_calculate_free_clusters_retn:
 999 00006B12 C3                        <1>       retn
1000                                    <1>
1001                                    <1> fd_init_get_next_cluster:
1002                                    <1>       ; 04/02/2016
1003                                    <1>       ; 02/02/2016
1004                                    <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
1005                                    <1>       ; 04/07/2009
1006                                    <1>       ; INPUT ->
1007                                    <1>       ;    EAX = Current cluster
1008                                    <1>       ;    ESI = Logical DOS drive description table address
1009                                    <1>       ;    EDX = 0
1010                                    <1>       ; OUTPUT ->
1011                                    <1>       ;    EAX = Next cluster
1012                                    <1>
1013 00006B13 A3[DA5A0100]             <1>       mov    [FAT_CurrentCluster], eax
1014                                    <1> fd_init_get_next_cluster_readnext:
1015 00006B18 29D2                     <1>       sub    edx, edx ; 0
1016 00006B1A BB00040000               <1>       mov    ebx, 1024 ; 400h
1017 00006B1F F7F3                     <1>       div    ebx
1018                                    <1>       ; EAX = Count of 3 FAT sectors
1019                                    <1>       ; EDX = Buffer entry index
1020 00006B21 89C1                     <1>       mov    ecx, eax
1021                                    <1>       ;mov   eax, 3
1022 00006B23 B003                     <1>       mov    al, 3
1023 00006B25 F7E2                     <1>       mul    edx ; Multiply by 3
1024 00006B27 66D1E8                   <1>       shr    ax, 1 ; Divide by 2
1025 00006B2A 89C3                     <1>       mov    ebx, eax ; Buffer byte offset
1026 00006B2C 81C3001C0900             <1>       add    ebx, FAT_Buffer
1027 00006B32 89C8                     <1>       mov    eax, ecx
1028                                    <1>       ;mov   edx, 3
1029 00006B34 66BA0300                 <1>       mov    dx, 3
1030 00006B38 F7E2                     <1>       mul    edx
1031                                    <1>       ; EAX = FAT Beginning Sector
1032                                    <1>       ; EDX = 0
1033 00006B3A 8A0E                     <1>       mov    cl, [esi+LD_Name]
1034                                    <1>       ;cmp   byte [FAT_BuffValidData], 0
1035                                    <1>       ;jna   short fd_init_load_FAT_sectors0
1036 00006B3C 3A0D[DF5A0100]           <1>       cmp    cl, [FAT_BuffDrvName]
1037 00006B42 751E                     <1>       jne    short fd_init_load_FAT_sectors0
1038 00006B44 3B05[E25A0100]           <1>       cmp    eax, [FAT_BuffSector]
1039 00006B4A 751C                     <1>       jne    short fd_init_load_FAT_sectors1
1040                                    <1>       ;mov   eax, [FAT_CurrentCluster]
1041 00006B4C A0[DA5A0100]             <1>       mov    al, [FAT_CurrentCluster]
1042                                    <1>       ;shr   eax, 1
1043 00006B51 D0E8                     <1>       shr    al, 1
1044 00006B53 668B03                   <1>       mov    ax, [ebx]
1045 00006B56 7306                     <1>       jnc    short fd_init_gnc_even
1046 00006B58 66C1E804                 <1>       shr    ax, 4
1047                                    <1> fd_init_gnc_clc_retn:
1048 00006B5C F8                       <1>       clc
1049 00006B5D C3                       <1>       retn
1050                                    <1>
1051                                    <1> fd_init_gnc_even:
1052 00006B5E 80E40F                   <1>       and    ah, 0Fh
1053 00006B61 C3                       <1>       retn
1054                                    <1>
1055                                    <1> fd_init_load_FAT_sectors0:
1056 00006B62 880D[DF5A0100]           <1>       mov    [FAT_BuffDrvName], cl
1057                                    <1> fd_init_load_FAT_sectors1:
1058 00006B68 C605[DE5A0100]00         <1>       mov    byte [FAT_BuffValidData], 0
1059 00006B6F A3[E25A0100]             <1>       mov    [FAT_BuffSector], eax
1060 00006B74 034660                   <1>       add    eax, [esi+LD_FATBegin]
1061 00006B77 BB001C0900               <1>       mov    ebx, FAT_Buffer
1062                                    <1>       ;movzx ecx, word [esi+LD_BPB+BPB_FATSz16]
1063 00006B7C 668B4E1C                 <1>       mov    cx, [esi+LD_BPB+BPB_FATSz16]
1064 00006B80 662B0D[E25A0100]         <1>       sub    cx, [FAT_BuffSector]
1065                                    <1>         ;cmp ecx, 3
1066 00006B87 6683F903                 <1>       cmp    cx, 3
1067 00006B8B 7605                     <1>       jna    short fdinit_pass_fix_sector_count_3
1068                                    <1>       ;mov   ecx, 3
1069 00006B8D B903000000               <1>       mov    ecx, 3
1070                                    <1> fdinit_pass_fix_sector_count_3:
1071 00006B92 E821860000               <1>       call   chs_read
1072 00006B97 730D                     <1>       jnc    short fd_init_FAT_sectors_no_load_error
1073 00006B99 C605[DE5A0100]00         <1>       mov    byte [FAT_BuffValidData], 0
1074                                    <1>              ; Drv not ready or read Error !
1075 00006BA0 B80F000000               <1>       mov    eax, ERR_DRV_NOT_RDY ; 15
1076                                    <1>       ;xor   edx, edx
1077 00006BA5 C3                       <1>       retn
1078                                    <1>
1079                                    <1> fd_init_FAT_sectors_no_load_error:
1080 00006BA6 C605[DE5A0100]01         <1>       mov    byte [FAT_BuffValidData], 1
1081 00006BAD A1[DA5A0100]             <1>       mov    eax, [FAT_CurrentCluster]
1082 00006BB2 E961FFFFFF               <1>         jmp     fd_init_get_next_cluster_readnext
1083                                    <1>
1084                                    <1> get_FAT_volume_name:
1085                                    <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
1086                                    <1>       ; 12/09/2009
1087                                    <1>       ; INPUT ->
1088                                    <1>       ;      BH = Logical DOS drive number (0,1,2,3,4 ...)
1089                                    <1>       ;       BL = 0
1090                                    <1>       ; OUTPUT ->
1091                                    <1>       ;      CF = 0 -> ESI = Volume name address
```

```
1092                                <1>       ;     CF = 1 -> Root volume name not found
1093                                <1>
1094                                <1>       ;mov   ah, 0FFh
1095                                <1>       ;mov   al, [Last_Dos_DiskNo]
1096                                <1>       ;cmp   al, bh
1097                                <1>       ;jb     short loc_gfvn_dir_load_err
1098                                <1>
1099 00006BB7 89DE                 <1>       mov    esi, ebx
1100 00006BB9 81E600FF0000         <1>       and    esi, 0FF00h ; esi = bh
1101 00006BBF 81C600010900         <1>       add    esi, Logical_DOSDisks
1102 00006BC5 8A06                 <1>       mov     al, [esi+LD_Name]
1103 00006BC7 8A6603               <1>       mov     ah, [esi+LD_FATType]
1104 00006BCA 80FC01               <1>       cmp     ah, 1
1105 00006BCD 7210                 <1>       jb     short loc_gfvn_dir_load_err
1106 00006BCF 3C41                 <1>       cmp    al, 'A'
1107 00006BD1 720C                 <1>       jb      short loc_gfvn_dir_load_err
1108 00006BD3 80FC02               <1>       cmp    ah, 2
1109 00006BD6 7708                 <1>       ja      short get_FAT32_root_cluster
1110                                <1>
1111 00006BD8 E89B4E0000           <1>       call    load_FAT_root_directory
1112 00006BDD 730B                 <1>       jnc     short loc_get_volume_name
1113                                <1>
1114                                <1> loc_gfvn_dir_load_err:
1115 00006BDF C3                   <1>       retn
1116                                <1>
1117                                <1> get_FAT32_root_cluster:
1118 00006BE0 8B4632               <1>       mov    eax, [esi+LD_BPB+BPB_RootClus]
1119 00006BE3 E81B4F0000           <1>       call    load_FAT_sub_directory
1120 00006BE8 7224                 <1>       jc     short loc_get_volume_name_retn
1121                                <1>
1122                                <1> loc_get_volume_name:
1123 00006BEA BE00000800           <1>         mov     esi, Directory_Buffer
1124 00006BEF 6631C9               <1>       xor    cx, cx ; 0
1125                                <1> check_root_volume_name:
1126 00006BF2 8A06                 <1>       mov    al, [esi]
1127 00006BF4 08C0                 <1>       or      al, al
1128 00006BF6 7416                 <1>       jz      short loc_get_volume_name_retn
1129 00006BF8 807E0B08             <1>       cmp     byte [esi+0Bh], 08h
1130 00006BFC 7410                 <1>       je      short loc_get_volume_name_retn
1131 00006BFE 663B0D[F35A0100]     <1>       cmp     cx, [DirBuff_LastEntry]
1132 00006C05 7308                 <1>       jnb     short pass_check_root_volume_name
1133 00006C07 6641                 <1>       inc     cx
1134 00006C09 83C620               <1>       add     esi, 32
1135 00006C0C EBE4                 <1>       jmp     short check_root_volume_name
1136                                <1>
1137                                <1> loc_get_volume_name_retn:
1138 00006C0E C3                   <1>       retn
1139                                <1>
1140                                <1> pass_check_root_volume_name:
1141 00006C0F 803D[EF5A0100]03     <1>       cmp    byte [DirBuff_FATType], 3
1142 00006C16 7230                 <1>       jb     short loc_get_volume_name_retn_xor
1143                                <1>
1144 00006C18 BB001C0900           <1>       mov    ebx, FAT_Buffer
1145 00006C1D BE00010900           <1>       mov    esi, Logical_DOSDisks
1146 00006C22 31C0                 <1>       xor    eax, eax
1147 00006C24 8A25[EE5A0100]       <1>       mov    ah, [DirBuff_DRV]
1148 00006C2A 80EC41               <1>       sub    ah, 'A'
1149 00006C2D 01C6                 <1>       add    esi, eax
1150 00006C2F A1[F55A0100]         <1>       mov    eax, [DirBuff_Cluster]
1151 00006C34 E8E44C0000           <1>       call   get_next_cluster
1152 00006C39 7305                 <1>       jnc    short loc_gfvn_load_FAT32_dir_cluster
1153                                <1>
1154 00006C3B 83F801               <1>       cmp     eax, 1
1155 00006C3E F5                   <1>       cmc
1156 00006C3F C3                   <1>       retn
1157                                <1>
1158                                <1> loc_gfvn_load_FAT32_dir_cluster:
1159 00006C40 E8BE4E0000           <1>       call   load_FAT_sub_directory
1160 00006C45 73A3                 <1>       jnc    short loc_get_volume_name
1161 00006C47 C3                   <1>       retn
1162                                <1>
1163                                <1> loc_get_volume_name_retn_xor:
1164 00006C48 31C0                 <1>       xor    eax, eax
1165 00006C4A C3                   <1>       retn
1166                                <1>
1167                                <1> get_media_change_status:
1168                                <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
1169                                <1>       ; 09/09/2009
1170                                <1>       ; INPUT:
1171                                <1>       ;     DL = Drive number (physical)
1172                                <1>       ; OUTPUT: clc & AH = 6 media changed
1173                                <1>       ;     clc & AH = 0 media not changed
1174                                <1>       ;     stc -> Drive not ready or an error
1175                                <1>
1176 00006C4B B416                 <1>       mov    ah, 16h
1177 00006C4D E8B4D5FFFF           <1>       call   int13h
1178 00006C52 80FC06               <1>       cmp    ah, 06h
1179 00006C55 7405                 <1>       je     short loc_gmc_status_retn
1180 00006C57 08E4                 <1>       or     ah, ah
1181 00006C59 7401                 <1>       jz     short loc_gmc_status_retn
1182                                <1> loc_gmc_status_stc_retn:
1183 00006C5B F9                   <1>       stc
1184                                <1> loc_gmc_status_retn:
1185 00006C5C C3                   <1>       retn
2306                                    %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: 07/01/2017
   5                                <1> ; ----------------------------------------------------------------------------
   6                                <1> ; Beginning: 06/01/2016
   7                                <1> ; ----------------------------------------------------------------------------
```

```
                                <1> ; Assembler: NASM version 2.11 (trdos386.s)
                                <1> ; ----------------------------------------------------------------------------
                                <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
                                <1> ; MAINPROG.ASM (09/11/2011)
                                <1> ; ****************************************************************************
                                <1> ; MAINPROG.ASM [ TRDOS KERNEL - COMMAND EXECUTER SECTION - MAIN PROGRAM ]
                                <1> ; (c) 2004-2011  Erdogan TAN  [ 17/01/2004 ]  Last Update: 09/11/2011
                                <1> ; CMD_INTR.ASM [ TRDOS Command Interpreter Procedure ] Last Update: 09/11/2011
                                <1> ; DIR.ASM [ DIRECTORY FUNCTIONS ] Last Update: 09/10/2011
                                <1> ; FILE.ASM [ FILE FUNCTIONS ] Last Update: 09/10/2011
                                <1>
                                <1> change_current_drive:
                                <1>       ; 16/10/2016
                                <1>       ; 02/02/2016
                                <1>       ; 15/01/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>       ; 18/08/2011
                                <1>       ; 09/09/2009
                                <1>       ; INPUT:
                                <1>       ;   DL = Logical DOS Drive Number
                                <1>       ; OUTPUT:
                                <1>       ;  cf=1 -> Not successful
                                <1>       ;   EAX = Error code
                                <1>       ;  cf=0 ->
                                <1>       ;   EAX = 0 (successful)
                                <1>
00006C5D 31DB                   <1>       xor    ebx, ebx
00006C5F 88D7                   <1>       mov    bh, dl
                                <1>
                                <1>       ;cmp   dl, 1
                                <1>       ;jna   short loc_ccdrv_initial_media_change_check
                                <1>       ;cmp   bh, [Last_Dos_DiskNo]
                                <1>       ;ja    short loc_ccdrv_drive_not_ready_err
                                <1>
                                <1> loc_ccdrv_initial_media_change_check:
00006C61 BE00010900             <1>       mov    esi, Logical_DOSDisks
00006C66 01DE                   <1>       add    esi, ebx
                                <1> loc_ccdrv_dos_drive_name_check:
00006C68 80FA02                 <1>       cmp    dl, 2
00006C6B 720F                   <1>       jb     short loc_ccdrv_dos_drive_name_check_ok
                                <1>
00006C6D 8A06                   <1>       mov    al, [esi+LD_Name]
00006C6F 2C41                   <1>       sub    al, 'A'
00006C71 38D0                   <1>       cmp    al, dl
00006C73 7407                   <1>       je     short loc_ccdrv_dos_drive_name_check_ok
                                <1>
                                <1> loc_ccdrv_drive_not_ready_err:
                                <1>       ; 16/10/2016 (15h -> 15)
00006C75 B80F000000             <1>       mov    eax, 15 ; Drive not ready
                                <1> loc_change_current_drive_stc_retn:
00006C7A F9                     <1>       stc
00006C7B C3                     <1>       retn
                                <1>
                                <1> loc_ccdrv_dos_drive_name_check_ok:
00006C7C 8A667E                 <1>       mov    ah, [esi+LD_MediaChanged]
00006C7F 80FC06                 <1>       cmp    ah, 6  ; VOLUME NAME CHECK/MOVE SIGN
00006C82 7455                   <1>       je     short loc_ccdrv_get_FAT_volume_name_0
                                <1>
00006C84 80FA01                 <1>       cmp    dl, 1
00006C87 777D                   <1>       ja     short loc_gmcs_init_drv_hd
                                <1>
                                <1> loc_gmcs_init_drv_fd:
00006C89 08E4                   <1>       or     ah, ah
                                <1>       ; AH = 1 is initialization sign (invalid_fd_parameter)
00006C8B 7517                   <1>       jnz    short loc_ccdrv_call_fd_init
                                <1>
00006C8D E8B9FFFFFF             <1>       call   get_media_change_status
00006C92 72E1                   <1>       jc     short loc_ccdrv_drive_not_ready_err
                                <1>
00006C94 20E4                   <1>       and    ah, ah
00006C96 7476                   <1>       jz     short loc_change_current_drv3
                                <1>
00006C98 80F406                 <1>       xor    ah, 6
00006C9B 75D8                   <1>       jnz    short loc_ccdrv_drive_not_ready_err
                                <1>
                                <1> loc_ccdrv_call_fd_init_check_vol_id:
00006C9D E8490A0000             <1>       call   get_volume_serial_number
00006CA2 730D                   <1>       jnc    short loc_ccdrv_check_vol_serial
                                <1>
                                <1> loc_ccdrv_call_fd_init:
00006CA4 E872FCFFFF             <1>       call   floppy_drv_init
00006CA9 731A                   <1>       jnc    short loc_reset_drv_fd_current_dir
                                <1>
                                <1> loc_ccdrv_fdinit_fail_retn:
                                <1>       ; 16/10/2016
00006CAB B80F000000             <1>       mov    eax, 15 ; Drive not ready
00006CB0 C3                     <1>       retn
                                <1>
                                <1> loc_ccdrv_check_vol_serial:
00006CB1 A3[BC520100]           <1>       mov    [Current_VolSerial], eax
                                <1>       ;mov   dl, bh
00006CB6 E860FCFFFF             <1>       call   floppy_drv_init
00006CBB 72EE                   <1>       jc     short loc_ccdrv_fdinit_fail_retn
                                <1>
00006CBD 3B05[BC520100]         <1>       cmp    eax, [Current_VolSerial]
00006CC3 7445                   <1>       je     short loc_change_current_drv2
                                <1>
                                <1> loc_reset_drv_fd_current_dir:
00006CC5 31C0                   <1>       xor    eax, eax
00006CC7 88467F                 <1>         mov  [esi+LD_CDirLevel], al
00006CCA 89F7                   <1>       mov    edi, esi
00006CCC 81C780000000           <1>       add    edi, LD_CurrentDirectory
00006CD2 B920000000             <1>       mov    ecx, 32
```

```
00006CD7 F3AB                <1>       rep    stosd
                             <1>
                             <1> loc_ccdrv_get_FAT_volume_name_0:
00006CD9 8A4603              <1>       mov    al, [esi+LD_FATType]
00006CDC 08C0                <1>       or     al, al
00006CDE 742A                <1>       jz     short loc_change_current_drv2
                             <1>
00006CE0 56                  <1>       push   esi
00006CE1 3C02                <1>       cmp    al, 2
00006CE3 7705                <1>       ja     short loc_ccdrv_get_FAT32_vol_name
                             <1>
                             <1> loc_ccdrv_get_FAT2_16_vol_name:
00006CE5 83C631              <1>       add    esi, LD_BPB + VolumeLabel
00006CE8 EB03                <1>       jmp    short loc_ccdrv_get_FAT_volume_name_1
                             <1>
                             <1> loc_ccdrv_get_FAT32_vol_name:
00006CEA 83C64D              <1>       add    esi, LD_BPB + FAT32_VolLab
                             <1> loc_ccdrv_get_FAT_volume_name_1:
00006CED 53                  <1>       push   ebx
00006CEE 56                  <1>       push   esi
00006CEF E8C3FEFFFF          <1>       call   get_FAT_volume_name
00006CF4 5F                  <1>       pop    edi
00006CF5 5B                  <1>       pop    ebx
                             <1>       ; BL = 0
00006CF6 720B                <1>       jc     short loc_change_current_drv1
00006CF8 20C0                <1>       and    al, al
00006CFA 7407                <1>       jz     short loc_change_current_drv1
                             <1>
                             <1> loc_ccdrv_move_FAT_volume_name:
00006CFC B90B000000          <1>       mov    ecx, 11
00006D01 F3A4                <1>       rep    movsb
                             <1>
                             <1> loc_change_current_drv1:
00006D03 5E                  <1>       pop    esi
00006D04 EB04                <1>       jmp    short loc_change_current_drv2
                             <1>
                             <1> loc_gmcs_init_drv_hd:
00006D06 08E4                <1>       or     ah, ah
00006D08 7404                <1>       jz     short loc_change_current_drv3
                             <1>       ; BL = 0, BH = Logical DOS drive number
                             <1> loc_change_current_drv2:
00006D0A C6467E00            <1>       mov    byte [esi+LD_MediaChanged], 0
                             <1> loc_change_current_drv3:
00006D0E 883D[C6520100]      <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>
00006D14 8A4603              <1>       mov    al, [esi+LD_FATType]
00006D17 A2[C5520100]        <1>       mov    [Current_FATType], al
                             <1>
00006D1C 8A26                <1>       mov    ah, [esi+LD_Name]
00006D1E 8825[C7520100]      <1>       mov    [Current_Dir_Drv], ah
                             <1>
00006D24 20C0                <1>       and    al, al
00006D26 741D                <1>       jz     short loc_restore_FS_current_directory
                             <1>
                             <1> loc_restore_FAT_current_directory:
00006D28 8A667F              <1>       mov    ah, [esi+LD_CDirLevel]
00006D2B 8825[C4520100]      <1>       mov    [Current_Dir_Level], ah
00006D31 08E4                <1>       or     ah, ah
00006D33 7416                <1>         jz   short loc_ccdrv_reset_cdir_FAT_12_16_32_fcluster
                             <1>
00006D35 0FB6D4              <1>       movzx  edx, ah
00006D38 C0E204              <1>       shl    dl, 4 ; * 16
00006D3B 01F2                <1>         add  edx, esi
00006D3D 8B828C000000        <1>       mov    eax, [edx+LD_CurrentDirectory+12]
00006D43 EB2C                <1>       jmp    short loc_ccdrv_reset_cdir_FAT_fcluster
                             <1>
                             <1> loc_restore_FS_current_directory:
00006D45 E8F44D0000          <1>       call   load_current_FS_directory
00006D4A C3                  <1>       retn
                             <1>
                             <1> loc_ccdrv_reset_cdir_FAT_12_16_32_fcluster:
00006D4B 3C03                <1>       cmp    al, 3
00006D4D 7205                <1>       jb     short loc_ccdrv_reset_cdir_FAT_12_16_fcluster
                             <1> loc_ccdrv_reset_cdir_FAT32_fcluster:
00006D4F 8B4632              <1>       mov    eax, [esi+LD_BPB+FAT32_RootFClust]
00006D52 EB04                <1>       jmp    short loc_ccdrv_check_rootdir_sign
                             <1> loc_ccdrv_reset_cdir_FAT_12_16_fcluster:
00006D54 30C0                <1>       xor    al, al  ; xor eax, eax
00006D56 31D2                <1>       xor    edx, edx
                             <1> loc_ccdrv_check_rootdir_sign:
00006D58 80BE8000000000      <1>       cmp    byte [esi+LD_CurrentDirectory], 0
00006D5F 7510                <1>       jne    short loc_ccdrv_reset_cdir_FAT_fcluster
                             <1> loc_ccdrv_set_rootdir_FAT_fcluster:
00006D61 89868C000000        <1>         mov     [esi+LD_CurrentDirectory+12], eax
00006D67 C786800000000524F4F- <1>       mov    dword [esi+LD_CurrentDirectory], 'ROOT'
00006D70 54                  <1>
```

```
                                <1>
                                <1> loc_ccdrv_reset_cdir_FAT_fcluster:
00006D71 A3[C0520100]           <1>       mov   [Current_Dir_FCluster], eax
                                <1>
00006D76 BF[275B0100]           <1>       mov   edi, PATH_Array
00006D7B 89F2                   <1>       mov   edx, esi
00006D7D 81C680000000           <1>       add   esi, LD_CurrentDirectory
00006D83 B920000000             <1>       mov   ecx, 32
00006D88 F3A5                   <1>       rep   movsd
                                <1>
00006D8A E8832D0000             <1>       call  change_prompt_dir_string
                                <1>
00006D8F 89D6                   <1>       mov   esi, edx
                                <1>
00006D91 29C0                   <1>         sub eax, eax
                                <1>        ;sub edx, edx
00006D93 BF[C7520100]           <1>       mov   edi, Current_Dir_Drv
                                <1>
00006D98 A2[AD060100]           <1>       mov   [Restore_CDIR], al ; 0
00006D9D 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
00006D9E C705[845F0100]-        <1>       mov   dword [mainprog_return_addr], return_from_cmd_interpreter
00006DA4 [526E0000]             <1>
                                <1>
                                <1> loc_TRDOS_prompt:
00006DA8 BF[C6530100]           <1>       mov   edi, TextBuffer
00006DAD C6075B                 <1>       mov   byte [edi], "["
00006DB0 47                     <1>       inc   edi
00006DB1 BE[00070100]           <1>       mov   esi, TRDOSPromptLabel
                                <1> get_next_prompt_label_char:
00006DB6 803E20                 <1>       cmp   byte [esi], 20h
00006DB9 7203                   <1>       jb    short pass_prompt_label
00006DBB A4                     <1>       movsb
00006DBC EBF8                   <1>       jmp   short get_next_prompt_label_char
                                <1> pass_prompt_label:
00006DBE C6075D                 <1>       mov   byte [edi], "]"
00006DC1 47                     <1>       inc   edi
00006DC2 C60720                 <1>       mov   byte [edi], 20h
00006DC5 47                     <1>       inc   edi
00006DC6 BE[C7520100]           <1>       mov   esi, Current_Dir_Drv
00006DCB 66A5                   <1>       movsw
00006DCD A4                     <1>       movsb
                                <1> loc_prompt_current_directory:
00006DCE 803E20                 <1>       cmp   byte [esi], 20h
00006DD1 7203                   <1>       jb    short pass_prompt_current_directory
00006DD3 A4                     <1>       movsb
00006DD4 EBF8                   <1>       jmp   short loc_prompt_current_directory
                                <1> pass_prompt_current_directory:
00006DD6 C6073E                 <1>       mov   byte [edi], '>'
00006DD9 47                     <1>       inc   edi
00006DDA C60700                 <1>       mov   byte [edi], 0
00006DDD BE[C6530100]           <1>       mov   esi, TextBuffer
00006DE2 E876F5FFFF             <1>       call  print_msg
                                <1>
                                <1>       ;sub   bh, bh ; video page = 0
                                <1>       ;call  get_cpos ; get cursor position
00006DE7 668B15[1E520100]       <1>       mov   dx, [CURSOR_POSN] ; video page 0
00006DEE 8815[26530100]         <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:
00006DF4 E899000000             <1>       call  rw_char
                                <1> loc_move_command:
00006DF9 BE[76530100]           <1>       mov   esi, CommandBuffer
00006DFE 89F7                   <1>       mov   edi, esi
00006E00 31C9                   <1>       xor   ecx, ecx
                                <1> first_command_char:
00006E02 AC                     <1>       lodsb
00006E03 3C20                   <1>       cmp   al, 20h
00006E05 772E                   <1>       ja    short pass_space_control
00006E07 7241                   <1>       jb    short loc_move_cmd_arguments_ok
00006E09 81FE[C5530100]         <1>       cmp   esi, CommandBuffer + 79
00006E0F 72F1                   <1>       jb    short first_command_char
00006E11 EB37                   <1>       jmp   short loc_move_cmd_arguments_ok
                                <1>
                                <1> next_command_char:
00006E13 AC                     <1>       lodsb
```

```
312 00006E14 3C20                <1>       cmp    al, 20h
313 00006E16 771D                <1>       ja     short pass_space_control
314 00006E18 7230                <1>       jb     short loc_move_cmd_arguments_ok
315                              <1>
316                              <1> loc_1st_cmd_arg: ; 30/01/2016
317 00006E1A AC                  <1>       lodsb
318 00006E1B 3C20                <1>       cmp    al, 20h
319 00006E1D 74FB                <1>       je     short loc_1st_cmd_arg
320 00006E1F 7229                <1>       jb     short loc_move_cmd_arguments_ok
321                              <1>
322 00006E21 C60700              <1>         mov     byte [edi], 0
323 00006E24 47                  <1>       inc    edi
324                              <1>
325                              <1> loc_move_cmd_arguments:
326 00006E25 AA                  <1>       stosb
327 00006E26 81FE[C5530100]      <1>       cmp    esi, CommandBuffer + 79
328 00006E2C 731C                <1>       jnb    short loc_move_cmd_arguments_ok
329 00006E2E AC                  <1>         lodsb
330 00006E2F 3C20                <1>       cmp    al, 20h
331 00006E31 73F2                <1>       jnb    short loc_move_cmd_arguments
332 00006E33 EB15                <1>       jmp    short loc_move_cmd_arguments_ok
333                              <1>
334                              <1> pass_space_control:
335 00006E35 3C61                <1>       cmp    al, 61h
336 00006E37 7206                <1>       jb     short pass_capitalize
337 00006E39 3C7A                <1>       cmp    al, 7Ah
338 00006E3B 7702                <1>       ja     short pass_capitalize
339 00006E3D 24DF                <1>       and    al, 0DFh
340                              <1> pass_capitalize:
341 00006E3F AA                  <1>       stosb
342 00006E40 FEC1                <1>       inc     cl
343 00006E42 81FE[C5530100]      <1>         cmp     esi, CommandBuffer + 79
344 00006E48 72C9                <1>       jb      short next_command_char
345                              <1>
346                              <1> loc_move_cmd_arguments_ok:
347 00006E4A C60700              <1>         mov     byte [edi], 0
348                              <1>
349                              <1> call_command_interpreter:
350 00006E4D E8D4080000          <1>       call    command_interpreter
351                              <1>
352                              <1> return_from_cmd_interpreter:
353 00006E52 B950000000          <1>         mov  ecx, 80
354                              <1>       ;mov   cx, 80
355 00006E57 BF[76530100]        <1>       mov    edi, CommandBuffer
356 00006E5C 30C0                <1>       xor    al, al
357 00006E5E F3AA                <1>       rep    stosb
358                              <1>       ;cmp   byte [Program_Exit], 0
359                              <1>       ;ja    short loc_terminate_trdos
360                              <1>
361                              <1>       ; 16/01/2016
362 00006E60 803D[C25E0000]03    <1>       cmp    byte [CRT_MODE], 3 ; 80*25 color
363 00006E67 741D                <1>       je     short pass_set_txt_mode
364                              <1>
365 00006E69 E8F5A6FFFF          <1>       call   set_txt_mode ; set vide mode to 03h
366                              <1>       ; 07/01/2017
367 00006E6E 30C0                <1>       xor    al, al
368                              <1>
369                              <1> loc_check_active_page:
370                              <1>       ;xor   al, al
371 00006E70 3805[2E520100]      <1>       cmp    [ACTIVE_PAGE], al ; 0
372 00006E76 0F842CFFFFFF        <1>         je      loc_TRDOS_prompt
373                              <1>       ; AL = 0 = video page 0
374 00006E7C E8FBAAFFFF          <1>       call   set_active_page
375 00006E81 E922FFFFFF          <1>         jmp     loc_TRDOS_prompt ; infinitive loop
376                              <1>
377                              <1> pass_set_txt_mode:
378 00006E86 BE[4B130100]        <1>       mov    esi, nextline
379 00006E8B E8CDF4FFFF          <1>       call   print_msg
380 00006E90 EBDE                <1>       jmp     short loc_check_active_page
381                              <1>
382                              <1> rw_char:
383                              <1>       ; 13/05/2016
384                              <1>       ; 30/01/2016
385                              <1>       ; 29/01/2016
386                              <1>       ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
387                              <1>       ; 2004-2005
388                              <1>
389                              <1>       ; DH = cursor row, DL = cursor column
390                              <1>       ; BH = 0 = video page number (active page)
391                              <1>
392                              <1>       ;xor   bh, bh ; 0 = video page 0
393                              <1>
394                              <1> readnextchar:
395 00006E92 30E4                <1>       xor     ah, ah
396 00006E94 E87D9DFFFF          <1>       call   int16h
397 00006E99 20C0                <1>       and    al, al
398 00006E9B 7434                <1>       jz     short loc_arrow
399 00006E9D 3CE0                <1>       cmp    al, 0E0h
400 00006E9F 7430                <1>       je     short loc_arrow
401 00006EA1 3C08                <1>       cmp    al, 08h
402 00006EA3 7544                <1>       jne    short char_return
403                              <1> loc_back:
404 00006EA5 3A15[26530100]      <1>       cmp    dl, [CursorColumn]
405 00006EAB 76E5                <1>       jna     short readnextchar
406                              <1> prev_column:
407 00006EAD FECA                <1>       dec    dl
408                              <1> set_cursor_pos:
409 00006EAF 6652                <1>       push   dx
410                              <1>       ;xor   bh, bh ; 0 = video page 0
411                              <1>       ; DH = Row, DL = Column
412 00006EB1 E892AEFFFF          <1>       call   _set_cpos ; 17/01/2016
413 00006EB6 665A                <1>         pop  dx
```

```
                                <1>       ;movzx ebx, dl
00006EB8 88D3                   <1>       mov    bl, dl
00006EBA 2A1D[26530100]         <1>       sub    bl, [CursorColumn]
00006EC0 B020                   <1>       mov    al, 20h
00006EC2 8883[76530100]         <1>       mov    [CommandBuffer+ebx], al
                                <1>       ;sub   bh, bh ; video page 0
                                <1>       ;mov   cx, 1
00006EC8 B307                   <1>       mov    bl, 7 ; color attribute
00006ECA E86AADFFFF             <1>       call   _write_c_current ; 17/01/2016
                                <1>       ;mov   dx, [CURSOR_POSN]
00006ECF EBC1                   <1>       jmp    short readnextchar
                                <1> loc_arrow:
00006ED1 80FC4B                 <1>       cmp    ah, 4Bh
00006ED4 74CF                   <1>       je     short loc_back
00006ED6 80FC53                 <1>       cmp    ah, 53h
00006ED9 74CA                   <1>       je       short loc_back
00006EDB 80FC4D                 <1>       cmp    ah, 4Dh
00006EDE 75B2                   <1>       jne    short readnextchar
00006EE0 80FA4F                 <1>       cmp    dl, 79
00006EE3 73AD                   <1>       jnb    short readnextchar
00006EE5 FEC2                   <1>       inc    dl
00006EE7 EBC6                   <1>       jmp    short set_cursor_pos
                                <1> char_return:
00006EE9 0FB6DA                 <1>       movzx  ebx, dl
00006EEC 2A1D[26530100]         <1>       sub    bl, [CursorColumn]
00006EF2 3C20                   <1>       cmp    al, 20h
00006EF4 7220                   <1>       jb     short loc_escape
00006EF6 8883[76530100]         <1>       mov    [CommandBuffer+ebx], al
00006EFC 80FA4F                 <1>       cmp    dl, 79
00006EFF 7391                   <1>       jnb    short readnextchar
00006F01 66BB0700               <1>       mov    bx, 7 ; color attribute
00006F05 E8A8ADFFFF             <1>       call   _write_tty
```

```
00006F0A 668B15[1E520100]       <1>       mov    dx, [CURSOR_POSN] ; video page 0
00006F11 E97CFFFFFF             <1>         jmp     readnextchar
                                <1> loc_escape:
00006F16 3C1B                   <1>       cmp    al, 1Bh
00006F18 7418                   <1>       je     short rw_char_retn
                                <1>       ;
00006F1A 3C0D                   <1>       cmp    al, 0Dh ; CR
00006F1C 0F8570FFFFFF           <1>         jne     readnextchar
                                <1>       ; 13/05/2016
00006F22 66BB0700               <1>       mov    bx, 7 ; attribute/color (bl)
                                <1>                    ; video page 0 (bh=0)
00006F26 E887ADFFFF             <1>       call   _write_tty
                                <1>       ;mov   bx, 7  ; attribute/color
                                <1>                    ; video page 0 (bh=0)
00006F2B B00A                   <1>       mov    al, 0Ah ; LF
00006F2D E880ADFFFF             <1>       call   _write_tty
                                <1> rw_char_retn:
00006F32 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
00006F33 E826EBFFFF             <1>       call   RTC_40 ; GET RTC DATE
                                <1>
00006F38 88D0                   <1>       mov    al, dl
00006F3A E8CE9CFFFF             <1>       call   bcd_to_ascii
00006F3F 66A3[EC070100]         <1>       mov    [Day], ax
                                <1>
00006F45 88F0                   <1>       mov    al, dh
00006F47 E8C19CFFFF             <1>       call   bcd_to_ascii
00006F4C 66A3[EF070100]         <1>       mov    [Month], ax
                                <1>
00006F52 88E8                   <1>       mov    al, ch
00006F54 E8B49CFFFF             <1>       call   bcd_to_ascii
00006F59 66A3[F2070100]         <1>       mov    [Century], ax
                                <1>
```

```
00006F5F 88C8                   <1>       mov    al, cl
00006F61 E8A79CFFFF             <1>       call   bcd_to_ascii
00006F66 66A3[F4070100]         <1>       mov    word [Year], ax
                                <1>
00006F6C BE[DC070100]           <1>       mov    esi, Msg_Show_Date
00006F71 E8E7F3FFFF             <1>       call   print_msg
                                <1>
00006F76 C3                     <1>       retn
                                <1>
                                <1> set_date:
                                <1>       ; 13/05/2016
                                <1>       ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>         ; 2004-2005
                                <1>
00006F77 BE[C0070100]           <1>       mov    esi, Msg_Enter_Date
00006F7C E8DCF3FFFF             <1>       call   print_msg
                                <1>
                                <1> loc_enter_day_1:
00006F81 30E4                   <1>       xor     ah, ah
00006F83 E88E9CFFFF             <1>       call   int16h
                                <1>       ; AL = ASCII Code of the Character
00006F88 3C0D                   <1>       cmp    al, 13
00006F8A 0F84B7010000           <1>       je     loc_set_date_retn
00006F90 3C1B                   <1>       cmp    al, 27
00006F92 0F84AF010000           <1>       je     loc_set_date_retn
00006F98 A2[EC070100]           <1>       mov    [Day], al
00006F9D 3C30                   <1>       cmp    al, '0'
00006F9F 0F82AD010000           <1>       jb     loc_set_date_stc_0
00006FA5 3C33                   <1>       cmp    al, '3'
00006FA7 0F87A5010000           <1>       ja     loc_set_date_stc_0
                                <1>       ; 13/05/2016
```

```
                            <1>       ;mov   bx, 7 ; attribute/color (bl)
                            <1>                    ; video page 0 (bh)
00006FAD B307               <1>       mov    bl, 7
00006FAF E8FEACFFFF         <1>       call   _write_tty
                            <1> loc_enter_day_2:
00006FB4 30E4               <1>       xor     ah, ah
00006FB6 E85B9CFFFF         <1>       call   int16h
                            <1>       ; AL = ASCII Code of the Character
00006FBB 3C1B               <1>       cmp    al, 27
00006FBD 0F8484010000       <1>         je      loc_set_date_retn
00006FC3 A2[ED070100]       <1>       mov    [Day+1], al
00006FC8 3C30               <1>       cmp    al, '0'
00006FCA 0F828C010000       <1>         jb      loc_set_date_stc_1
00006FD0 3C39               <1>       cmp    al, '9'
00006FD2 0F8784010000       <1>         ja      loc_set_date_stc_1
00006FD8 803D[EC070100]33   <1>       cmp    byte [Day], '3'
00006FDF 7208               <1>       jb     short pass_set_day_31
00006FE1 3C31               <1>       cmp    al, '1'
00006FE3 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)
00006FE9 B307               <1>       mov    bl, 7
00006FEB E8C2ACFFFF         <1>       call   _write_tty
                            <1> loc_enter_separator_1:
00006FF0 28E4               <1>       sub     ah, ah ; 0
00006FF2 E81F9CFFFF         <1>       call   int16h
                            <1>       ; AL = ASCII Code of the Character
00006FF7 3C1B               <1>       cmp    al, 27
00006FF9 0F8448010000       <1>         je      loc_set_date_retn
00006FFF 3C2D               <1>       cmp    al, '-'
00007001 7408               <1>       je     short pass_set_date_separator_1
00007003 3C2F               <1>       cmp    al, '/'
00007005 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)
0000700B B307               <1>       mov    bl, 7
0000700D E8A0ACFFFF         <1>       call   _write_tty
                            <1> loc_enter_month_1:
00007012 30E4               <1>       xor     ah, ah ; 0
00007014 E8FD9BFFFF         <1>       call   int16h
                            <1>       ; AL = ASCII Code of the Character
00007019 3C1B               <1>       cmp    al, 27
0000701B 0F8426010000       <1>         je      loc_set_date_retn
00007021 A2[EF070100]       <1>       mov    [Month], al
00007026 3C30               <1>       cmp    al, '0'
00007028 0F8264010000       <1>         jb      loc_set_date_stc_3
0000702E 3C31               <1>       cmp    al, '1'
00007030 0F875C010000       <1>         ja      loc_set_date_stc_3
                            <1>       ; 13/05/2016
                            <1>       ;mov   bx, 7 ; attribute/color (bl)
                            <1>                    ; video page 0 (bh)
00007036 B307               <1>       mov    bl, 7
00007038 E875ACFFFF         <1>       call   _write_tty
                            <1> loc_enter_month_2:
0000703D 30E4               <1>       xor     ah, ah
0000703F E8D29BFFFF         <1>       call   int16h
                            <1>       ; AL = ASCII Code of the Character
00007044 3C1B               <1>       cmp    al, 27
00007046 0F84FB000000       <1>         je      loc_set_date_retn
0000704C A2[F0070100]       <1>       mov    [Month+1], al
00007051 3C30               <1>       cmp    al, '0'
00007053 0F8254010000       <1>         jb      loc_set_date_stc_4
00007059 3C39               <1>       cmp    al, '9'
0000705B 0F874C010000       <1>         ja      loc_set_date_stc_4
00007061 803D[EF070100]31   <1>       cmp    byte [Month], '1'
00007068 7208               <1>       jb     short pass_set_month_12
0000706A 3C32               <1>       cmp    al, '2'
0000706C 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)
00007072 B307               <1>       mov    bl, 7
00007074 E839ACFFFF         <1>       call   _write_tty
                            <1> loc_enter_separator_2:
00007079 28E4               <1>       sub     ah, ah
0000707B E8969BFFFF         <1>       call   int16h
                            <1>       ; AL = ASCII Code of the Character
00007080 3C1B               <1>       cmp    al, 27
00007082 0F84BF000000       <1>         je      loc_set_date_retn
00007088 3C2D               <1>       cmp    al, '-'
0000708A 7408               <1>       je     short pass_set_date_separator_2
0000708C 3C2F               <1>       cmp    al, '/'
0000708E 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)
00007094 B307               <1>       mov    bl, 7
00007096 E817ACFFFF         <1>       call   _write_tty
                            <1> loc_enter_year_1:
0000709B 30E4               <1>       xor    ah, ah
0000709D E8749BFFFF         <1>       call   int16h
                            <1>       ; AL = ASCII Code of the Character
000070A2 3C1B               <1>       cmp    al, 27
000070A4 0F849D000000       <1>         je      loc_set_date_retn
000070AA A2[F4070100]       <1>       mov    [Year], al
000070AF 3C30               <1>       cmp    al, '0'
```

```
618 000070B1 0F822C010000        <1>         jb      loc_set_date_stc_6
619 000070B7 3C39                <1>       cmp   al, '9'
620 000070B9 0F8724010000        <1>         ja      loc_set_date_stc_6
621                              <1>       ; 13/05/2016
622                              <1>       ;mov   bx, 7 ; attribute/color (bl)
623                              <1>                    ; video page 0 (bh)
624 000070BF B307                <1>       mov   bl, 7
625 000070C1 E8ECABFFFF          <1>       call  _write_tty
626                              <1> loc_enter_year_2:
627 000070C6 30E4                <1>       xor   ah, ah
628 000070C8 E8499BFFFF          <1>       call  int16h
629                              <1>       ; AL = ASCII Code of the Character
630 000070CD 3C1B                <1>       cmp   al, 27
631 000070CF 7476                <1>       je    short loc_set_date_retn
632 000070D1 A2[F5070100]        <1>       mov   byte [Year+1], al
633 000070D6 3C30                <1>       cmp   al, '0'
634 000070D8 0F8220010000        <1>         jb      loc_set_date_stc_7
635 000070DE 3C39                <1>       cmp   al, '9'
636 000070E0 0F8718010000        <1>         ja      loc_set_date_stc_7
637                              <1>       ; 13/05/2016
638                              <1>       ;mov   bx, 7 ; attribute/color (bl)
639                              <1>                    ; video page 0 (bh)
640 000070E6 B307                <1>       mov   bl, 7
641 000070E8 E8C5ABFFFF          <1>       call  _write_tty
642                              <1> loc_set_date_get_lchar_again:
643 000070ED 28E4                <1>       sub   ah, ah ; 0
644 000070EF E8229BFFFF          <1>       call  int16h
645                              <1>       ; AL = ASCII Code of the Character
646 000070F4 3C0D                <1>       cmp   al, 13 ; ENTER key
647 000070F6 7412                <1>       je    short loc_set_date_progress
648 000070F8 3C1B                <1>       cmp   al, 27 ; ESC key
649 000070FA 744B                <1>       je    short loc_set_date_retn
650                              <1>       ;
651 000070FC E82A010000          <1>       call  check_for_backspace
652 00007101 75EA                <1>       jne   short loc_set_date_get_lchar_again
653                              <1>
654                              <1> loc_set_date_bs_8:
655 00007103 E811010000          <1>       call  write_backspace
656 00007108 EBBC                <1>       jmp   short loc_enter_year_2
657                              <1>
658                              <1> loc_set_date_progress:
659                              <1>       ; Get Current Date
660                              <1>       ;mov   ah, 04h
661                              <1>       ;call  int1Ah
662 0000710A E84FE9FFFF          <1>       call  RTC_40 ; GET RTC DATE
663                              <1>       ; CH = century (in BCD)
664                              <1>
665 0000710F 66A1[F4070100]      <1>       mov   ax, [Year]
666 00007115 662D3030            <1>       sub   ax, '00'
667 00007119 C0E004              <1>       shl   al, 4 ; * 16
668 0000711C 88C1                <1>       mov   cl, al
669 0000711E 00E1                <1>       add   cl, ah
670 00007120 66A1[EF070100]      <1>       mov   ax, [Month]
671 00007126 662D3030            <1>       sub   ax, '00'
672 0000712A C0E004              <1>       shl   al, 4 ; * 16
673 0000712D 88C6                <1>       mov   dh, al
674 0000712F 00E6                <1>       add   dh, ah
675 00007131 66A1[EC070100]      <1>       mov   ax, [Day]
676 00007137 662D3030            <1>       sub   ax, '00'
677 0000713B C0E004              <1>       shl   al, 4 ; * 16
678 0000713E 88C2                <1>       mov   dl, al
679 00007140 00E2                <1>       add   dl, ah
680                              <1>
681                              <1>       ;mov   ah, 05h
682                              <1>       ;call  int1Ah
683 00007142 E844E9FFFF          <1>       call  RTC_50 ; SET RTC DATE
684                              <1>
685                              <1> loc_set_date_retn:
686 00007147 BE[4B130100]        <1>       mov   esi, nextline
687 0000714C E80CF2FFFF          <1>       call  print_msg
688 00007151 C3                  <1>       retn
689                              <1>
690                              <1> loc_set_date_stc_0:
691                              <1>       ;xor   bh, bh ; video page 0
692 00007152 E83BACFFFF          <1>       call  beeper ; BEEP !
693 00007157 E925FEFFFF          <1>         jmp     loc_enter_day_1
694                              <1> loc_set_date_stc_1:
695 0000715C E8CA000000          <1>       call  check_for_backspace
696 00007161 740A                <1>       je    short loc_set_date_bs_1
697                              <1>       ;xor   bh, bh ; video page 0
698 00007163 E82AACFFFF          <1>       call  beeper ; BEEP !
699 00007168 E947FEFFFF          <1>         jmp     loc_enter_day_2
700                              <1> loc_set_date_bs_1:
701 0000716D E8A7000000          <1>       call  write_backspace
702 00007172 E90AFEFFFF          <1>         jmp     loc_enter_day_1
703                              <1> loc_set_date_stc_2:
704 00007177 E8AF000000          <1>       call  check_for_backspace
705 0000717C 740A                <1>       je    short loc_set_date_bs_2
706                              <1>       ;xor   bh, bh ; video page 0
707 0000717E E80FACFFFF          <1>       call  beeper ; BEEP !
708 00007183 E968FEFFFF          <1>         jmp     loc_enter_separator_1
709                              <1> loc_set_date_bs_2:
710 00007188 E88C000000          <1>       call  write_backspace
711 0000718D E922FEFFFF          <1>         jmp     loc_enter_day_2
712                              <1> loc_set_date_stc_3:
713 00007192 E894000000          <1>       call  check_for_backspace
714 00007197 740A                <1>       je short loc_set_date_bs_3
715                              <1>       ;xor   bh, bh ; video page 0
716 00007199 E8F4ABFFFF          <1>       call  beeper ; BEEP !
717 0000719E E96FFEFFFF          <1>         jmp     loc_enter_month_1
718                              <1> loc_set_date_bs_3:
719 000071A3 E871000000          <1>       call  write_backspace
```

```
000071A8 E943FEFFFF          <1>         jmp     loc_enter_separator_1
                             <1> loc_set_date_stc_4:
000071AD E879000000          <1>       call   check_for_backspace
000071B2 740A                <1>       je     short loc_set_date_bs_4
                             <1>       ;xor   bh, bh ; video page 0
000071B4 E8D9ABFFFF          <1>       call   beeper ; BEEP !
000071B9 E97FFEFFFF          <1>         jmp     loc_enter_month_2
                             <1> loc_set_date_bs_4:
000071BE E856000000          <1>       call   write_backspace
000071C3 E94AFEFFFF          <1>         jmp     loc_enter_month_1
                             <1> loc_set_date_stc_5:
000071C8 E85E000000          <1>       call   check_for_backspace
000071CD 740A                <1>       je     short loc_set_date_bs_5
                             <1>       ;xor   bh, bh ; video page 0
000071CF E8BEABFFFF          <1>       call   beeper ; BEEP !
000071D4 E9A0FEFFFF          <1>         jmp     loc_enter_separator_2
                             <1> loc_set_date_bs_5:
000071D9 E83B000000          <1>       call   write_backspace
000071DE E95AFEFFFF          <1>         jmp     loc_enter_month_2
                             <1> loc_set_date_stc_6:
000071E3 E843000000          <1>       call   check_for_backspace
000071E8 740A                <1>         je      short  loc_set_date_bs_6
                             <1>       ;xor   bh, bh ; video page 0
000071EA E8A3ABFFFF          <1>       call   beeper ; BEEP !
000071EF E9A7FEFFFF          <1>         jmp     loc_enter_year_1
                             <1> loc_set_date_bs_6:
000071F4 E820000000          <1>       call   write_backspace
000071F9 E97BFEFFFF          <1>         jmp     loc_enter_separator_2
                             <1> loc_set_date_stc_7:
000071FE E828000000          <1>       call   check_for_backspace
00007203 740A                <1>       je     short loc_set_date_bs_7
                             <1>       ;xor   bh, bh ; video page 0
00007205 E888ABFFFF          <1>       call   beeper ; BEEP !
0000720A E9B7FEFFFF          <1>         jmp     loc_enter_year_2
                             <1> loc_set_date_bs_7:
0000720F E805000000          <1>       call   write_backspace
00007214 E982FEFFFF          <1>         jmp     loc_enter_year_1
                             <1>
                             <1> write_backspace:
                             <1>       ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
00007219 B008                <1>       mov    al, 08h ; BACKSPACE
                             <1>       ; 13/05/2016
0000721B 66BB0700            <1>       mov    bx, 7 ; bl = attribute/color
                             <1>                    ; bh = video page = 0
0000721F E88EAAFFFF          <1>       call   _write_tty
00007224 B020                <1>       mov    al, 20h ; BLANK/SPACE char
                             <1>       ;mov   bx, 7 ; attribute/color
                             <1>       ;call  _write_c_current
                             <1>       ;retn
00007226 E90EAAFFFF          <1>       jmp    _write_c_current
                             <1>
                             <1> check_for_backspace:
                             <1>       ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
0000722B 663D080E            <1>       cmp    ax, 0E08h
0000722F 7410                <1>       je     short cfbs_retn
00007231 663DE04B            <1>       cmp    ax, 4BE0h
00007235 740A                <1>       je     short cfbs_retn
00007237 663D004B            <1>       cmp    ax, 4B00h
0000723B 7404                <1>       je     short cfbs_retn
0000723D 663DE053            <1>       cmp    ax, 53E0h
                             <1> cfbs_retn:
00007241 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
00007242 E8A6E7FFFF          <1>       call   RTC_20 ; GET RTC TIME
                             <1>
00007247 88E8                <1>       mov    al, ch
00007249 E8BF99FFFF          <1>       call   bcd_to_ascii
0000724E 66A3[1A080100]      <1>       mov    [Hour], ax
                             <1>
00007254 88C8                <1>       mov    al, cl
00007256 E8B299FFFF          <1>       call   bcd_to_ascii
0000725B 66A3[1D080100]      <1>       mov    [Minute], ax
                             <1>
00007261 88F0                <1>       mov    al, dh
00007263 E8A599FFFF          <1>       call   bcd_to_ascii
00007268 66A3[20080100]      <1>       mov    [Second], ax
                             <1>
0000726E BE[0A080100]        <1>       mov    esi, Msg_Show_Time
00007273 E8E5F0FFFF          <1>       call   print_msg
00007278 C3                  <1>       retn
                             <1>
                             <1> set_time:
                             <1>       ; 13/05/2016
                             <1>       ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>         ; 2004-2005
                             <1>
00007279 BE[F9070100]        <1>       mov    esi, Msg_Enter_Time
0000727E E8DAF0FFFF          <1>       call   print_msg
                             <1>
                             <1> loc_enter_hour_1:
00007283 30E4                <1>       xor     ah, ah
00007285 E88C99FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
0000728A 3C0D                <1>       cmp    al, 13 ; ENTER key
0000728C 0F84AE010000        <1>         je      loc_set_time_retn
00007292 3C1B                <1>       cmp    al, 27 ; ESC key
```

```
822 00007294 0F84A6010000        <1>         je      loc_set_time_retn
823 0000729A A2[1A080100]        <1>       mov    [Hour], al
824 0000729F 3C30                <1>       cmp    al, '0'
825 000072A1 0F82A4010000        <1>         jb      loc_set_time_stc_0
826 000072A7 3C32                <1>       cmp    al, '2'
827 000072A9 0F879C010000        <1>         ja      loc_set_time_stc_0
828                              <1>       ; 13/05/2016
829                              <1>       ;mov   bx, 7 ; attribute/color (bl)
830                              <1>                    ; video page 0 (bh)
831 000072AF B307                <1>       mov    bl, 7
832 000072B1 E8FCA9FFFF          <1>       call   _write_tty
833                              <1> loc_enter_hour_2:
834 000072B6 30E4                <1>       xor     ah, ah
835 000072B8 E85999FFFF          <1>       call   int16h
836                              <1>       ; AL = ASCII Code of the Character
837 000072BD 3C1B                <1>       cmp    al, 27
838 000072BF 0F847B010000        <1>         je      loc_set_time_retn
839 000072C5 A2[1B080100]        <1>       mov    [Hour+1], al
840 000072CA 3C30                <1>       cmp    al, '0'
841 000072CC 0F8283010000        <1>         jb      loc_set_time_stc_1
842 000072D2 3C39                <1>       cmp    al, '9'
843 000072D4 0F877B010000        <1>       ja     loc_set_time_stc_1
844 000072DA 803D[1A080100]32    <1>         cmp     byte [Hour], '2'
845 000072E1 7208                <1>       jb     short pass_set_time_24
846 000072E3 3C34                <1>       cmp    al, '4'
847 000072E5 0F876A010000        <1>         ja      loc_set_time_stc_1
848                              <1> pass_set_time_24:
849                              <1>       ; 13/05/2016
850                              <1>       ;mov   bx, 7 ; attribute/color (bl)
851                              <1>                    ; video page 0 (bh)
852 000072EB B307                <1>       mov    bl, 7
853 000072ED E8C0A9FFFF          <1>       call   _write_tty
854                              <1> loc_enter_time_separator_1:
855 000072F2 28E4                <1>       sub    ah, ah ; 0
856 000072F4 E81D99FFFF          <1>       call   int16h
857                              <1>       ; AL = ASCII Code of the Character
858 000072F9 3C1B                <1>       cmp    al, 27
859 000072FB 0F843F010000        <1>         je      loc_set_time_retn
860 00007301 3C3A                <1>       cmp    al, ':'
861 00007303 0F8567010000        <1>         jne     loc_set_time_stc_2
862                              <1>       ; 13/05/2016
863                              <1>       ;mov   bx, 7 ; attribute/color (bl)
864                              <1>                    ; video page 0 (bh)
865 00007309 B307                <1>       mov    bl, 7
866 0000730B E8A2A9FFFF          <1>       call   _write_tty
867                              <1> loc_enter_minute_1:
868 00007310 30E4                <1>       xor     ah, ah
869 00007312 E8FF98FFFF          <1>       call   int16h
870                              <1>       ; AL = ASCII Code of the Character
871 00007317 3C1B                <1>       cmp    al, 27
872 00007319 0F8421010000        <1>         je      loc_set_time_retn
873 0000731F A2[1D080100]        <1>       mov    [Minute], al
874 00007324 3C30                <1>       cmp    al, '0'
875 00007326 0F825F010000        <1>         jb      loc_set_time_stc_3
876 0000732C 3C35                <1>       cmp    al, '5'
877 0000732E 0F8757010000        <1>         ja      loc_set_time_stc_3
878                              <1>       ; 13/05/2016
879                              <1>       ;mov   bx, 7 ; attribute/color (bl)
880                              <1>                    ; video page 0 (bh)
881 00007334 B307                <1>       mov    bl, 7
882 00007336 E877A9FFFF          <1>       call   _write_tty
883                              <1> loc_enter_minute_2:
884 0000733B 30E4                <1>       xor     ah, ah
885 0000733D E8D498FFFF          <1>       call   int16h
886                              <1>       ; AL = ASCII Code of the Character
887 00007342 3C1B                <1>       cmp    al, 27
888 00007344 0F84F6000000        <1>         je      loc_set_time_retn
889 0000734A A2[1E080100]        <1>       mov    [Minute+1], al
890 0000734F 3C30                <1>       cmp    al, '0'
891 00007351 0F824F010000        <1>         jb      loc_set_time_stc_4
892 00007357 3C39                <1>       cmp    al, '9'
893 00007359 0F8747010000        <1>         ja      loc_set_time_stc_4
894                              <1>       ; 13/05/2016
895                              <1>       ;mov   bx, 7 ; attribute/color (bl)
896                              <1>                    ; video page 0 (bh)
897 0000735F B307                <1>       mov    bl, 7
898 00007361 E84CA9FFFF          <1>       call   _write_tty
899                              <1> loc_enter_time_separator_2:
900 00007366 66C705[20080100]30- <1>       mov    word [Second], 3030h
900 0000736E 30                  <1>
901 0000736F 28E4                <1>       sub     ah, ah
902 00007371 E8A098FFFF          <1>       call   int16h
903                              <1>       ; AL = ASCII Code of the Character
904 00007376 3C0D                <1>       cmp    al, 13
905 00007378 0F8485000000        <1>         je      loc_set_time_progress
906 0000737E 3C1B                <1>       cmp    al, 27
907 00007380 0F84BA000000        <1>         je      loc_set_time_retn
908 00007386 3C3A                <1>       cmp    al, ':'
909 00007388 0F8533010000        <1>         jne     loc_set_time_stc_5
910                              <1>       ; 13/05/2016
911                              <1>       ;mov   bx, 7 ; attribute/color (bl)
912                              <1>                    ; video page 0 (bh)
913 0000738E B307                <1>       mov    bl, 7
914 00007390 E81DA9FFFF          <1>       call   _write_tty
915                              <1> loc_enter_second_1:
916 00007395 30E4                <1>       xor     ah, ah
917 00007397 E87A98FFFF          <1>       call   int16h
918                              <1>       ; AL = ASCII Code of the Character
919 0000739C 3C0D                <1>       cmp    al, 13
920 0000739E 7463                <1>       je     short loc_set_time_progress
921 000073A0 3C1B                <1>       cmp    al, 27
922 000073A2 0F8498000000        <1>         je      loc_set_time_retn
```

```
000073A8 A2[20080100]        <1>       mov    [Second], al
000073AD 3C30                <1>       cmp    al, '0'
000073AF 0F8227010000        <1>         jb      loc_set_time_stc_6
000073B5 3C35                <1>       cmp    al, '5'
000073B7 0F871F010000        <1>         ja      loc_set_time_stc_6
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
000073BD B307                <1>       mov    bl, 7
000073BF E8EEA8FFFF          <1>       call   _write_tty
                             <1> loc_enter_second_2:
000073C4 30E4                <1>       xor     ah, ah
000073C6 E84B98FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
000073CB 3C1B                <1>       cmp    al, 27
000073CD 7471                <1>       je     short loc_set_time_retn
000073CF 3C30                <1>       cmp    al, '0'
000073D1 0F8229010000        <1>         jb      loc_set_time_stc_7
000073D7 3C39                <1>       cmp    al, '9'
000073D9 0F8721010000        <1>         ja      loc_set_time_stc_7
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
000073DF B307                <1>       mov    bl, 7
000073E1 E8CCA8FFFF          <1>       call   _write_tty
                             <1> loc_set_time_get_lchar_again:
000073E6 28E4                <1>       sub    ah, ah ; 0
000073E8 E82998FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
000073ED 3C0D                <1>       cmp    al, 13
000073EF 7412                <1>       je     short loc_set_time_progress
000073F1 3C1B                <1>       cmp    al, 27
000073F3 744B                <1>       je     short loc_set_time_retn
                             <1>       ;
000073F5 E831FEFFFF          <1>       call   check_for_backspace
000073FA 75EA                <1>       jne    short loc_set_time_get_lchar_again
                             <1>
                             <1> loc_set_time_bs_8:
000073FC E818FEFFFF          <1>       call   write_backspace
00007401 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
00007403 E8E5E5FFFF          <1>       call   RTC_20 ; GET RTC TIME
                             <1>       ;DL = Daylight Savings Enable option (0-1)
                             <1>
00007408 66A1[1A080100]      <1>       mov    ax, [Hour]
0000740E 662D3030            <1>       sub    ax, '00'
00007412 C0E004              <1>       shl    al, 4 ; * 16
00007415 88C5                <1>       mov    ch, al
00007417 00E5                <1>       add    ch, ah
00007419 66A1[1D080100]      <1>       mov    ax, [Minute]
0000741F 662D3030            <1>       sub    ax, '00'
00007423 C0E004              <1>       shl    al, 4 ; * 16
00007426 88C1                <1>       mov    cl, al
00007428 00E1                <1>       add    cl, ah
0000742A 66A1[20080100]      <1>       mov    ax, [Second]
00007430 662D3030            <1>       sub    ax, '00'
00007434 C0E004              <1>       shl    al, 4 ; * 16
00007437 88C6                <1>       mov    dh, al
00007439 00E6                <1>       add    dh, ah
                             <1>
                             <1>       ;mov   ah, 03h
                             <1>       ;call  int1Ah
0000743B E8DCE5FFFF          <1>       call   RTC_30 ; SET RTC TIME
                             <1>
                             <1> loc_set_time_retn:
00007440 BE[4B130100]        <1>       mov    esi, nextline
00007445 E813EFFFFF          <1>       call   print_msg
0000744A C3                  <1>       retn
                             <1>
                             <1> loc_set_time_stc_0:
                             <1>       ;xor   bh, bh ; video page 0
0000744B E842A9FFFF          <1>       call   beeper ; BEEP !
00007450 E92EFEFFFF          <1>         jmp     loc_enter_hour_1
                             <1> loc_set_time_stc_1:
00007455 E8D1FDFFFF          <1>       call   check_for_backspace
0000745A 740A                <1>       je     short loc_set_time_bs_1
                             <1>       ;xor   bh, bh ; video page 0
0000745C E831A9FFFF          <1>       call   beeper ; BEEP !
00007461 E950FEFFFF          <1>         jmp     loc_enter_hour_2
                             <1> loc_set_time_bs_1:
00007466 E8AEFDFFFF          <1>       call   write_backspace
0000746B E913FEFFFF          <1>         jmp     loc_enter_hour_1
                             <1> loc_set_time_stc_2:
00007470 E8B6FDFFFF          <1>       call   check_for_backspace
00007475 740A                <1>       je     short loc_set_time_bs_2
                             <1>       ;xor   bh, bh ; video page 0
00007477 E816A9FFFF          <1>       call   beeper ; BEEP !
0000747C E971FEFFFF          <1>         jmp     loc_enter_time_separator_1
                             <1> loc_set_time_bs_2:
00007481 E893FDFFFF          <1>       call   write_backspace
00007486 E92BFEFFFF          <1>         jmp     loc_enter_hour_2
                             <1> loc_set_time_stc_3:
0000748B E89BFDFFFF          <1>       call   check_for_backspace
00007490 740A                <1>       je     short loc_set_time_bs_3
                             <1>       ;xor   bh, bh ; video page 0
00007492 E8FBA8FFFF          <1>       call   beeper ; BEEP !6
00007497 E974FEFFFF          <1>         jmp     loc_enter_minute_1
                             <1> loc_set_time_bs_3:
```

```
1025 0000749C E878FDFFFF          <1>       call   write_backspace
1026 000074A1 E94CFEFFFF          <1>         jmp     loc_enter_time_separator_1
1027                              <1> loc_set_time_stc_4:
1028 000074A6 E880FDFFFF          <1>       call   check_for_backspace
1029 000074AB 740A                <1>       je     short loc_set_time_bs_4
1030                              <1>       ;xor   bh, bh ; video page 0
1031 000074AD E8E0A8FFFF          <1>       call   beeper ; BEEP !
1032 000074B2 E984FEFFFF          <1>         jmp     loc_enter_minute_2
1033                              <1> loc_set_time_bs_4:
1034 000074B7 E85DFDFFFF          <1>       call   write_backspace
1035 000074BC E94FFEFFFF          <1>         jmp     loc_enter_minute_1
1036                              <1> loc_set_time_stc_5:
1037 000074C1 E865FDFFFF          <1>       call   check_for_backspace
1038 000074C6 740A                <1>       je     short loc_set_time_bs_5
1039                              <1>       ;xor   bh, bh ; video page 0
1040 000074C8 E8C5A8FFFF          <1>       call   beeper ; BEEP !
1041 000074CD E994FEFFFF          <1>         jmp     loc_enter_time_separator_2
1042                              <1> loc_set_time_bs_5:
1043 000074D2 E842FDFFFF          <1>       call   write_backspace
1044 000074D7 E95FFEFFFF          <1>         jmp     loc_enter_minute_2
1045                              <1> loc_set_time_stc_6:
1046 000074DC E84AFDFFFF          <1>       call   check_for_backspace
1047 000074E1 7413                <1>       je     short loc_set_time_bs_6
1048                              <1>       ;xor   bh, bh ; video page 0
1049 000074E3 E8AAA8FFFF          <1>       call   beeper ; BEEP !
1050 000074E8 66C705[20080100]30- <1>       mov    word [Second], 3030h
1050 000074F0 30                  <1>
1051 000074F1 E99FFEFFFF          <1>         jmp     loc_enter_second_1
1052                              <1> loc_set_time_bs_6:
1053 000074F6 E81EFDFFFF          <1>       call   write_backspace
1054 000074FB E966FEFFFF          <1>         jmp     loc_enter_time_separator_2
1055                              <1> loc_set_time_stc_7:
1056 00007500 E826FDFFFF          <1>       call   check_for_backspace
1057 00007505 740A                <1>       je     short loc_set_time_bs_7
1058                              <1>       ;xor   bh, bh ; video page 0
1059 00007507 E886A8FFFF          <1>       call   beeper ; BEEP !
1060 0000750C E9B3FEFFFF          <1>         jmp     loc_enter_second_2
1061                              <1> loc_set_time_bs_7:
1062 00007511 E803FDFFFF          <1>       call   write_backspace
1063 00007516 E97AFEFFFF          <1>         jmp     loc_enter_second_1
1064                              <1>
1065                              <1> print_volume_info:
1066                              <1>       ; 01/03/2016
1067                              <1>       ; 08/02/2016
1068                              <1>       ; 06/02/2016
1069                              <1>       ; 04/02/2016
1070                              <1>       ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
1071                              <1>       ; 25/10/2009
1072                              <1>       ;
1073                              <1>       ; "Volume Serial No: "
1074                              <1>       ;
1075                              <1>       ; INPUT  : AL = DOS Drive Number
1076                              <1>       ; OUTPUT : AH = FS Type
1077                              <1>       ;          AL = DOS Drive Name
1078                              <1>       ; CF = 0 -> OK
1079                              <1>       ; CF = 1 -> Drive not ready
1080                              <1>
1081 0000751B 88C4                <1>       mov    ah, al
1082 0000751D 28C0                <1>       sub    al, al
1083 0000751F 0FB7F0              <1>       movzx  esi, ax
1084 00007522 81C600010900        <1>       add    esi, Logical_DOSDisks
1085 00007528 8A06                <1>       mov    al, [esi]
1086 0000752A 3C41                <1>       cmp    al, 'A'
1087 0000752C 7304                <1>       jnb    short loc_pvi_set_vol_name
1088 0000752E 8A6604              <1>       mov    ah, [esi+LD_FSType]
1089 00007531 C3                  <1>       retn
1090                              <1>
1091                              <1> loc_pvi_set_vol_name:
1092 00007532 A2[54080100]        <1>       mov    [Vol_Drv_Name], al
1093 00007537 56                  <1>       push   esi
1094 00007538 E858010000          <1>       call   move_volume_name_and_serial_no ;;;
1095 0000753D 7302                <1>       jnc    short loc_pvi_mvn_ok
1096 0000753F 5E                  <1>       pop    esi
1097 00007540 C3                  <1>       retn
1098                              <1>
1099                              <1> loc_pvi_mvn_ok:
1100 00007541 8B3424              <1>       mov    esi, [esp]
1101 00007544 807E04A1            <1>       cmp    byte [esi+LD_FSType], 0A1h
1102 00007548 7509                <1>       jne    short loc_pvi_fat_vol_size
1103 0000754A 8B4670              <1>       mov    eax, [esi+LD_FS_VolumeSize]
1104 0000754D 0FB75E11            <1>       movzx  ebx, word [esi+LD_FS_BytesPerSec]
1105 00007551 EB07                <1>       jmp    short loc_vol_size_mul32
1106                              <1> loc_pvi_fat_vol_size:
1107 00007553 8B4670              <1>       mov    eax, [esi+LD_TotalSectors]
1108 00007556 0FB75E11            <1>       movzx  ebx, word [esi+LD_BPB+BPB_BytsPerSec]
1109                              <1> loc_vol_size_mul32:
1110 0000755A F7E3                <1>       mul    ebx
1111 0000755C 09D2                <1>       or     edx, edx
1112 0000755E 7507                <1>       jnz    short loc_vol_size_in_kbytes
1113                              <1> loc_vol_size_in_bytes:
1114 00007560 B9[32080100]        <1>       mov    ecx, VolSize_Bytes
1115 00007565 EB0D                <1>       jmp    short loc_write_vol_size_str
1116                              <1> loc_vol_size_in_kbytes:
1117 00007567 66BB0004            <1>       mov    bx, 1024
1118 0000756B F7F3                <1>       div    ebx
1119 0000756D B9[25080100]        <1>       mov    ecx, VolSize_KiloBytes
1120 00007572 31D2                <1>       xor    edx, edx ; 0
1121                              <1> loc_write_vol_size_str:
1122 00007574 890D[FF5A0100]      <1>       mov    [VolSize_Unit1], ecx
1123                              <1>       ;
1124 0000757A BF[155B0100]        <1>       mov    edi, Vol_Tot_Sec_Str_End
1125                              <1>         ;mov byte [edi], 0
```

```
1126 0000757F B90A000000          <1>       mov    ecx, 10
1127                              <1> loc_write_vol_size_chr:
1128 00007584 F7F1                <1>       div    ecx
1129 00007586 80C230              <1>       add    dl, '0'
1130 00007589 4F                  <1>       dec    edi
1131 0000758A 8817                <1>       mov    [edi], dl
1132 0000758C 85C0                <1>       test   eax, eax
1133 0000758E 7404                <1>       jz     short loc_write_vol_size_str_ok
1134 00007590 28D2                <1>       sub    dl, dl ; 0
1135 00007592 EBF0                <1>       jmp    short loc_write_vol_size_chr
1136                              <1>
1137                              <1> loc_write_vol_size_str_ok:
1138 00007594 893D[075B0100]      <1>       mov    [Vol_Tot_Sec_Str_Start], edi
1139                              <1>       ;
1140 0000759A BF[3D080100]        <1>       mov    edi, Vol_FS_Name
1141 0000759F 8A4E03              <1>       mov    cl, [esi+LD_FATType]
1142 000075A2 20C9                <1>       and    cl, cl ; 0 ?
1143 000075A4 7515                <1>       jnz    short loc_write_vol_FAT_str_1
1144 000075A6 66C7075452          <1>       mov    word [edi], 'TR'
1145 000075AB C7470420465331      <1>       mov    dword [edi+4], ' FS1'
1146                              <1>       ;movzx ebx, word [esi+LD_FS_BytesPerSec]
1147 000075B2 668B5E11            <1>       mov    bx, [esi+LD_FS_BytesPerSec]
1148 000075B6 8B4674              <1>       mov    eax, [esi+LD_FS_FreeSectors]
1149 000075B9 EB36                <1>       jmp    short loc_vol_freespace_mul32
1150                              <1>
1151                              <1> loc_write_vol_FAT_str_1:
1152 000075BB 66B83332            <1>       mov    ax, '32' ; FAT32
1153 000075BF 80F902              <1>       cmp    cl, 2 ; [esi+LD_FATType]
1154 000075C2 7708                <1>       ja     short loc_write_vol_FAT_str_2
1155 000075C4 66B83132            <1>       mov    ax, '12' ; FAT12
1156 000075C8 7202                <1>       jb     short loc_write_vol_FAT_str_2
1157 000075CA B436                <1>       mov    ah, '6'  ; FAT16
1158                              <1> loc_write_vol_FAT_str_2:
1159 000075CC C70746415420        <1>       mov    dword [edi], 'FAT '
1160 000075D2 66894704            <1>       mov    word [edi+4], ax
1161                              <1>       ;
1162                              <1>       ;movzx ebx, word [esi+LD_BPB+BPB_BytsPerSec]
1163 000075D6 668B5E11            <1>       mov    bx, [esi+LD_BPB+BPB_BytsPerSec]
1164 000075DA 8B4674              <1>       mov    eax, [esi+LD_FreeSectors]
1165                              <1>
1166                              <1> loc_vol_freespace_recalc0:
1167                              <1>       ; 01/03/2016
1168 000075DD 83F8FF              <1>       cmp    eax, 0FFFFFFFFh
1169 000075E0 720F                <1>       jb     short loc_vol_freespace_mul32
1170                              <1>       ;inc   eax ; 0
1171 000075E2 20C9                <1>       and    cl, cl ; byte [esi+LD_FATType]
1172 000075E4 740B                <1>       jz     short loc_vol_freespace_mul32
1173 000075E6 53                  <1>       push   ebx
1174 000075E7 66BB00FF            <1>       mov    bx, 0FF00h ; recalculate free sectors
1175 000075EB E8A9490000          <1>       call   calculate_fat_freespace
1176 000075F0 5B                  <1>       pop    ebx
1177                              <1>
1178                              <1> loc_vol_freespace_mul32:
1179 000075F1 F7E3                <1>       mul    ebx
1180 000075F3 09D2                <1>       or     edx, edx
1181 000075F5 7507                <1>       jnz    short loc_vol_fspace_in_kbytes
1182                              <1> loc_vol_fspace_in_bytes:
1183 000075F7 B9[32080100]        <1>       mov    ecx, VolSize_Bytes
1184 000075FC EB0D                <1>       jmp    short loc_write_vol_fspace_str
1185                              <1> loc_vol_fspace_in_kbytes:
1186 000075FE 66BB0004            <1>       mov    bx, 1024
1187 00007602 F7F3                <1>       div    ebx
1188 00007604 B9[25080100]        <1>       mov    ecx, VolSize_KiloBytes
1189 00007609 31D2                <1>       xor    edx, edx ; 0
1190                              <1> loc_write_vol_fspace_str:
1191 0000760B 890D[035B0100]      <1>       mov    [VolSize_Unit2], ecx
1192                              <1>       ;
1193 00007611 BF[255B0100]        <1>       mov    edi, Vol_Free_Sectors_Str_End
1194                              <1>         ;mov byte [edi], 0
1195 00007616 B90A000000          <1>       mov    ecx, 10
1196                              <1> loc_write_vol_fspace_chr:
1197 0000761B F7F1                <1>       div    ecx
1198 0000761D 80C230              <1>       add    dl, '0'
1199 00007620 4F                  <1>       dec    edi
1200 00007621 8817                <1>       mov    [edi], dl
1201 00007623 85C0                <1>       test   eax, eax
1202 00007625 7404                <1>       jz     short loc_write_vol_fspace_str_ok
1203 00007627 28D2                <1>       sub    dl, dl ; 0
1204 00007629 EBF0                <1>       jmp    short loc_write_vol_fspace_chr
1205                              <1>
1206                              <1> loc_write_vol_fspace_str_ok:
1207 0000762B 893D[175B0100]      <1>       mov    [Vol_Free_Sectors_Str_Start], edi
1208                              <1>       ;
1209 00007631 BE[3B080100]        <1>       mov    esi, Volume_in_drive
1210 00007636 E822EDFFFF          <1>       call   print_msg
1211 0000763B BE[7B080100]        <1>       mov    esi, Vol_Name
1212 00007640 E818EDFFFF          <1>       call   print_msg
1213 00007645 BE[4B130100]        <1>       mov    esi, nextline
1214 0000764A E80EEDFFFF          <1>       call   print_msg
1215                              <1>       ;
1216 0000764F BE[DC080100]        <1>       mov    esi, Vol_Total_Sector_Header
1217 00007654 E804EDFFFF          <1>       call   print_msg
1218 00007659 8B35[075B0100]      <1>       mov    esi, [Vol_Tot_Sec_Str_Start]
1219 0000765F E8F9ECFFFF          <1>       call   print_msg
1220 00007664 8B35[FF5A0100]      <1>       mov    esi, [VolSize_Unit1]
1221 0000766A E8EEECFFFF          <1>       call   print_msg
1222                              <1>       ;
1223 0000766F BE[ED080100]        <1>       mov    esi, Vol_Free_Sectors_Header
1224 00007674 E8E4ECFFFF          <1>       call   print_msg
1225 00007679 8B35[175B0100]      <1>       mov    esi, [Vol_Free_Sectors_Str_Start]
1226 0000767F E8D9ECFFFF          <1>       call   print_msg
1227 00007684 8B35[035B0100]      <1>       mov    esi, [VolSize_Unit2]
```

```
1228 0000768A E8CEECFFFF          <1>       call   print_msg
1229                              <1>       ;
1230 0000768F 5E                  <1>       pop    esi
1231                              <1>
1232                              <1>       ;mov   ah, [esi+LD_FSType]
1233                              <1>       ;mov   al, [esi+LD_FATType]
1234 00007690 668B4603            <1>       mov    ax, [esi+LD_FATType]
1235                              <1>
1236 00007694 C3                  <1>       retn
1237                              <1>
1238                              <1> move_volume_name_and_serial_no:
1239                              <1>       ; 08/02/2016  (TRDOS 386 = TRDOS v2.0)
1240                              <1>       ; this routine will be called by
1241                              <1>       ; "print_volume_info" and "print_directory"
1242                              <1>       ; INPUT ->
1243                              <1>       ;      ESI = Logical DOS drv descripton table address
1244                              <1>       ; OUTPUT ->
1245                              <1>       ;      *Volume name will be moved to text area
1246                              <1>       ;      *Volume serial number will be converted to
1247                              <1>       ;       text and will be moved to text area
1248                              <1>       ;   cf = 1 -> invalid/unknown dos drive
1249                              <1>       ;   cf = 0 -> ecx = 0
1250                              <1>       ;
1251                              <1>       ; (eax, edx, ecx, esi, edi will be changed)
1252                              <1>
1253 00007695 BF[7B080100]        <1>       mov    edi, Vol_Name
1254                              <1>
1255                              <1>       ;mov   ah, [esi+LD_FSType]
1256                              <1>       ;mov   al, [esi+LD_FATType]
1257 0000769A 668B4603            <1>       mov    ax, [esi+LD_FATType]
1258 0000769E 80FCA1              <1>       cmp    ah, 0A1h
1259 000076A1 7418                <1>       je     short mvn_2
1260 000076A3 08E4                <1>       or     ah, ah
1261 000076A5 7404                <1>       jz     short mvn_0
1262 000076A7 08C0                <1>       or     al, al
1263 000076A9 7504                <1>       jnz    short mvn_1
1264                              <1> mvn_0:
1265 000076AB 8A06                <1>       mov    al, [esi]
1266 000076AD F9                  <1>       stc
1267 000076AE C3                  <1>       retn
1268                              <1> mvn_1:
1269 000076AF 3C02                <1>       cmp    al, 2
1270 000076B1 7717                <1>       ja     short mvn_3
1271                              <1>       ;or    al, al
1272                              <1>       ;jz    short mvn_2
1273 000076B3 8B462D              <1>       mov    eax, [esi+LD_BPB+VolumeID]
1274 000076B6 83C631              <1>       add    esi, LD_BPB+VolumeLabel
1275 000076B9 EB15                <1>       jmp    short mvn_4
1276                              <1> mvn_2:
1277 000076BB 8B4628              <1>       mov    eax, [esi+LD_FS_VolumeSerial]
1278 000076BE 83C62C              <1>       add    esi, LD_FS_VolumeName
1279 000076C1 B910000000          <1>       mov    ecx, 16
1280 000076C6 F3A5                <1>       rep    movsd
1281 000076C8 EB10                <1>       jmp    short mvn_5
1282                              <1> mvn_3:
1283 000076CA 8B4649              <1>       mov    eax, [esi+LD_BPB+FAT32_VolID]
1284 000076CD 83C64D              <1>       add    esi, LD_BPB+FAT32_VolLab
1285                              <1> mvn_4:
1286 000076D0 B90B000000          <1>       mov    ecx, 11
1287 000076D5 F3A4                <1>       rep    movsb
1288 000076D7 C60700              <1>       mov    byte [edi], 0
1289                              <1> mvn_5:
1290                              <1>       ;mov   [Current_VolSerial], eax
1291 000076DA E82ABCFFFF          <1>       call   dwordtohex
1292 000076DF 8915[D0080100]      <1>       mov    [Vol_Serial1], edx
1293 000076E5 A3[D5080100]        <1>       mov    [Vol_Serial2], eax
1294                              <1>       ; ecx = 0
1295 000076EA C3                  <1>       retn
1296                              <1>
1297                              <1> get_volume_serial_number:
1298                              <1>       ; 19/01/2016 (TRDOS 386 = TRDOS v2.0)
1299                              <1>       ; 08/08/2010
1300                              <1>       ;
1301                              <1>       ; INPUT -> DL = Logical DOS Drive number
1302                              <1>       ; OUTPUT -> EAX = Volume serial number
1303                              <1>       ;          BL= FAT Type
1304                              <1>       ;          BH = Logical DOS drv Number (DL input)
1305                              <1>       ; cf = 1 -> Drive not ready
1306                              <1>
1307 000076EB 31DB                <1>       xor    ebx, ebx
1308 000076ED 88D7                <1>       mov    bh, dl
1309 000076EF 3815[AC060100]      <1>       cmp    [Last_DOS_DiskNo], dl
1310 000076F5 7304                <1>       jnb    short loc_gvsn_start
1311                              <1> loc_gvsn_stc_retn:
1312 000076F7 31C0                <1>       xor    eax, eax
1313 000076F9 F9                  <1>       stc
1314 000076FA C3                  <1>         retn
1315                              <1> loc_gvsn_start:
1316 000076FB 56                  <1>       push   esi
1317 000076FC BE00010900          <1>       mov    esi, Logical_DOSDisks
1318 00007701 01DE                <1>       add    esi, ebx
1319 00007703 8A5E03              <1>       mov    bl, [esi+LD_FATType]
1320 00007706 20DB                <1>       and    bl, bl
1321 00007708 740F                <1>       jz     short loc_gvsn_fs
1322 0000770A 80FB02              <1>       cmp    bl, 2
1323 0000770D 7705                <1>       ja     short loc_gvsn_fat32
1324                              <1> loc_gvsn_fat:
1325 0000770F 83C62D              <1>       add    esi, LD_BPB + VolumeID
1326 00007712 EB0E                <1>       jmp    short loc_gvsn_return
1327                              <1> loc_gvsn_fat32:
1328 00007714 83C649              <1>       add    esi, LD_BPB + FAT32_VolID
1329 00007717 EB09                <1>       jmp    short loc_gvsn_return
```

```
1330                                    <1> loc_gvsn_fs:
1331 00007719 807E04A1                 <1>         cmp    byte [esi+LD_FSType], 0A1h
1332 0000771D 75D8                     <1>         jne    short loc_gvsn_stc_retn
1333 0000771F 83C628                   <1>         add    esi, LD_FS_VolumeSerial
1334                                    <1> loc_gvsn_return:
1335 00007722 8B06                     <1>         mov    eax, [esi]
1336 00007724 5E                       <1>         pop    esi
1337 00007725 C3                       <1>         retn
1338                                    <1>
1339                                    <1> ; CMD_INTR.ASM [ TRDOS Command Interpreter Procedure ]
1340                                    <1> ; 09/11/2011
1341                                    <1> ; 29/01/2005
1342                                    <1>
1343                                    <1> command_interpreter:
1344                                    <1>         ; 16/10/2016
1345                                    <1>         ; 12/10/2016
1346                                    <1>         ; 13/05/2016
1347                                    <1>         ; 07/05/2016
1348                                    <1>         ; 04/03/2016
1349                                    <1>         ; 04/02/2016
1350                                    <1>         ; 03/02/2016
1351                                    <1>         ; 30/01/2016
1352                                    <1>         ; 29/01/2016 (TRDOS 386 = TRDOS 2.0)
1353                                    <1>         ; 15/09/2011
1354                                    <1>         ; 29/01/2005
1355                                    <1>
1356                                    <1>         ; Input: ecx = command word length (CL)
1357                                    <1>         ;       CommandBuffer = Command string offset
1358                                    <1>
1359 00007726 C605[B85B0100]00          <1>         mov    byte [Program_Exit],0
1360 0000772D 80F904                   <1>         cmp    cl, 4
1361 00007730 0F87AE020000             <1>           ja      c_6
1362 00007736 0F822E010000             <1>           jb      c_2
1363                                    <1> c_4:
1364                                    <1>
1365                                    <1> cmp_cmd_exit:
1366 0000773C BF[1A070100]             <1>         mov    edi, Cmd_Exit
1367 00007741 E8BA030000               <1>         call   cmp_cmd
1368 00007746 7208                     <1>         jc     short cmp_cmd_date
1369                                    <1>
1370 00007748 C605[B85B0100]01          <1>           mov     byte [Program_Exit], 1
1371 0000774F C3                       <1>           retn
1372                                    <1>
1373                                    <1> cmp_cmd_date:
1374 00007750 B104                     <1>         mov    cl, 4
1375 00007752 BF[36070100]             <1>         mov    edi, Cmd_Date
1376 00007757 E8A4030000               <1>         call   cmp_cmd
1377 0000775C 720B                     <1>           jc   short cmp_cmd_time
1378                                    <1>
1379 0000775E E8D0F7FFFF               <1>         call   show_date
1380 00007763 E80FF8FFFF               <1>         call   set_date
1381 00007768 C3                       <1>         retn
1382                                    <1>
1383                                    <1> cmp_cmd_time:
1384 00007769 B104                     <1>         mov    cl, 4
1385 0000776B BF[3B070100]             <1>         mov    edi, Cmd_Time
1386 00007770 E88B030000               <1>         call   cmp_cmd
1387 00007775 720B                     <1>         jc     short cmp_cmd_show
1388                                    <1>
1389 00007777 E8C6FAFFFF               <1>         call   show_time
1390 0000777C E8F8FAFFFF               <1>         call   set_time
1391 00007781 C3                       <1>         retn
1392                                    <1>
1393                                    <1> cmp_cmd_show:
1394 00007782 B104                     <1>         mov    cl, 4
1395 00007784 BF[4C070100]             <1>         mov    edi, Cmd_Show
1396 00007789 E872030000               <1>         call   cmp_cmd
1397 0000778E 0F83EF090000             <1>           jnc     show_file
1398                                    <1>
1399                                    <1> cmp_cmd_echo:
1400 00007794 B104                     <1>         mov    cl, 4
1401 00007796 BF[88070100]             <1>         mov    edi, Cmd_Echo
1402 0000779B E860030000               <1>         call   cmp_cmd
1403 000077A0 721B                     <1>         jc     short cmp_cmd_copy
1404                                    <1>
1405                                    <1>         ; 14/04/2016
1406 000077A2 56                       <1>         push   esi
1407                                    <1> cmd_echo_asciiz:
1408 000077A3 46                       <1>           inc  esi
1409 000077A4 8A06                     <1>         mov    al, [esi]
1410 000077A6 3C20                     <1>         cmp    al, 20h
1411 000077A8 73F9                     <1>         jnb    short cmd_echo_asciiz
1412 000077AA C60600                   <1>         mov    byte [esi], 0
1413 000077AD 5E                       <1>         pop    esi
1414 000077AE E8AAEBFFFF               <1>         call   print_msg
1415 000077B3 BE[94130100]             <1>         mov    esi, NextLine
1416                                    <1>         ;call  print_msg
1417                                    <1>         ;retn
1418 000077B8 E9A0EBFFFF               <1>         jmp    print_msg
1419                                    <1>
1420                                    <1> cmp_cmd_copy:
1421 000077BD B104                     <1>         mov    cl, 4
1422 000077BF BF[6F070100]             <1>         mov    edi, Cmd_Copy
1423 000077C4 E837030000               <1>         call   cmp_cmd
1424 000077C9 0F8305180000             <1>         jnc    copy_file
1425                                    <1>
1426                                    <1> cmp_cmd_move:
1427 000077CF B104                     <1>         mov    cl, 4
1428 000077D1 BF[74070100]             <1>         mov    edi, Cmd_Move
1429 000077D6 E825030000               <1>         call   cmp_cmd
1430 000077DB 0F8399160000             <1>         jnc    move_file
1431                                    <1>
```

```
1432                                    <1> cmp_cmd_path:
1433 000077E1 B104                     <1>       mov    cl, 4
1434 000077E3 BF[79070100]             <1>       mov    edi, Cmd_Path
1435 000077E8 E813030000               <1>       call   cmp_cmd
1436 000077ED 0F83391A0000             <1>       jnc    set_get_path
1437                                    <1>
1438                                    <1> cmp_cmd_beep:
1439 000077F3 B104                     <1>       mov    cl, 4
1440 000077F5 BF[A6070100]             <1>       mov    edi, Cmd_Beep
1441 000077FA E801030000               <1>       call   cmp_cmd
1442 000077FF 720B                     <1>       jc     short cmp_cmd_find
1443                                    <1>       ; 13/05/2016
1444 00007801 8A3D[2E520100]           <1>       mov    bh, [ptty] ; [ACTIVE_PAGE]
1445 00007807 E986A5FFFF               <1>       jmp    beeper
1446                                    <1>
1447                                    <1> cmp_cmd_find:
1448 0000780C B104                     <1>       mov    cl, 4
1449 0000780E BF[83070100]             <1>       mov    edi, Cmd_Find
1450 00007813 E8E8020000               <1>       call   cmp_cmd
1451 00007818 0F82C5020000             <1>         jc     cmp_cmd_external
1452                                    <1>
1453                                    <1>       ;call  find_and_list_files
1454 0000781E E9EA220000               <1>       jmp    find_and_list_files
1455                                    <1>       ;retn
1456                                    <1>
1457                                    <1> c_1:
1458 00007823 AD                       <1>       lodsd
1459                                    <1> cmp_cmd_help:
1460 00007824 3C3F                     <1>       cmp    al, '?'
1461 00007826 751D                     <1>         jne    short cmp_cmd_remark
1462                                    <1>
1463 00007828 BE[0C070100]             <1>       mov    esi, Command_List
1464                                    <1> cmd_help_next_w:
1465 0000782D E82BEBFFFF               <1>       call   print_msg
1466                                    <1>
1467 00007832 803E20                   <1>       cmp    byte [esi], 20h ; 0
1468 00007835 7232                     <1>       jb     short cmd_help_retn
1469                                    <1>
1470 00007837 56                       <1>       push   esi
1471 00007838 BE[4B130100]             <1>       mov    esi, nextline
1472 0000783D E81BEBFFFF               <1>       call   print_msg
1473 00007842 5E                       <1>       pop    esi
1474 00007843 EBE8                     <1>       jmp    short cmd_help_next_w
1475                                    <1>
1476                                    <1> cmp_cmd_remark:
1477 00007845 3C2A                     <1>       cmp    al, '*'
1478 00007847 0F8596020000             <1>         jne    cmp_cmd_external
1479 0000784D 46                       <1>       inc    esi
1480 0000784E BF[28530100]             <1>       mov    edi, Remark
1481 00007853 8A06                     <1>       mov    al, [esi]
1482 00007855 3C20                     <1>       cmp    al, 20h
1483 00007857 7707                     <1>       ja     short cmd_remark_write
1484 00007859 89FE                     <1>       mov    esi, edi ; Remark
1485 0000785B E9FDEAFFFF               <1>       jmp    print_msg
1486                                    <1>
1487                                    <1> cmd_remark_write:
1488 00007860 AA                       <1>       stosb
1489 00007861 AC                       <1>       lodsb
1490 00007862 3C20                     <1>       cmp    al, 20h
1491 00007864 73FA                     <1>       jnb    short cmd_remark_write
1492 00007866 C60700                   <1>       mov    byte [edi], 0
1493                                    <1>
1494                                    <1> cmd_help_retn:
1495                                    <1> cmd_remark_retn:
1496                                    <1> cd_retn:
1497 00007869 C3                       <1>       retn
1498                                    <1>
1499                                    <1> c_2:
1500 0000786A 80F902                   <1>       cmp    cl, 2
1501 0000786D 0F87B1000000             <1>         ja     c_3
1502 00007873 BE[76530100]             <1>       mov    esi, CommandBuffer
1503 00007878 72A9                     <1>       jb     short c_1
1504                                    <1>
1505                                    <1> cmp_cmd_cd:
1506 0000787A 66AD                     <1>       lodsw
1507 0000787C 663D4344                 <1>       cmp    ax, 'CD'
1508 00007880 7553                     <1>       jne    short cmp_cmd_drive
1509 00007882 46                       <1>         inc  esi
1510                                    <1> cd_0:
1511 00007883 668B06                   <1>       mov    ax, [esi]
1512 00007886 3C20                     <1>       cmp    al, 20h
1513 00007888 76DF                     <1>       jna    short cd_retn
1514                                    <1>       ; 10/02/2016
1515 0000788A 80FC3A                   <1>       cmp    ah, ':'
1516 0000788D 7504                     <1>       jne    short cd_1
1517 0000788F 46                       <1>       inc    esi
1518 00007890 46                       <1>       inc    esi
1519 00007891 EB4B                     <1>       jmp    short cd_2
1520                                    <1>
1521                                    <1> cd_1: ; change current directory
1522                                    <1>       ; 29/11/2009
1523                                    <1>       ; AH = CDh   ; to separate 'CD' command from others
1524                                    <1>                    ; for restoring current directory
1525                                    <1>                    ; 0CDh sign is for saving cdir into
1526                                    <1>                    ; DOS drv description table cdir area
1527                                    <1>
1528 00007893 B4CD                     <1>       mov    ah, 0CDh ; mov byte [CD_COMMAND], 0CDh
1529                                    <1>
1530 00007895 E858230000               <1>       call   change_current_directory
1531 0000789A 0F8372220000             <1>         jnc    change_prompt_dir_string
1532                                    <1>
1533                                    <1> cd_error_messages:
```

```
1534 000078A0 3C03              <1>       cmp    al, 3
1535 000078A2 740C              <1>       je     short cd_path_not_found
1536                            <1>       ; 16/10/2016 (15h -> 15)
1537 000078A4 3C0F              <1>       cmp    al, 15 ; drive not ready error
1538 000078A6 745B              <1>       je     short cd_drive_not_ready
1539 000078A8 3C11              <1>       cmp    al, 17 ; read error
1540 000078AA 7457              <1>       je     short cd_drive_not_ready
1541 000078AC 3C13              <1>       cmp    al, 19 ; ; Bad directory/path name
1542 000078AE 7468              <1>       je     short cd_command_failed
1543                            <1>
1544                            <1> cd_path_not_found:
1545 000078B0 6650              <1>       push   ax
1546 000078B2 BE[AF090100]      <1>       mov    esi, Msg_Dir_Not_Found
1547 000078B7 E8A1EAFFFF        <1>       call   print_msg
1548 000078BC 6658              <1>       pop    ax
1549 000078BE 3A25[C4520100]    <1>       cmp    ah, [Current_Dir_Level]
1550 000078C4 0F8348220000      <1>         jnb     change_prompt_dir_string
1551 000078CA 8825[C4520100]    <1>       mov    [Current_Dir_Level], ah
1552 000078D0 E93D220000        <1>         jmp     change_prompt_dir_string
1553                            <1>
1554                            <1> cmp_cmd_drive: ; change current drive
1555                            <1>       ; C:, D:, E: etc.
1556 000078D5 80FC3A            <1>       cmp    ah, ':'
1557 000078D8 0F8505020000      <1>         jne     cmp_cmd_external
1558                            <1>
1559                            <1> cd_2: ; 'CD C:', 'CD D:' ...
1560 000078DE 803E20            <1>       cmp    byte [esi], 20h
1561 000078E1 0F8706020000      <1>         ja      loc_cmd_failed
1562                            <1>
1563 000078E7 24DF              <1>       and    al, 0DFh
1564 000078E9 2C41              <1>       sub    al, 'A'
1565 000078EB 0F82FC010000      <1>         jc      loc_cmd_failed
1566                            <1>
1567 000078F1 3A05[AC060100]    <1>         cmp     al, [Last_DOS_DiskNo]
1568 000078F7 770A              <1>         ja  short cd_drive_not_ready
1569                            <1>
1570 000078F9 88C2              <1>       mov    dl, al
1571 000078FB E85DF3FFFF        <1>       call   change_current_drive
1572 00007900 7201              <1>       jc     short cd_drive_not_ready
1573 00007902 C3                <1>       retn
1574                            <1>
1575                            <1> cd_drive_not_ready:
1576 00007903 BE[6C090100]      <1>       mov    esi, Msg_Not_Ready_Read_Err
1577 00007908 E850EAFFFF        <1>       call   print_msg
1578                            <1>
1579                            <1> cd_fail_drive_restart:
1580 0000790D 8A15[C6520100]    <1>       mov    dl, [Current_Drv]
1581                            <1>       ;call  change_current_drive
1582 00007913 E945F3FFFF        <1>         jmp     change_current_drive
1583                            <1>       ;retn
1584                            <1>
1585                            <1> cd_command_failed:
1586 00007918 BE[4D090100]      <1>       mov    esi, Msg_Bad_Command
1587 0000791D E83BEAFFFF        <1>       call   print_msg
1588 00007922 EBE9              <1>       jmp    short cd_fail_drive_restart
1589                            <1>
1590                            <1> c_3:
1591                            <1> cmp_cmd_dir:
1592 00007924 BF[0C070100]      <1>       mov    edi, Cmd_Dir
1593 00007929 E8D2010000        <1>       call   cmp_cmd
1594 0000792E 0F8371020000      <1>       jnc    print_directory_list
1595                            <1>
1596                            <1> cmp_cmd_cls:
1597 00007934 B103              <1>       mov    cl, 3
1598 00007936 BF[48070100]      <1>       mov    edi, Cmd_Cls
1599 0000793B E8C0010000        <1>       call   cmp_cmd
1600 00007940 0F832DEAFFFF      <1>         jnc  clear_screen
1601                            <1>
1602                            <1> cmp_cmd_ver:
1603 00007946 B103              <1>       mov    cl, 3
1604 00007948 BF[16070100]      <1>       mov    edi, Cmd_Ver
1605 0000794D E8AE010000        <1>       call   cmp_cmd
1606 00007952 720A              <1>       jc     short cmp_cmd_mem
1607                            <1>
1608 00007954 BE[B4060100]      <1>       mov    esi, mainprog_Version
1609                            <1>       ;call  print_msg
1610 00007959 E9FFE9FFFF        <1>       jmp    print_msg
1611                            <1>       ;retn
1612                            <1>
1613                            <1> cmp_cmd_mem:
1614 0000795E B103              <1>       mov    cl, 3
1615 00007960 BF[7E070100]      <1>       mov    edi, Cmd_Mem
1616 00007965 E896010000        <1>       call   cmp_cmd
1617 0000796A 0F83E4B8FFFF      <1>       jnc    memory_info
1618                            <1>
1619                            <1> cmp_cmd_del:
1620 00007970 B103              <1>       mov    cl, 3
1621 00007972 BF[51070100]      <1>       mov    edi, Cmd_Del
1622 00007977 E884010000        <1>       call   cmp_cmd
1623 0000797C 0F832D0F0000      <1>         jnc     delete_file
1624                            <1>
1625                            <1> cmp_cmd_set:
1626 00007982 B103              <1>       mov    cl, 3
1627 00007984 BF[44070100]      <1>       mov    edi, Cmd_Set
1628 00007989 E872010000        <1>       call   cmp_cmd
1629 0000798E 0F8310180000      <1>         jnc     set_get_env
1630                            <1>
1631                            <1> cmp_cmd_run:
1632 00007994 B103              <1>       mov    cl, 3
1633 00007996 BF[40070100]      <1>       mov    edi, Cmd_Run
1634 0000799B E860010000        <1>       call   cmp_cmd
1635                            <1>       ; 07/05/2016
```

```
1636 000079A0 0F823D010000        <1>         jc      cmp_cmd_external
1637 000079A6 E9471E0000          <1>       jmp    load_and_execute_file
1638                              <1> c_5:
1639                              <1> cmp_cmd_mkdir:
1640 000079AB BF[69070100]        <1>       mov    edi, Cmd_Mkdir
1641 000079B0 E84B010000          <1>       call   cmp_cmd
1642 000079B5 0F838C0A0000        <1>         jnc     make_directory
1643                              <1>
1644                              <1> cmp_cmd_rmdir:
1645 000079BB B105                <1>       mov    cl, 5
1646 000079BD BF[63070100]        <1>       mov    edi, Cmd_Rmdir
1647 000079C2 E839010000          <1>       call   cmp_cmd
1648 000079C7 0F83990B0000        <1>         jnc     delete_directory
1649                              <1>
1650                              <1> cmp_cmd_chdir:
1651 000079CD B105                <1>       mov    cl, 5
1652 000079CF BF[A0070100]        <1>       mov    edi, Cmd_Chdir
1653 000079D4 E827010000          <1>       call   cmp_cmd
1654 000079D9 0F8204010000        <1>         jc      cmp_cmd_external
1655                              <1>
1656 000079DF E99FFEFFFF          <1>       jmp    cd_0
1657                              <1>
1658                              <1> c_6:
1659 000079E4 80F906              <1>       cmp    cl, 6
1660 000079E7 0F87DF000000        <1>         ja      c_8
1661 000079ED 72BC                <1>       jb     short c_5
1662                              <1> cmp_cmd_prompt:
1663 000079EF BF[1F070100]        <1>       mov    edi, Cmd_Prompt
1664 000079F4 E807010000          <1>       call   cmp_cmd
1665 000079F9 722E                <1>         jc   short cmp_cmd_volume
1666                              <1> get_prompt_name_fchar:
1667 000079FB AC                  <1>       lodsb
1668 000079FC 3C20                <1>       cmp    al, 20h
1669 000079FE 74FB                <1>       je     short get_prompt_name_fchar
1670 00007A00 7712                <1>       ja     short loc_change_prompt_label
1671 00007A02 BE[00070100]        <1>       mov    esi, TRDOSPromptLabel
1672 00007A07 C7065452444F        <1>       mov    dword [esi], "TRDO"
1673 00007A0D 66C746045300        <1>              mov    word [esi+4], "S"
1674                              <1> loc_cmd_prompt_return:
1675 00007A13 C3                  <1>       retn
1676                              <1> loc_change_prompt_label:
1677 00007A14 66B90B00            <1>       mov    cx, 11
1678 00007A18 BF[00070100]        <1>       mov    edi, TRDOSPromptLabel
1679                              <1> put_char_new_prompt_label:
1680 00007A1D AA                  <1>       stosb
1681 00007A1E AC                  <1>       lodsb
1682 00007A1F 3C20                <1>       cmp    al, 20h
1683 00007A21 7202                <1>       jb     short pass_put_new_prompt_label
1684 00007A23 E2F8                <1>       loop   put_char_new_prompt_label
1685                              <1> pass_put_new_prompt_label:
1686 00007A25 C60700              <1>       mov    byte [edi], 0
1687 00007A28 C3                  <1>       retn
1688                              <1>
1689                              <1> cmp_cmd_volume:
1690 00007A29 B106                <1>       mov    cl, 6
1691 00007A2B BF[26070100]        <1>       mov    edi, Cmd_Volume
1692 00007A30 E8CB000000          <1>       call   cmp_cmd
1693 00007A35 7255                <1>         jc   short cmp_cmd_attrib
1694                              <1>
1695                              <1> cmd_vol1:
1696 00007A37 AC                  <1>       lodsb
1697 00007A38 3C20                <1>       cmp    al, 20h
1698 00007A3A 7707                <1>       ja     short cmd_vol2
1699 00007A3C A0[C6520100]        <1>       mov    al, [Current_Drv]
1700 00007A41 EB3D                <1>       jmp    short cmd_vol4
1701                              <1> cmd_vol2:
1702 00007A43 3C41                <1>       cmp    al, 'A'
1703 00007A45 0F82A2000000        <1>         jb      loc_cmd_failed
1704 00007A4B 3C7A                <1>       cmp    al, 'z'
1705 00007A4D 0F879A000000        <1>         ja      loc_cmd_failed
1706 00007A53 3C5A                <1>       cmp    al, 'Z'
1707 00007A55 760A                <1>       jna    short cmd_vol3
1708 00007A57 3C61                <1>       cmp    al, 'a'
1709 00007A59 0F828E000000        <1>         jb      loc_cmd_failed
1710 00007A5F 24DF                <1>       and    al, 0DFh
1711                              <1> cmd_vol3:
1712 00007A61 8A26                <1>       mov    ah, [esi]
1713 00007A63 80FC3A              <1>       cmp    ah, ':'
1714 00007A66 0F8581000000        <1>         jne     loc_cmd_failed
1715 00007A6C 2C41                <1>       sub    al, 'A'
1716 00007A6E 3A05[AC060100]      <1>         cmp     al, [Last_DOS_DiskNo]
1717 00007A74 760A                <1>       jna    short cmd_vol4
1718                              <1>
1719 00007A76 BE[6C090100]        <1>       mov    esi, Msg_Not_Ready_Read_Err
1720 00007A7B E9DDE8FFFF          <1>       jmp    print_msg
1721                              <1>
1722                              <1> cmd_vol4:
1723 00007A80 E896FAFFFF          <1>       call   print_volume_info
1724 00007A85 0F8278FEFFFF        <1>         jc      cd_drive_not_ready
1725 00007A8B C3                  <1>       retn
1726                              <1>
1727                              <1> cmp_cmd_attrib:
1728 00007A8C B106                <1>       mov    cl, 6
1729 00007A8E BF[55070100]        <1>       mov    edi, Cmd_Attrib
1730 00007A93 E868000000          <1>       call   cmp_cmd
1731 00007A98 0F83310F0000        <1>         jnc     set_file_attributes
1732                              <1>
1733                              <1> cmp_cmd_rename:
1734 00007A9E B106                <1>       mov    cl, 6
1735 00007AA0 BF[5C070100]        <1>       mov    edi, Cmd_Rename
1736 00007AA5 E856000000          <1>       call   cmp_cmd
1737 00007AAA 0F8367110000        <1>         jnc     rename_file
```

```
                                <1>
                                <1> cmp_cmd_device:
00007AB0 B106                   <1>       mov    cl, 6
00007AB2 BF[91070100]           <1>       mov    edi, Cmd_Device
00007AB7 E844000000             <1>       call   cmp_cmd
00007ABC 7225                   <1>         jc   short cmp_cmd_external
                                <1>
00007ABE C3                     <1>       retn
                                <1>
                                <1> c_7:
                                <1> cmp_cmd_devlist:
00007ABF BF[98070100]           <1>       mov    edi, Cmd_DevList
00007AC4 E837000000             <1>       call   cmp_cmd
00007AC9 7218                   <1>         jc   short cmp_cmd_external
                                <1>
                                <1> loc_cmd_return:
00007ACB C3                     <1>       retn
                                <1>
                                <1> c_8:
00007ACC 80F908                 <1>         cmp  cl, 8
00007ACF 7712                   <1>       ja     short cmp_cmd_external
00007AD1 72EC                   <1>       jb     short c_7
                                <1>
                                <1> cmp_cmd_longname:
00007AD3 BF[2D070100]           <1>       mov    edi, Cmd_LongName
00007AD8 E823000000             <1>       call   cmp_cmd
00007ADD 0F8342060000           <1>         jnc     get_and_print_longname
                                <1>
                                <1> cmp_cmd_external:
                                <1>       ; 07/05/2016
                                <1>       ; 22/04/2016
00007AE3 BE[76530100]           <1>       mov    esi, CommandBuffer
00007AE8 E9051D0000             <1>       jmp    loc_run_check_filename
                                <1>
                                <1> loc_cmd_failed:
00007AED 803D[76530100]20       <1>       cmp    byte [CommandBuffer], 20h
00007AF4 76D5                   <1>       jna    short loc_cmd_return
00007AF6 BE[4D090100]           <1>       mov    esi, Msg_Bad_Command
                                <1> ;     call   print_msg
                                <1> ;loc_cmd_return:
                                <1> ;     retn
00007AFB E95DE8FFFF             <1>       jmp    print_msg
                                <1>
                                <1> cmp_cmd:
                                <1>        ; 29/01/2016 (TRDOS 386 = TRDOS v2.0)
00007B00 BE[76530100]           <1>          mov esi, CommandBuffer
                                <1>          ; edi = internal command word (ASCIIZ)
                                <1>        ; ecx = command length (<=8)
                                <1> cmp_cmd_1:
00007B05 AC                     <1>       lodsb
00007B06 AE                     <1>       scasb
00007B07 750D                   <1>       jne    short cmp_cmd_3
00007B09 E2FA                   <1>       loop   cmp_cmd_1
00007B0B AC                     <1>       lodsb
00007B0C 3C20                   <1>       cmp    al, 20h
00007B0E 7703                   <1>       ja     short cmp_cmd_2
00007B10 30C0                   <1>       xor    al, al
                                <1>       ; ZF = 1 -> internal command word matches
00007B12 C3                     <1>       retn
                                <1> cmp_cmd_2:
                                <1>       ; ZF = 0 (CF = 0) -> external command word
00007B13 58                     <1>       pop    eax ; no return to the caller from here
00007B14 EBCD                   <1>       jmp    cmp_cmd_external
                                <1> cmp_cmd_3:
00007B16 F9                     <1>       stc
                                <1>       ; CF = 1 -> internal command word does not match
00007B17 C3                     <1>       retn
                                <1>
                                <1> loc_run_cmd_failed:
                                <1>       ; 15/03/2016
                                <1>       ; 15/02/2016 (TRDOS 386 =  TRDOS v2.0)
                                <1>       ; 07/12/2009 (CMD_INTR.ASM)
                                <1>       ; 29/11/2009
                                <1>
00007B18 E855000000             <1>       call   restore_cdir_after_cmd_fail
                                <1>
                                <1> loc_run_cmd_failed_cmp_al:
                                <1>       ; End of Restore_CDIR code (29/11/2009)
                                <1>
00007B1D 3C01                   <1>       cmp    al, 1 ; Bad command or file name
00007B1F 74CC                   <1>       je     loc_cmd_failed
                                <1> loc_run_dir_not_found:
00007B21 3C03                   <1>       cmp    al, 3
00007B23 750A                   <1>       jne    short loc_run_file_notfound_msg
                                <1>       ; Path not found (MS-DOS Error Code = 3)
00007B25 BE[AF090100]           <1>       mov    esi, Msg_Dir_Not_Found
00007B2A E92EE8FFFF             <1>       jmp    print_msg
                                <1>
                                <1> loc_run_file_notfound_msg:
00007B2F 3C02                   <1>       cmp    al, 2 ; File not found
00007B31 750A                   <1>       jne    short loc_run_file_drv_read_err
                                <1>
                                <1> loc_print_file_notfound_msg:
00007B33 BE[C6090100]           <1>         mov     esi, Msg_File_Not_Found
                                <1>       ;call  proc_printmsg
                                <1>       ;retn
00007B38 E920E8FFFF             <1>       jmp    print_msg
                                <1>
                                <1> loc_run_file_drv_read_err:
                                <1>       ; Err: 17 (Read fault)
00007B3D 3C11                   <1>       cmp    al, 17 ; Drive not ready or read error
00007B3F 7404                   <1>       je     short loc_run_file_print_drv_read_err
```

```
1840                                    <1>       ;
1841 00007B41 3C0F                     <1>       cmp    al, 15 ; Drive not ready (or read error)
1842 00007B43 750A                     <1>       jne    short loc_run_file_toobig
1843                                    <1>
1844                                    <1> loc_run_file_print_drv_read_err:
1845 00007B45 BE[6C090100]             <1>       mov    esi, Msg_Not_Ready_Read_Err
1846 00007B4A E90EE8FFFF               <1>       jmp    print_msg
1847                                    <1>
1848                                    <1> loc_run_file_toobig:
1849 00007B4F 3C08                     <1>       cmp    al, 8 ; Not enough free memory to load&run file
1850 00007B51 750A                     <1>       jne    short loc_run_misc_error
1851 00007B53 BE[110A0100]             <1>       mov    esi, Msg_Insufficient_Memory
1852 00007B58 E900E8FFFF               <1>       jmp    print_msg
1853                                    <1>
1854                                    <1>       ; 15/03/2016
1855                                    <1> print_misc_error_msg:
1856                                    <1> loc_run_misc_error:
1857                                    <1>       ; AL = Error code
1858 00007B5D E867B7FFFF               <1>       call   bytetohex
1859 00007B62 66A3[450A0100]           <1>         mov     [error_code_hex], ax
1860                                    <1>
1861 00007B68 BE[280A0100]             <1>       mov    esi, Msg_Error_Code
1862                                    <1>       ;call  print_msg
1863                                    <1>       ;retn
1864                                    <1>
1865 00007B6D E9EBE7FFFF               <1>       jmp    print_msg
1866                                    <1>
1867                                    <1> restore_cdir_after_cmd_fail:
1868                                    <1>       ; 15/02/2016 (TRDOS 386 =  TRDOS v2.0)
1869 00007B72 50                       <1>       push   eax
1870 00007B73 8A3D[265B0100]           <1>       mov    bh, [RUN_CDRV] ; it is set at the beginning
1871                                    <1>                              ; of the 'run' command.
1872 00007B79 3A3D[C6520100]           <1>       cmp    bh, [Current_Drv]
1873 00007B7F 7409                     <1>       je     short loc_run_restore_cdir
1874 00007B81 88FA                     <1>       mov    dl, bh
1875 00007B83 E8D5F0FFFF               <1>       call   change_current_drive
1876 00007B88 EB19                     <1>       jmp    short loc_run_err_pass_restore_cdir
1877                                    <1>
1878                                    <1> loc_run_restore_cdir:
1879 00007B8A 803D[AD060100]00         <1>       cmp    byte [Restore_CDIR], 0
1880 00007B91 7610                     <1>       jna    short loc_run_err_pass_restore_cdir
1881 00007B93 30DB                     <1>       xor    bl, bl
1882 00007B95 0FB7F3                   <1>       movzx  esi, bx
1883 00007B98 81C600010900             <1>       add    esi, Logical_DOSDisks
1884 00007B9E E871F1FFFF               <1>       call   restore_current_directory
1885                                    <1>
1886                                    <1> loc_run_err_pass_restore_cdir:
1887 00007BA3 58                       <1>       pop    eax
1888 00007BA4 C3                       <1>       retn
1889                                    <1>
1890                                    <1> print_directory_list:
1891                                    <1>       ; 10/02/2016
1892                                    <1>       ; 08/02/2016 (TRDOS 386 =  TRDOS v2.0)
1893                                    <1>       ; 06/12/2009 ('cmp_cmd_dir')
1894                                    <1>       ;
1895 00007BA5 66C705[685C0100]00-      <1>       mov    word [AttributesMask], 0800h ; ..except volume names..
1895 00007BAD 08                       <1>
1896 00007BAE A0[C6520100]             <1>       mov    al, [Current_Drv]
1897 00007BB3 A2[265B0100]             <1>       mov    [RUN_CDRV], al
1898                                    <1> get_dfname_fchar:
1899 00007BB8 AC                       <1>       lodsb
1900 00007BB9 3C20                     <1>       cmp    al, 20h
1901 00007BBB 74FB                     <1>       je     short get_dfname_fchar
1902 00007BBD 0F82A4000000             <1>         jb      loc_print_dir_call_all
1903 00007BC3 3C2D                     <1>       cmp    al, '-'
1904 00007BC5 7542                     <1>       jne    short loc_print_dir_call_flt
1905                                    <1> get_next_attr_char:
1906 00007BC7 AC                       <1>       lodsb
1907 00007BC8 3C20                     <1>       cmp    al, 20h
1908 00007BCA 74FB                     <1>       je     short get_next_attr_char
1909 00007BCC 0F821BFFFFFF             <1>         jb      loc_cmd_failed
1910 00007BD2 24DF                     <1>       and    al, 0DFh
1911 00007BD4 3C44                     <1>       cmp    al, 'D' ; directories only ?
1912 00007BD6 7512                     <1>       jne    short pass_only_directories
1913 00007BD8 AC                       <1>       lodsb
1914 00007BD9 3C20                     <1>       cmp    al, 20h
1915 00007BDB 0F870CFFFFFF             <1>         ja      loc_cmd_failed
1916 00007BE1 800D[685C0100]10         <1>       or     byte [AttributesMask], 10h ; ..directory..
1917 00007BE8 EB18                     <1>       jmp    short get_dfname_fchar_attr
1918                                    <1> pass_only_directories:
1919 00007BEA 3C46                     <1>       cmp    al, 'F'      ; files only ?
1920 00007BEC 0F85B0000000             <1>         jne     check_attr_s
1921 00007BF2 AC                       <1>       lodsb
1922 00007BF3 3C20                     <1>       cmp    al, 20h
1923 00007BF5 0F87F2FEFFFF             <1>         ja      loc_cmd_failed
1924 00007BFB 800D[695C0100]10         <1>       or     byte [AttributesMask+1], 10h ; ..except directories..
1925                                    <1> get_dfname_fchar_attr:
1926 00007C02 AC                       <1>       lodsb
1927 00007C03 3C20                     <1>       cmp    al, 20h
1928 00007C05 74FB                     <1>       je     short get_dfname_fchar_attr
1929 00007C07 725E                     <1>       jb     short loc_print_dir_call_all
1930                                    <1>
1931                                    <1> loc_print_dir_call_flt:
1932 00007C09 4E                       <1>       dec    esi
1933 00007C0A BF[6A5C0100]             <1>       mov    edi, FindFile_Drv
1934 00007C0F E8F2250000               <1>       call   parse_path_name
1935 00007C14 7308                     <1>       jnc    short loc_print_dir_change_drv_1
1936 00007C16 3C01                     <1>       cmp    al, 1
1937 00007C18 0F87FAFEFFFF             <1>         ja      loc_run_cmd_failed
1938                                    <1>
1939                                    <1> loc_print_dir_change_drv_1:
1940 00007C1E 8A15[6A5C0100]           <1>       mov    dl, [FindFile_Drv]
```

```
1941                                    <1> loc_print_dir_change_drv_2:
1942 00007C24 3A15[265B0100]           <1>         cmp    dl, [RUN_CDRV]
1943 00007C2A 740B                     <1>         je     short loc_print_dir_change_directory
1944 00007C2C E82CF0FFFF               <1>         call   change_current_drive
1945 00007C31 0F82E1FEFFFF             <1>           jc       loc_run_cmd_failed
1946                                    <1> loc_print_dir_change_directory:
1947 00007C37 803D[6B5C0100]20         <1>         cmp    byte [FindFile_Directory], 20h ; 0 or 20h ?
1948 00007C3E 761D                     <1>         jna    short pass_print_dir_change_directory
1949                                    <1>
1950 00007C40 FE05[AD060100]           <1>         inc    byte [Restore_CDIR]
1951 00007C46 BE[6B5C0100]             <1>         mov    esi, FindFile_Directory
1952 00007C4B 30E4                     <1>         xor    ah, ah ; CD_COMMAND sign -> 0
1953 00007C4D E8A01F0000               <1>         call   change_current_directory
1954 00007C52 0F82C0FEFFFF             <1>           jc       loc_run_cmd_failed
1955                                    <1>
1956                                    <1> loc_print_dir_change_prompt_dir_string:
1957 00007C58 E8B51E0000               <1>         call   change_prompt_dir_string
1958                                    <1>
1959                                    <1> pass_print_dir_change_directory:
1960 00007C5D BE[AC5C0100]             <1>         mov    esi, FindFile_Name
1961 00007C62 803E20                   <1>         cmp    byte [esi], 20h ; ; 0 or 20h ?
1962 00007C65 7706                     <1>         ja     short loc_print_dir_call
1963                                    <1>
1964                                    <1> loc_print_dir_call_all:
1965 00007C67 C7062A2E2A00             <1>         mov    dword [esi], '*.*'
1966                                    <1> loc_print_dir_call:
1967 00007C6D E87E000000               <1>         call   print_directory
1968                                    <1>
1969 00007C72 8A15[265B0100]           <1>         mov    dl, [RUN_CDRV]  ; it is set at the beginning
1970 00007C78 3A15[C6520100]           <1>         cmp    dl, [Current_Drv]
1971 00007C7E 7406                     <1>         je     short loc_print_dir_call_restore_cdir_retn
1972 00007C80 E8D8EFFFFF               <1>         call   change_current_drive
1973 00007C85 C3                       <1>         retn
1974                                    <1>
1975                                    <1> loc_print_dir_call_restore_cdir_retn:
1976 00007C86 803D[AD060100]00         <1>         cmp    byte [Restore_CDIR], 0
1977 00007C8D 7610                     <1>         jna    short pass_print_dir_call_restore_cdir_retn
1978                                    <1>
1979 00007C8F BE00010900               <1>         mov    esi, Logical_DOSDisks
1980 00007C94 31C0                     <1>         xor    eax, eax
1981 00007C96 88D4                     <1>         mov    ah, dl
1982 00007C98 01C6                     <1>         add    esi, eax
1983                                    <1>
1984 00007C9A E875F0FFFF               <1>         call   restore_current_directory
1985                                    <1>
1986                                    <1> pass_print_dir_call_restore_cdir_retn:
1987 00007C9F C3                       <1>         retn
1988                                    <1>
1989                                    <1> check_attr_s_cap:
1990 00007CA0 24DF                     <1>         and    al, 0DFh
1991                                    <1> check_attr_s:
1992 00007CA2 3C53                     <1>         cmp    al, 'S'
1993 00007CA4 7514                     <1>         jne    short pass_attr_s
1994 00007CA6 800D[685C0100]04         <1>         or     byte [AttributesMask], 4 ; system
1995 00007CAD AC                       <1>         lodsb
1996 00007CAE 3C20                     <1>         cmp    al, 20h
1997 00007CB0 0F844CFFFFFF             <1>           je       get_dfname_fchar_attr
1998 00007CB6 72AF                     <1>         jb     short loc_print_dir_call_all
1999 00007CB8 24DF                     <1>         and    al, 0DFh
2000                                    <1> pass_attr_s:
2001 00007CBA 3C48                     <1>         cmp    al, 'H'
2002 00007CBC 7514                     <1>         jne    short pass_attr_h
2003 00007CBE 800D[685C0100]02         <1>         or     byte [AttributesMask], 2 ; hidden
2004                                    <1> pass_attr_shr:
2005 00007CC5 AC                       <1>         lodsb
2006 00007CC6 3C20                     <1>         cmp    al, 20h
2007 00007CC8 0F8434FFFFFF             <1>           je       get_dfname_fchar_attr
2008 00007CCE 7297                     <1>         jb     short loc_print_dir_call_all
2009 00007CD0 EBCE                     <1>         jmp    short check_attr_s_cap
2010                                    <1>
2011                                    <1> pass_attr_h:
2012 00007CD2 3C52                     <1>         cmp    al, 'R'
2013 00007CD4 7509                     <1>         jne    short pass_attr_r
2014 00007CD6 800D[685C0100]01         <1>         or     byte [AttributesMask], 1 ; read only
2015 00007CDD EBE6                     <1>         jmp    short pass_attr_shr
2016                                    <1>
2017                                    <1> pass_attr_r:
2018 00007CDF 3C41                     <1>         cmp    al, 'A'
2019 00007CE1 0F8506FEFFFF             <1>           jne      loc_cmd_failed
2020 00007CE7 800D[685C0100]20         <1>         or     byte [AttributesMask], 20h ; archive
2021 00007CEE EBD5                     <1>         jmp    short pass_attr_shr
2022                                    <1>
2023                                    <1> print_directory:
2024                                    <1>         ; 13/05/2016
2025                                    <1>         ; 11/02/2016
2026                                    <1>         ; 10/02/2016
2027                                    <1>         ; 08/02/2016 (TRDOS 386 =  TRDOS v2.0)
2028                                    <1>         ; 30/10/2010 ('proc_print_directory')
2029                                    <1>         ; 19/09/2009
2030                                    <1>         ; 2005
2031                                    <1>         ; INPUT ->
2032                                    <1>         ;      ESI = Asciiz File/Dir Name Address
2033                                    <1>
2034 00007CF0 56                       <1>         push   esi
2035                                    <1>
2036 00007CF1 29C0                     <1>         sub    eax, eax
2037                                    <1>
2038 00007CF3 66A3[F45C0100]           <1>         mov    word [Dir_Count], ax ; 0
2039 00007CF9 66A3[F25C0100]           <1>         mov    word [File_Count], ax ; 0
2040 00007CFF A3[F65C0100]             <1>         mov    dword [Total_FSize], eax ; 0
2041                                    <1>
2042 00007D04 E86AE6FFFF               <1>         call    clear_screen
```

```
2043                                    <1>
2044 00007D09 31C9                      <1>       xor    ecx, ecx
2045 00007D0B 8A2D[C6520100]            <1>       mov     ch, [Current_Drv] ; DirBuff_Drv - 'A'
2046 00007D11 A0[C7520100]              <1>       mov     al, [Current_Dir_Drv]
2047 00007D16 A2[6A080100]              <1>       mov     [Dir_Drive_Name], al
2048 00007D1B BE00010900                <1>       mov    esi, Logical_DOSDisks
2049 00007D20 01CE                      <1>       add    esi, ecx
2050                                    <1>
2051 00007D22 E86EF9FFFF                <1>       call   move_volume_name_and_serial_no
2052 00007D27 730C                      <1>       jnc    short print_dir_strlen_check
2053                                    <1>
2054 00007D29 5E                        <1>       pop    esi
2055 00007D2A 8A3D[2E520100]            <1>       mov    bh, [ptty] ; [ACTIVE_PAGE]
2056                                    <1>       ;call  beeper
2057                                    <1>       ;retn
2058 00007D30 E95DA0FFFF                <1>       jmp    beeper  ; beep ! and return
2059                                    <1>
2060                                    <1> print_dir_strlen_check:
2061 00007D35 BE[C9520100]              <1>       mov    esi, Current_Dir_Root
2062 00007D3A BF[07090100]              <1>       mov    edi, Dir_Str_Root
2063                                    <1>
2064                                    <1>       ;xor   ecx, ecx
2065 00007D3F 8A0D[25530100]            <1>         mov     cl, [Current_Dir_StrLen]
2066 00007D45 FEC1                      <1>       inc    cl
2067 00007D47 80F940                    <1>       cmp    cl, 64
2068 00007D4A 760D                      <1>       jna    short pass_print_dir_strlen_shorting
2069 00007D4C 46                        <1>       inc    esi
2070 00007D4D 01CE                      <1>       add    esi, ecx
2071 00007D4F 83EE40                    <1>       sub    esi, 64
2072 00007D52 47                        <1>       inc    edi
2073 00007D53 B82E2E2E20                <1>       mov    eax, '... '
2074 00007D58 AB                        <1>       stosd
2075                                    <1>
2076                                    <1> pass_print_dir_strlen_shorting:
2077 00007D59 F3A4                      <1>       rep    movsb
2078                                    <1>
2079 00007D5B BE[5D080100]              <1>       mov    esi, Dir_Drive_Str
2080 00007D60 E8F8E5FFFF                <1>       call   print_msg
2081                                    <1>
2082 00007D65 BE[BC080100]              <1>       mov    esi, Vol_Serial_Header
2083 00007D6A E8EEE5FFFF                <1>       call   print_msg
2084                                    <1>
2085 00007D6F BE[FC080100]              <1>       mov    esi, Dir_Str_Header
2086 00007D74 E8E4E5FFFF                <1>       call   print_msg
2087                                    <1>
2088 00007D79 BE[49130100]              <1>       mov    esi, next2line
2089 00007D7E E8DAE5FFFF                <1>       call   print_msg
2090                                    <1>
2091                                    <1> loc_print_dir_first_file:
2092 00007D83 C605[095D0100]10          <1>       mov    byte [PrintDir_RowCounter], 16
2093 00007D8A 66A1[685C0100]            <1>       mov    ax, [AttributesMask]
2094 00007D90 5E                        <1>       pop    esi
2095                                    <1>
2096 00007D91 E859020000                <1>       call   find_first_file
2097 00007D96 0F826F010000              <1>         jc      loc_dir_ok
2098                                    <1>
2099                                    <1> loc_dfname_use_this:
2100                                    <1>       ; bl = File Attributes (bh = Long Name Entry Length)
2101 00007D9C F6C310                    <1>       test   bl, 10h  ; Is it a directory?
2102 00007D9F 741B                      <1>       jz     short loc_not_dir
2103                                    <1>
2104 00007DA1 66FF05[F45C0100]          <1>       inc    word [Dir_Count]
2105 00007DA8 89F2                      <1>       mov    edx, esi     ; FindFile_DirEntry address
2106 00007DAA BE[4C0A0100]              <1>       mov    esi, Type_Dir; '<DIR>     '
2107 00007DAF BF[630A0100]              <1>       mov    edi, Dir_Or_FileSize
2108                                    <1>       ; move 10 bytes
2109 00007DB4 A5                        <1>       movsd
2110 00007DB5 A5                        <1>       movsd
2111 00007DB6 66A5                      <1>       movsw
2112 00007DB8 89D6                      <1>       mov    esi, edx
2113 00007DBA EB36                      <1>       jmp     short loc_dir_attribute
2114                                    <1>
2115                                    <1> loc_not_dir:
2116 00007DBC 66FF05[F25C0100]          <1>       inc    word [File_Count]
2117 00007DC3 0105[F65C0100]            <1>       add    [Total_FSize], eax
2118                                    <1>
2119 00007DC9 B90A000000                <1>       mov    ecx, 10  ; 32 bit divisor
2120 00007DCE 89CF                      <1>       mov    edi, ecx
2121 00007DD0 81C7[630A0100]            <1>       add    edi, Dir_Or_FileSize
2122                                    <1> loc_dir_rdivide:
2123 00007DD6 29D2                      <1>       sub    edx, edx
2124 00007DD8 F7F1                      <1>       div    ecx     ; remainder in dl (< 10)
2125 00007DDA 80C230                    <1>       add     dl, '0'      ; to make visible (ascii)
2126 00007DDD 4F                        <1>       dec    edi
2127 00007DDE 8817                      <1>       mov     [edi], dl
2128 00007DE0 21C0                      <1>       and    eax, eax
2129 00007DE2 75F2                      <1>       jnz    short loc_dir_rdivide
2130                                    <1>
2131                                    <1> loc_dir_fill_space:
2132 00007DE4 81FF[630A0100]            <1>       cmp     edi, Dir_Or_FileSize
2133 00007DEA 7606                      <1>       jna     short loc_dir_attribute
2134 00007DEC 4F                        <1>       dec     edi
2135 00007DED C60720                    <1>       mov     byte [edi], 20h
2136 00007DF0 EBF2                      <1>       jmp     short loc_dir_fill_space
2137                                    <1>
2138                                    <1> loc_dir_attribute:
2139 00007DF2 C705[6E0A0100]2020-       <1>       mov    dword [File_Attribute], 20202020h
2139 00007DFA 2020                      <1>
2140                                    <1>
2141 00007DFC 80FB20                    <1>       cmp    bl, 20h  ; Is it an archive file?
2142 00007DFF 7207                      <1>       jb     short loc_dir_pass_arch
2143 00007E01 C605[710A0100]41          <1>       mov    byte [File_Attribute+3], 'A'
```

```
                                <1>
                                <1> loc_dir_pass_arch:
00007E08 80E307                 <1>       and    bl, 7
00007E0B 7428                   <1>       jz     short loc_dir_file_name
00007E0D 88DF                   <1>       mov    bh, bl
00007E0F 80E303                 <1>       and    bl, 3
00007E12 38DF                   <1>       cmp    bh, bl
00007E14 7607                   <1>       jna    short loc_dir_pass_s
00007E16 C605[6E0A0100]53       <1>       mov    byte [File_Attribute], 'S'
                                <1>
                                <1> loc_dir_pass_s:
00007E1D 80E302                 <1>       and     bl,2
00007E20 7407                   <1>       jz      short loc_dir_pass_h
00007E22 C605[6F0A0100]48       <1>       mov     byte [File_Attribute+1], 'H'
                                <1> loc_dir_pass_h:
00007E29 80E701                 <1>       and     bh,1
00007E2C 7407                   <1>       jz      short loc_dir_file_name
00007E2E C605[700A0100]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
00007E35 8B5E16                 <1>       mov    ebx, [esi+16h]
00007E38 89F1                   <1>       mov    ecx, esi ; FindFile_DirEntry address
00007E3A BF[560A0100]           <1>       mov     edi, File_Name
                                <1>       ; move 8 bytes
00007E3F A5                     <1>       movsd
00007E40 A5                     <1>       movsd
00007E41 C60720                 <1>       mov    byte [edi], 20h
00007E44 47                     <1>       inc    edi
                                <1>       ; move 3 bytes
00007E45 66A5                   <1>       movsw
00007E47 A4                     <1>       movsb
00007E48 89CE                   <1>       mov    esi, ecx
                                <1>
                                <1> Dir_Time_start:
                                <1>       ;mov   ax, dx       ; Time
00007E4A 6689D8                 <1>       mov    ax, bx
00007E4D 66C1E805               <1>       shr    ax, 5        ; shift right 5 times
00007E51 6683E03F               <1>       and    ax, 0000111111b     ; Minute Mask
00007E55 D40A                   <1>       aam                 ; Q([AL]/10)->AH
                                <1>                           ; R([AL]/10)->AL
                                <1>                           ; [AL]+[AH]= Minute as BCD
00007E57 660D3030               <1>       or     ax, '00'     ; Convert to ASCII
00007E5B 86E0                   <1>       xchg   ah, al
00007E5D 66A3[810A0100]         <1>       mov    [File_Minute], ax
                                <1>
                                <1>       ;mov   al, dh
00007E63 88F8                   <1>       mov    al, bh
00007E65 C0E803                 <1>       shr    al, 3        ; shift right 3 times
00007E68 D40A                   <1>       aam                 ; [AL]+[AH]= Hours as BCD
00007E6A 660D3030               <1>       or     ax, '00'
00007E6E 86E0                   <1>       xchg   ah, al
00007E70 66A3[7E0A0100]         <1>       mov     [File_Hour], ax
                                <1>
00007E76 C1EB10                 <1>       shr    ebx, 16             ; BX = Date
                                <1>
                                <1> Dir_Date_start:
00007E79 6689D8                 <1>       mov    ax, bx       ; Date
00007E7C 6683E01F               <1>       and    ax, 00011111b; Day Mask
00007E80 D40A                   <1>       aam                 ; Q([AL]/10)->AH
                                <1>                           ; R([AL]/10)->AL
                                <1>                           ; [AL]+[AH]= Day as BCD
00007E82 660D3030               <1>       or     ax, '00'     ; Convert to ASCII
00007E86 86C4                   <1>       xchg   al, ah
                                <1>
00007E88 66A3[730A0100]         <1>       mov    [File_Day], ax
                                <1>
00007E8E 6689D8                 <1>       mov    ax, bx
00007E91 66C1E805               <1>       shr    ax, 5        ; shift right 5 times
00007E95 6683E00F               <1>       and    ax, 00001111b; Month Mask
00007E99 D40A                   <1>       aam
00007E9B 660D3030               <1>       or     ax, '00'
00007E9F 86E0                   <1>       xchg   ah, al
00007EA1 66A3[760A0100]         <1>       mov    [File_Month], ax
                                <1>
00007EA7 6689D8                 <1>       mov    ax, bx
00007EAA 66C1E809               <1>       shr     ax, 9
00007EAE 6683E07F               <1>       and    ax, 01111111b; Result = Year - 1980
00007EB2 6605BC07               <1>       add    ax, 1980
                                <1>
00007EB6 B10A                   <1>       mov    cl, 10
00007EB8 F6F1                   <1>       div    cl           ; Q -> AL, R -> AH
00007EBA 80CC30                 <1>       or     ah, '0'
00007EBD 8825[7C0A0100]         <1>       mov    [File_Year+3], ah
00007EC3 D40A                   <1>       aam
00007EC5 86E0                   <1>       xchg   ah, al
00007EC7 80CC30                 <1>       or     ah, '0'        ; Convert to ASCII
00007ECA 8825[7B0A0100]         <1>       mov    [File_Year+2], ah
00007ED0 D40A                   <1>       aam
00007ED2 86C4                   <1>       xchg   al, ah
00007ED4 660D3030               <1>       or     ax, '00'
00007ED8 66A3[790A0100]         <1>       mov    [File_Year], ax
                                <1>
                                <1> loc_show_line:
00007EDE 56                     <1>       push   esi
00007EDF BE[560A0100]           <1>       mov     esi, File_Name
00007EE4 E874E4FFFF             <1>       call   print_msg
00007EE9 BE[4B130100]           <1>       mov    esi, nextline
00007EEE E86AE4FFFF             <1>       call   print_msg
00007EF3 5E                     <1>       pop    esi
                                <1>
00007EF4 FE0D[095D0100]         <1>       dec    byte [PrintDir_RowCounter]
```

```
2246 00007EFA 0F84D4000000        <1>         jz      pause_dir_scroll
2247                              <1>
2248                              <1> loc_next_entry:
2249 00007F00 E899010000          <1>       call   find_next_file
2250 00007F05 0F8391FEFFFF        <1>         jnc     loc_dfname_use_this
2251                              <1>
2252                              <1> loc_dir_ok:
2253 00007F0B B90A000000          <1>       mov     ecx, 10
2254 00007F10 66A1[F45C0100]      <1>       mov    ax, [Dir_Count]
2255 00007F16 BF[970A0100]        <1>       mov    edi, Decimal_Dir_Count
2256 00007F1B 6639C8              <1>       cmp    ax, cx ; 10
2257 00007F1E 7216                <1>       jb     short pass_ddc
2258 00007F20 47                  <1>       inc    edi
2259 00007F21 6683F864            <1>       cmp    ax, 100
2260 00007F25 720F                <1>       jb     short pass_ddc
2261 00007F27 47                  <1>       inc    edi
2262 00007F28 663DE803            <1>       cmp    ax, 1000
2263 00007F2C 7208                <1>       jb     short pass_ddc
2264 00007F2E 47                  <1>       inc    edi
2265 00007F2F 663D1027            <1>       cmp    ax, 10000
2266 00007F33 7201                <1>       jb     short pass_ddc
2267 00007F35 47                  <1>       inc    edi
2268                              <1> pass_ddc:
2269 00007F36 886F01              <1>       mov     [edi+1], ch ; 0
2270                              <1> loc_ddc_rediv:
2271 00007F39 31D2                <1>       xor     edx, edx
2272 00007F3B 66F7F1              <1>       div     cx   ; 10
2273 00007F3E 80C230              <1>       add     dl, '0'
2274 00007F41 8817                <1>       mov     [edi], dl
2275 00007F43 4F                  <1>       dec     edi
2276 00007F44 6609C0              <1>       or     ax, ax
2277 00007F47 75F0                <1>       jnz    short loc_ddc_rediv
2278                              <1>
2279 00007F49 66A1[F25C0100]      <1>       mov     ax, [File_Count]
2280 00007F4F BF[860A0100]        <1>       mov     edi, Decimal_File_Count
2281 00007F54 6639C8              <1>       cmp     ax, cx ; 10
2282 00007F57 7216                <1>       jb      short pass_dfc
2283 00007F59 47                  <1>       inc     edi
2284 00007F5A 6683F864            <1>       cmp     ax, 100
2285 00007F5E 720F                <1>       jb      short pass_dfc
2286 00007F60 47                  <1>       inc     edi
2287 00007F61 663DE803            <1>       cmp     ax, 1000
2288 00007F65 7208                <1>       jb      short pass_dfc
2289 00007F67 47                  <1>       inc     edi
2290 00007F68 663D1027            <1>       cmp     ax, 10000
2291 00007F6C 7201                <1>       jb      short pass_dfc
2292 00007F6E 47                  <1>       inc     edi
2293                              <1> pass_dfc:
2294                              <1>       ;mov    cx, 10
2295 00007F6F 886F01              <1>       mov     [edi+1], ch ; 00
2296                              <1> loc_dfc_rediv:
2297                              <1>       ;xor   dx, dx
2298 00007F72 30D2                <1>       xor    dl, dl
2299 00007F74 66F7F1              <1>       div    cx
2300 00007F77 80C230              <1>       add    dl, '0'
2301 00007F7A 8817                <1>       mov    [edi], dl
2302 00007F7C 4F                  <1>       dec    edi
2303 00007F7D 6609C0              <1>       or     ax, ax
2304 00007F80 75F0                <1>       jnz    short loc_dfc_rediv
2305                              <1>
2306 00007F82 BF[085D0100]        <1>       mov     edi, TFS_Dec_End
2307                              <1>         ;mov    byte [edi], 0
2308 00007F87 A1[F65C0100]        <1>       mov     eax, [Total_FSize]
2309                              <1>       ;mov    ecx, 10
2310                              <1> rediv_tfs_hex:
2311                              <1>       ;sub   edx, edx
2312 00007F8C 28D2                <1>       sub    dl, dl
2313 00007F8E F7F1                <1>       div    ecx
2314 00007F90 80C230              <1>       add    dl, '0'
2315 00007F93 4F                  <1>       dec     edi
2316 00007F94 8817                <1>       mov     [edi], dl
2317 00007F96 21C0                <1>       and    eax, eax
2318 00007F98 75F2                <1>       jnz    short rediv_tfs_hex
2319                              <1>
2320 00007F9A 893D[FA5C0100]      <1>       mov    [TFS_Dec_Begin], edi
2321 00007FA0 BE[840A0100]        <1>       mov    esi, Decimal_File_Count_Header
2322 00007FA5 E8B3E3FFFF          <1>       call   print_msg
2323 00007FAA BE[8C0A0100]        <1>       mov    esi, str_files
2324 00007FAF E8A9E3FFFF          <1>       call   print_msg
2325 00007FB4 BE[9D0A0100]        <1>       mov    esi, str_dirs
2326 00007FB9 E89FE3FFFF          <1>       call   print_msg
2327 00007FBE 8B35[FA5C0100]      <1>       mov    esi, [TFS_Dec_Begin]
2328 00007FC4 E894E3FFFF          <1>       call   print_msg
2329 00007FC9 BE[AE0A0100]        <1>       mov    esi, str_bytes
2330 00007FCE E88AE3FFFF          <1>       call   print_msg
2331                              <1>
2332 00007FD3 C3                  <1>       retn
2333                              <1>
2334                              <1> pause_dir_scroll:
2335 00007FD4 28E4                <1>       sub    ah, ah
2336 00007FD6 E83B8CFFFF          <1>       call   int16h
2337 00007FDB 3C1B                <1>       cmp    al, 1Bh
2338 00007FDD 0F8428FFFFFF        <1>         je      loc_dir_ok
2339 00007FE3 C605[095D0100]10    <1>       mov    byte [PrintDir_RowCounter], 16 ; Reset counter
2340 00007FEA E911FFFFFF          <1>         jmp     loc_next_entry
2341                              <1>
2342                              <1> find_first_file:
2343                              <1>       ; 11/02/2016
2344                              <1>       ; 10/02/2016
2345                              <1>       ; 08/02/2016 (TRDOS 386 =  TRDOS v2.0)
2346                              <1>       ; 09/10/2011
2347                              <1>       ; 17/09/2009
```

```
                                <1>       ; 2005
                                <1>       ; INPUT ->
                                <1>       ;     ESI = ASCIIZ File/Dir Name Address (in Current Directory)
                                <1>       ;     AL = Attributes AND mask (The AND result must be equal to AL)
                                <1>       ;           bit 0 = Read Only
                                <1>       ;           bir 1 = Hidden
                                <1>       ;           bit 2 = System
                                <1>       ;           bit 3 = Volume Label
                                <1>       ;           bit 4 = Directory
                                <1>       ;           bit 5 = Archive
                                <1>       ;           bit 6 = Reserved, must be 0
                                <1>       ;           bit 7 = Reserved, must be 0
                                <1>       ;      AH = Attributes Negative AND mask (The AND result must be ZERO)
                                <1>       ;
                                <1>       ; OUTPUT ->
                                <1>       ;     CF = 1 -> Error, Error Code in EAX (AL)
                                <1>       ;     CF = 0 ->
                                <1>       ;          ESI = Directory Entry (FindFile_DirEntry) Location
                                <1>       ;          EDI = Directory Buffer Directory Entry Location
                                <1>       ;          EAX = File Size
                                <1>       ;           BL = Attributes of The File/Directory
                                <1>       ;           BH = Long Name Yes/No Status (>0 is YES)
                                <1>       ;             DX > 0 : Ambiguous filename chars are used
                                <1>       ;
                                <1>       ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)
                                <1>
00007FEF 66A3[BA5C0100]         <1>       mov   [FindFile_AttributesMask], ax
00007FF5 BF[BC5C0100]           <1>       mov   edi, FindFile_DirEntry ; TR-DOS Fullfilename formatted buffer
00007FFA 31C0                   <1>       xor   eax, eax
00007FFC B90B000000             <1>       mov   ecx, 11
00008001 F3AB                   <1>       rep   stosd  ; 44 bytes
                                <1>       ;stosw       ; +2 bytes
                                <1>
00008003 BF[AC5C0100]           <1>       mov   edi, FindFile_Name ; FFF structure, offset 66
00008008 39FE                   <1>       cmp   esi, edi
0000800A 7408                   <1>       je    short loc_fff_mfn_ok
0000800C 89FA                   <1>       mov   edx, edi
                                <1>        ; move 13 bytes
0000800E A5                     <1>       movsd
0000800F A5                     <1>       movsd
00008010 A5                     <1>       movsd
00008011 AA                     <1>       stosb
00008012 89D6                   <1>       mov   esi, edx
                                <1> loc_fff_mfn_ok:
00008014 BF[5B5C0100]           <1>       mov   edi, Dir_Entry_Name ; Dir Entry Format File Name
00008019 E81D210000             <1>       call  convert_file_name
0000801E 89FE                   <1>       mov   esi, edi ; offset Dir_Entry_Name
                                <1>
00008020 66A1[BA5C0100]         <1>       mov   ax, [FindFile_AttributesMask]
                                <1>       ;xor  ecx, ecx
00008026 30C9                   <1>       xor   cl, cl
00008028 E8191E0000             <1>       call  locate_current_dir_file
0000802D 726E                   <1>       jc    short loc_fff_retn
                                <1>       ; EDI = Directory Entry
                                <1>       ; EBX = Directory Buffer Entry Index/Number
                                <1>
                                <1> loc_fff_fnf_ln_check:
0000802F 30ED                   <1>       xor   ch, ch
00008031 80F60F                 <1>       xor   dh, 0Fh
00008034 7408                   <1>       jz    short loc_fff_longname_yes
00008036 882D[B95C0100]         <1>       mov   [FindFile_LongNameYes], ch ; 0
0000803C EB0C                   <1>       jmp   short loc_fff_longname_no
                                <1>
                                <1> loc_fff_longname_yes:
                                <1>       ;inc   byte [FindFile_LongNameYes]
0000803E 8A0D[C65B0100]         <1>       mov   cl, [LFN_EntryLength]
00008044 880D[B95C0100]         <1>       mov   [FindFile_LongNameEntryLength], cl ; FindFile_LongNameYes
                                <1>
                                <1> loc_fff_longname_no:
                                <1>       ;mov   bx, [DirBuff_CurrentEntry]
0000804A 66891D[E45C0100]       <1>       mov   [FindFile_DirEntryNumber], bx
00008051 6689C2                 <1>       mov   dx, ax ; Ambigouos Filename chars used sign > 0
                                <1>
00008054 A0[C6520100]           <1>       mov   al, [Current_Drv]
00008059 A2[6A5C0100]           <1>       mov   [FindFile_Drv], al
                                <1>
0000805E A1[C0520100]           <1>       mov   eax, [Current_Dir_FCluster]
00008063 A3[DC5C0100]           <1>       mov   [FindFile_DirFirstCluster], eax
                                <1>
00008068 A1[F55A0100]           <1>       mov   eax, [DirBuff_Cluster]
0000806D A3[E05C0100]           <1>       mov   [FindFile_DirCluster], eax
                                <1>
00008072 66FF05[E65C0100]       <1>       inc   word [FindFile_MatchCounter]
                                <1>
00008079 89FB                   <1>       mov   ebx, edi
0000807B 89FE                   <1>       mov   esi, edi
0000807D BF[BC5C0100]           <1>       mov   edi, FindFile_DirEntry
00008082 89F8                   <1>       mov   eax, edi
00008084 B108                   <1>       mov   cl, 8
00008086 F3A5                   <1>       rep   movsd
00008088 89C6                   <1>       mov   esi, eax
0000808A 89DF                   <1>       mov   edi, ebx
                                <1>
0000808C A1[D85C0100]           <1>       mov   eax, [FindFile_DirEntry+28] ; File Size
                                <1>
00008091 8A1D[C75C0100]         <1>       mov   bl, [FindFile_DirEntry+11] ; File Attributes
00008097 8A3D[B95C0100]         <1>       mov   bh, [FindFile_LongNameYes]
                                <1>
                                <1>       ;mov   cx, [DirBuff_EntryCounter]
                                <1>       ;mov   [FindFile_DirEntryNumber], cx
                                <1>       ;mov   cx, [FindFile_DirEntryNumber]
                                <1>       ; ecx = 0
```

```
2450                                    <1>
2451                                    <1> loc_fff_retn:
2452 0000809D C3                       <1>       retn
2453                                    <1>
2454                                    <1> find_next_file:
2455                                    <1>       ; 15/10/2016
2456                                    <1>       ; 10/02/2016
2457                                    <1>       ; 08/02/2016 (TRDOS 386 =  TRDOS v2.0)
2458                                    <1>       ; 06/02/2011
2459                                    <1>       ; 17/09/2009
2460                                    <1>       ; 2005
2461                                    <1>       ; INPUT ->
2462                                    <1>       ;     NONE, Find First File Parameters
2463                                    <1>       ; OUTPUT ->
2464                                    <1>       ;     CF = 1 -> Error, Error Code in EAX (AL)
2465                                    <1>       ;     CF = 0 ->
2466                                    <1>       ;         ESI = Directory Entry (FindFile_DirEntry) Location
2467                                    <1>       ;         EDI = Directory Buffer Directory Entry Location
2468                                    <1>       ;         EAX = File Size
2469                                    <1>       ;           BL = Attributes of The File/Directory
2470                                    <1>       ;           BH = Long Name Yes/No Status (>0 is YES)
2471                                    <1>       ;             DX > 0 : Ambiguous filename chars are used
2472                                    <1>       ;
2473                                    <1>       ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)
2474                                    <1>
2475 0000809E 66833D[E65C0100]00       <1>       cmp    word [FindFile_MatchCounter], 0
2476 000080A6 7707                     <1>       ja     short loc_start_search_next_file
2477                                    <1>
2478                                    <1> loc_fnf_stc_retn:
2479 000080A8 F9                       <1>       stc
2480                                    <1> loc_fnf_ax12h_retn:
2481 000080A9 B80C000000               <1>       mov    eax, 12 ; No More files
2482                                    <1> ;loc_fnf_retn:
2483 000080AE C3                       <1>       retn
2484                                    <1>
2485                                    <1> loc_start_search_next_file:
2486 000080AF 668B1D[E45C0100]         <1>       mov    bx, [FindFile_DirEntryNumber]
2487 000080B6 6643                     <1>       inc    bx
2488 000080B8 663B1D[F35A0100]         <1>       cmp    bx, [DirBuff_LastEntry]
2489 000080BF 7719                     <1>       ja     short loc_cont_search_next_file
2490                                    <1>
2491                                    <1> loc_fnf_search:
2492 000080C1 BE[5B5C0100]             <1>       mov    esi, Dir_Entry_Name
2493 000080C6 66A1[BA5C0100]           <1>       mov    ax, [FindFile_AttributesMask]
2494 000080CC 6631C9                   <1>       xor    cx, cx
2495 000080CF E8741E0000               <1>       call   find_directory_entry
2496 000080D4 0F8355FFFFFF             <1>         jnc     loc_fff_fnf_ln_check
2497                                    <1>
2498                                    <1> loc_cont_search_next_file:
2499 000080DA 31DB                     <1>       xor    ebx, ebx
2500 000080DC 8A3D[C6520100]           <1>       mov    bh, [Current_Drv]
2501 000080E2 BE00010900               <1>       mov    esi, Logical_DOSDisks
2502 000080E7 01DE                     <1>       add    esi, ebx
2503                                    <1>
2504 000080E9 803D[C4520100]00         <1>       cmp    byte [Current_Dir_Level], 0
2505 000080F0 7608                     <1>       jna    short loc_fnf_check_FAT_type
2506 000080F2 807E0301                 <1>       cmp    byte [esi+LD_FATType], 1
2507 000080F6 72B1                     <1>       jb     short loc_fnf_ax12h_retn
2508 000080F8 EB06                     <1>       jmp    short loc_fnf_check_next_cluster
2509                                    <1>
2510                                    <1> loc_fnf_check_FAT_type:
2511 000080FA 807E0303                 <1>       cmp    byte [esi+LD_FATType], 3
2512 000080FE 72A9                     <1>       jb     short loc_fnf_ax12h_retn
2513                                    <1>
2514                                    <1> loc_fnf_check_next_cluster:
2515 00008100 A1[F55A0100]             <1>       mov    eax, [DirBuff_Cluster]
2516 00008105 E813380000               <1>       call   get_next_cluster
2517 0000810A 7306                     <1>       jnc    short loc_fnf_load_next_dir_cluster
2518 0000810C 09C0                     <1>       or     eax, eax
2519 0000810E 7498                     <1>       jz     short loc_fnf_stc_retn
2520                                    <1>       ;mov   eax, 17 ;Drive not ready or read error
2521 00008110 F5                       <1>       cmc    ;stc
2522                                    <1> loc_fnf_retn:
2523 00008111 C3                       <1>       retn
2524                                    <1>
2525                                    <1> loc_fnf_load_next_dir_cluster:
2526 00008112 E8EC390000               <1>       call   load_FAT_sub_directory
2527 00008117 72F8                     <1>       jc     short loc_fnf_retn
2528 00008119 6631DB                   <1>       xor    bx, bx
2529 0000811C 66891D[E45C0100]         <1>       mov    [FindFile_DirEntryNumber], bx
2530 00008123 EB9C                     <1>       jmp    short loc_fnf_search
2531                                    <1>
2532                                    <1> get_and_print_longname:
2533                                    <1>       ; 16/10/2016
2534                                    <1>       ; 13/02/2016 (TRDOS 386 =  TRDOS v2.0)
2535                                    <1>       ; 24/01/2010
2536                                    <1>       ; 17/10/2009 (CMD_INTR.ASM, 'cmp_cmd_longname')
2537                                    <1> get_longname_fchar:
2538 00008125 803E20                   <1>       cmp    byte [esi], 20h
2539 00008128 7701                     <1>       ja     short loc_find_longname
2540                                    <1>       ;jb    short loc_longname_retn
2541                                    <1>       ;inc   esi
2542                                    <1>       ;je    short get_longname_fchar
2543                                    <1> ;loc_longname_retn:
2544 0000812A C3                       <1>       retn
2545                                    <1> loc_find_longname:
2546 0000812B E87F210000               <1>       call   find_longname
2547 00008130 7328                     <1>       jnc    short loc_print_longname
2548                                    <1>
2549 00008132 08C0                     <1>       or     al, al
2550 00008134 741A                     <1>       jz     short loc_longname_not_found
2551                                    <1>
```

```
2552                                    <1>       ; 16/10/2016 (15h -> 15, 17)
2553 00008136 3C0F                      <1>       cmp    al, 15
2554 00008138 0F84C5F7FFFF              <1>       je     cd_drive_not_ready ; drive not ready
2555                                    <1>                               ; or
2556 0000813E 3C11                      <1>       cmp    al, 17           ; read error
2557 00008140 0F84BDF7FFFF              <1>       je     cd_drive_not_ready
2558                                    <1>
2559                                    <1> loc_ln_file_dir_not_found:
2560 00008146 BE[D8090100]              <1>       mov    esi, Msg_File_Directory_Not_Found
2561                                    <1>       ;call  print_msg
2562                                    <1>         ;retn
2563 0000814B E90DE2FFFF                <1>       jmp    print_msg
2564                                    <1>
2565                                    <1> loc_longname_not_found:
2566 00008150 BE[F7090100]              <1>         mov     esi, Msg_LongName_Not_Found
2567                                    <1>       ;call  print_msg
2568                                    <1>         ;retn
2569 00008155 E903E2FFFF                <1>       jmp    print_msg
2570                                    <1>
2571                                    <1> loc_print_longname:
2572                                    <1>       ;mov   esi, LongFileName
2573 0000815A BF[C6530100]              <1>       mov    edi, TextBuffer
2574 0000815F 57                        <1>       push   edi
2575 00008160 3C00                      <1>       cmp    al, 0
2576 00008162 7708                      <1>       ja     short loc_print_longname_1
2577                                    <1> loc_print_FS_longname: ; Singlix FS (64 byte ASCIIZ file name)
2578 00008164 AC                        <1>       lodsb
2579 00008165 AA                        <1>       stosb
2580 00008166 08C0                      <1>       or     al, al
2581 00008168 75FA                      <1>       jnz    short loc_print_FS_longname
2582 0000816A EB07                      <1>       jmp    short loc_print_longname_2
2583                                    <1>       ;
2584                                    <1> loc_print_longname_1: ; MS Windows long name (UNICODE chars)
2585 0000816C 66AD                      <1>       lodsw
2586 0000816E AA                        <1>       stosb
2587 0000816F 08C0                      <1>       or     al, al
2588 00008171 75F9                      <1>       jnz    short loc_print_longname_1
2589                                    <1>       ;
2590                                    <1> loc_print_longname_2:
2591 00008173 5E                        <1>       pop    esi
2592 00008174 E8E4E1FFFF                <1>       call   print_msg
2593 00008179 BE[4B130100]              <1>       mov    esi, nextline
2594                                    <1>       ;call  print_msg
2595                                    <1>       ;retn
2596 0000817E E9DAE1FFFF                <1>       jmp    print_msg
2597                                    <1>
2598                                    <1> show_file:
2599                                    <1>       ; 18/02/2016
2600                                    <1>       ; 17/02/2016
2601                                    <1>       ; 15/02/2016 (TRDOS 386 =  TRDOS v2.0)
2602                                    <1>       ; 13/09/2011 (CMD_INTR.ASM, 'cmp_cmd_show')
2603                                    <1>       ; 08/11/2009
2604                                    <1>
2605                                    <1> loc_show_parse_path_name:
2606 00008183 BF[6A5C0100]              <1>       mov    edi, FindFile_Drv
2607 00008188 E879200000                <1>       call   parse_path_name
2608 0000818D 0F825AF9FFFF              <1>       jc     loc_cmd_failed
2609                                    <1>
2610                                    <1> loc_show_check_filename_exists:
2611 00008193 BE[AC5C0100]              <1>       mov    esi, FindFile_Name
2612 00008198 803E20                    <1>       cmp    byte [esi], 20h
2613 0000819B 0F864CF9FFFF              <1>       jna    loc_cmd_failed
2614                                    <1>
2615                                    <1>       ; 15/02/2016 (invalid file name check)
2616 000081A1 E809020000                <1>       call   check_filename
2617 000081A6 730A                      <1>       jnc    short loc_show_change_drv
2618                                    <1>
2619 000081A8 BE[C40A0100]              <1>       mov    esi, Msg_invalid_name_chars
2620 000081AD E9ABE1FFFF                <1>       jmp    print_msg
2621                                    <1>
2622                                    <1> loc_show_change_drv:
2623 000081B2 8A35[C6520100]            <1>       mov    dh, [Current_Drv]
2624 000081B8 8835[265B0100]            <1>       mov    [RUN_CDRV], dh
2625 000081BE 8A15[6A5C0100]            <1>       mov    dl, [FindFile_Drv]
2626 000081C4 38F2                      <1>       cmp    dl, dh
2627 000081C6 740B                      <1>       je     short loc_show_change_directory
2628 000081C8 E890EAFFFF                <1>       call   change_current_drive
2629                                    <1>       ;jc    loc_file_rw_cmd_failed
2630 000081CD 0F8245F9FFFF              <1>       jc     loc_run_cmd_failed
2631                                    <1>
2632                                    <1> loc_show_change_directory:
2633 000081D3 803D[6B5C0100]20          <1>       cmp    byte [FindFile_Directory], 20h
2634 000081DA 7618                      <1>       jna    short loc_findload_showfile
2635                                    <1>
2636 000081DC FE05[AD060100]            <1>       inc    byte [Restore_CDIR]
2637 000081E2 BE[6B5C0100]              <1>       mov    esi, FindFile_Directory
2638 000081E7 30E4                      <1>       xor    ah, ah ; CD_COMMAND sign -> 0
2639 000081E9 E8041A0000                <1>       call   change_current_directory
2640                                    <1>       ;jc    loc_file_rw_cmd_failed
2641 000081EE 0F8224F9FFFF              <1>       jc     loc_run_cmd_failed
2642                                    <1>
2643                                    <1> ;loc_show_change_prompt_dir_string:
2644                                    <1>       ;call  change_prompt_dir_string
2645                                    <1>
2646                                    <1> loc_findload_showfile:
2647                                    <1>       ; 15/02/2016
2648 000081F4 BE[AC5C0100]              <1>       mov    esi, FindFile_Name
2649 000081F9 BF[5B5C0100]              <1>       mov    edi, Dir_Entry_Name ; Dir Entry Format File Name
2650 000081FE E8381F0000                <1>       call   convert_file_name
2651 00008203 89FE                      <1>       mov    esi, edi ; offset Dir_Entry_Name
2652                                    <1>
2653 00008205 28C0                      <1>       sub    al, al ; Attrib AND mask = 0
```

```
2654                                    <1>       ; Directory attribute : 10h
2655                                    <1>       ; Volume name attribute: 8h
2656 00008207 B418                     <1>       mov    ah, 00011000b ; 18h (Attrib NAND, AND --> zero mask)
2657                                    <1>       ;
2658 00008209 6631C9                   <1>       xor    cx, cx
2659 0000820C E8351C0000               <1>       call   locate_current_dir_file
2660                                    <1>       ;jc    loc_file_rw_cmd_failed
2661 00008211 0F8201F9FFFF             <1>       jc     loc_run_cmd_failed
2662                                    <1>
2663                                    <1> loc_show_load_file:
2664                                    <1>       ; EDI = Directory Entry
2665 00008217 668B4714                 <1>       mov    ax, [edi+DirEntry_FstClusHI] ; First Cluster High Word
2666 0000821B C1E010                   <1>       shl    eax, 16
2667 0000821E 668B471A                 <1>       mov    ax, [edi+DirEntry_FstClusLO] ; First Cluster Low Word
2668 00008222 A3[145D0100]             <1>       mov    [Show_Cluster], eax
2669 00008227 8B471C                   <1>       mov    eax, [edi+DirEntry_FileSize] ; File Size
2670 0000822A 21C0                     <1>       and    eax, eax ; Empty file !
2671 0000822C 0F8491000000             <1>         jz     end_of_show_file
2672 00008232 A3[185D0100]             <1>       mov    [Show_FileSize], eax
2673 00008237 31C0                     <1>       xor    eax, eax
2674 00008239 A3[1C5D0100]             <1>       mov    [Show_FilePointer], eax ; 0
2675 0000823E 66A3[205D0100]           <1>       mov    [Show_ClusterPointer], ax ; 0
2676 00008244 29DB                     <1>       sub    ebx, ebx
2677 00008246 8A3D[C6520100]           <1>       mov    bh, [Current_Drv]
2678 0000824C BE00010900               <1>       mov    esi, Logical_DOSDisks
2679 00008251 01DE                     <1>       add    esi, ebx
2680 00008253 8935[105D0100]           <1>       mov    [Show_LDDDT], esi ; Logical DOS Drv Description Table addr
2681                                    <1>
2682 00008259 807E0300                 <1>       cmp    byte [esi+LD_FATType], 0
2683 0000825D 7713                     <1>       ja     short loc_show_calculate_cluster_size
2684                                    <1>       ; Singlix FS
2685                                    <1>       ; First Cluster Number is FDT number (in compatibility buffer)
2686 0000825F 8B15[145D0100]           <1>       mov    edx, [Show_Cluster] ; Compatibility dir. buffer value (FDT)
2687 00008265 8915[0C5D0100]           <1>       mov    [Show_FDT], edx
2688 0000826B 31C0                     <1>       xor    eax, eax
2689 0000826D A3[145D0100]             <1>       mov    [Show_Cluster], eax ; Sector index  = 0
2690                                    <1>                                  ; (next time it will be 1)
2691                                    <1> loc_show_calculate_cluster_size:
2692 00008272 668B5E11                 <1>       mov    bx, [esi+LD_BPB+BPB_BytsPerSec] ; FAT 12-16-32 (512)
2693                                    <1>       ; BX = 512 = [esi+LD_FS_BytesPerSec] ; Singlix FS
2694 00008276 8A4613                   <1>       mov    al, [esi+LD_BPB+BPB_SecPerClust] ; FAT 12-16-32 (<= 128)
2695                                    <1>       ; AL = 1 = [esi+LD_FS_Reserved2] ; SectPerClust for Singlix FS
2696 00008279 F7E3                     <1>       mul    ebx
2697                                    <1>
2698                                    <1>       ;cmp   eax, 65536 ; non-compatible (very big) cluster size
2699                                    <1>       ;ja    short end_of_show_file
2700 0000827B 66A3[225D0100]           <1>       mov    [Show_ClusterSize], ax
2701                                    <1>
2702                                    <1> loc_start_show_file:
2703 00008281 BE[4B130100]             <1>       mov    esi, nextline
2704 00008286 E8D2E0FFFF               <1>       call   print_msg
2705                                    <1>
2706 0000828B A1[145D0100]             <1>       mov    eax, [Show_Cluster]
2707 00008290 C605[245D0100]17         <1>       mov    byte [Show_RowCount], 23
2708                                    <1>
2709                                    <1>       ; 17/02/2016
2710 00008297 8B35[105D0100]           <1>       mov    esi, [Show_LDDDT]
2711                                    <1>
2712                                    <1> loc_show_next_cluster:
2713                                    <1>       ; 15/02/2016
2714 0000829D BB00000700               <1>       mov    ebx, Cluster_Buffer ; 70000h (for current TRDOS 386 version)
2715                                    <1>       ; ESI = Logical DOS drv description table address
2716 000082A2 E89A380000               <1>       call   read_cluster
2717                                    <1>       ;jc    loc_file_rw_cmd_failed
2718 000082A7 0F826BF8FFFF             <1>       jc     loc_run_cmd_failed
2719                                    <1>
2720 000082AD 31DB                     <1>       xor    ebx, ebx
2721                                    <1> loc_show_next_byte:
2722 000082AF 803D[245D0100]00         <1>       cmp    byte [Show_RowCount], 0
2723 000082B6 7521                     <1>       jne    short pass_show_wait_for_key
2724 000082B8 30E4                     <1>       xor    ah, ah
2725 000082BA E85789FFFF               <1>       call   int16h
2726 000082BF 3C1B                     <1>       cmp    al, 1Bh
2727 000082C1 750F                     <1>       jne    short pass_exit_show
2728                                    <1> end_of_show_file:
2729                                    <1> pass_show_file:
2730 000082C3 BE[4B130100]             <1>       mov    esi, nextline
2731 000082C8 E890E0FFFF               <1>       call   print_msg
2732 000082CD E94D010000               <1>       jmp    loc_file_rw_restore_retn
2733                                    <1>
2734                                    <1> pass_exit_show:
2735 000082D2 C605[245D0100]14         <1>       mov    byte [Show_RowCount], 20
2736                                    <1> pass_show_wait_for_key:
2737 000082D9 81C300000700             <1>       add    ebx, Cluster_Buffer
2738 000082DF 8A03                     <1>       mov    al, [ebx]
2739 000082E1 3C0D                     <1>       cmp    al, 0Dh
2740 000082E3 0F8590000000             <1>         jne    loc_show_check_tab_space
2741 000082E9 FE0D[245D0100]           <1>       dec    byte [Show_RowCount]
2742                                    <1> pass_show_dec_rowcount:
2743 000082EF B307                     <1>       mov    bl, 7 ; (light gray character color, black background)
2744 000082F1 8A3D[2E520100]           <1>       mov    bh, [ACTIVE_PAGE] ; [ptty]
2745 000082F7 E8B699FFFF               <1>       call   _write_tty
2746                                    <1> loc_show_check_eof:
2747 000082FC FF05[1C5D0100]           <1>       inc    dword [Show_FilePointer]
2748 00008302 A1[1C5D0100]             <1>       mov    eax, [Show_FilePointer]
2749 00008307 3B05[185D0100]           <1>       cmp    eax, [Show_FileSize]
2750 0000830D 73B4                     <1>       jnb    short end_of_show_file
2751 0000830F 66FF05[205D0100]         <1>       inc    word [Show_ClusterPointer]
2752 00008316 0FB71D[205D0100]         <1>       movzx  ebx, word [Show_ClusterPointer]
2753                                    <1>
2754                                    <1>       ; 17/02/2016
2755                                    <1>       ; (sector boundary -9 bits- check, 512 = 0)
```

```
2756 0000831D 66F7C3FF01          <1>         test    bx, 1FFh ;  1 to 511
2757 00008322 758B                <1>       jnz    short loc_show_next_byte
2758                              <1>
2759                              <1>       ; 16/02/2016
2760 00008324 8B35[105D0100]      <1>       mov    esi, [Show_LDDDT]
2761                              <1>       ;
2762 0000832A 807E0300            <1>       cmp    byte [esi+LD_FATType], 0
2763 0000832E 7719                <1>       ja     short loc_show_check_fat_cluster_size
2764                              <1>
2765                              <1>       ; Singlix FS
2766                              <1>       ; 1 sector, more... (cluster size = 1 sector)
2767 00008330 A1[145D0100]        <1>       mov    eax, [Show_Cluster]
2768 00008335 40                  <1>       inc    eax
2769 00008336 A3[145D0100]        <1>       mov    [Show_Cluster], eax
2770                              <1>
2771 0000833B 6621DB              <1>       and    bx, bx ; 65536 -> 0
2772 0000833E 0F856BFFFFFF        <1>         jnz  loc_show_next_byte
2773 00008344 E954FFFFFF          <1>       jmp     loc_show_next_cluster
2774                              <1>
2775                              <1> loc_show_check_fat_cluster_size:
2776                              <1>       ; 17/02/2016
2777 00008349 663B1D[225D0100]    <1>       cmp    bx, [Show_ClusterSize] ; cluster size in bytes
2778 00008350 0F8259FFFFFF        <1>         jb   loc_show_next_byte
2779 00008356 66C705[205D0100]00- <1>       mov    word [Show_ClusterPointer], 0
2779 0000835E 00                  <1>
2780                              <1>
2781 0000835F A1[145D0100]        <1>       mov    eax, [Show_Cluster]
2782                              <1>       ;mov   esi, [Show_LDDDT]
2783                              <1> loc_show_get_next_cluster:
2784 00008364 E8B4350000          <1>       call   get_next_cluster
2785                              <1>       ;jc    loc_file_rw_cmd_failed
2786 00008369 0F82A9F7FFFF        <1>       jc     loc_run_cmd_failed
2787                              <1> loc_show_update_ccluster:
2788 0000836F A3[145D0100]        <1>       mov    [Show_Cluster], eax
2789 00008374 E924FFFFFF          <1>         jmp     loc_show_next_cluster
2790                              <1>
2791                              <1> loc_show_check_tab_space:
2792 00008379 3C09                <1>       cmp    al, 09h
2793 0000837B 0F856EFFFFFF        <1>         jne     pass_show_dec_rowcount
2794                              <1> loc_show_put_tab_space:
2795 00008381 8A3D[2E520100]      <1>       mov    bh, [ACTIVE_PAGE] ; [ptty]
2796 00008387 E8B595FFFF          <1>       call   get_cpos
2797                              <1>       ; dl = cursor column
2798 0000838C 80E207              <1>       and    dl, 7 ; 18/02/2016
2799                              <1>       ;shr   bh, 1 ; [ACTIVE_PAGE]
2800 0000838F 8A3D[2E520100]      <1>       mov    bh, [ACTIVE_PAGE]
2801 00008395 B307                <1>       mov    bl, 7 ; color attribute
2802                              <1> loc_show_put_space_chars:
2803 00008397 B020                <1>       mov    al, 20h ; space
2804                              <1>       ;mov   bh, [ACTIVE_PAGE] ; [ptty]
2805                              <1>       ;mov   bl, 7 ; color attribute
2806 00008399 6652                <1>       push   dx
2807 0000839B E81299FFFF          <1>       call   _write_tty
2808 000083A0 665A                <1>       pop    dx
2809                              <1>       ; 18/02/2016
2810 000083A2 80FA07              <1>       cmp    dl, 7
2811 000083A5 0F8351FFFFFF        <1>       jnb    loc_show_check_eof
2812 000083AB FEC2                <1>       inc    dl
2813 000083AD EBE8                <1>       jmp    short loc_show_put_space_chars
2814                              <1>
2815                              <1> check_filename:
2816                              <1>       ; 10/10/2016
2817                              <1>       ; 15/02/2016 (TRDOS 386 =  TRDOS v2.0)
2818                              <1>       ; 07/08/2010 (FILE.ASM, 'proc_check_filename')
2819                              <1>       ; 10/07/2010
2820                              <1>       ; Derived from 'proc_check_filename'
2821                              <1>       ; in the old TRDOS.ASM (09/02/2005).
2822                              <1>       ;
2823                              <1>       ; INPUT ->
2824                              <1>       ;     ESI = Dot File Name Location
2825                              <1>       ; OUTPUT ->
2826                              <1>       ;     cf = 1 -> error code in AL
2827                              <1>       ;          AL = ERR_INV_FILE_NAME (=26)
2828                              <1>       ;              Invalid file name chars
2829                              <1>       ;     cf = 0 -> valid file name
2830                              <1>       ;
2831                              <1>       ;(EAX, ECX, EDI will be changed)
2832                              <1>
2833                              <1> check_invalid_filename_chars:
2834                              <1>       ; 15/02/2016 (TRDOS 386 =  TRDOS v2.0)
2835                              <1>       ; 10/07/2010 (FILE.ASM, 'proc_check_invalid_filename_chars')
2836                              <1>       ; 10/02/2010
2837                              <1>       ; Derived from 'proc_check_invalid_filename_chars'
2838                              <1>       ; in the old TRDOS.ASM (09/02/2005).
2839                              <1>       ;
2840                              <1>       ; INPUT ->
2841                              <1>       ;     ESI = ASCIIZ FileName
2842                              <1>       ; OUTPUT ->
2843                              <1>       ;     cf = 1 -> invalid
2844                              <1>       ;     cf = 0 -> valid
2845                              <1>       ;
2846                              <1>       ;(EAX, ECX, EDI will be changed)
2847                              <1>
2848 000083AF 56                  <1>       push   esi
2849                              <1>
2850 000083B0 BF[AC070100]        <1>         mov     edi, invalid_fname_chars
2851 000083B5 AC                  <1>       lodsb
2852                              <1> check_filename_next_char:
2853 000083B6 B914000000          <1>       mov    ecx, sizeInvFnChars
2854 000083BB BF[AC070100]        <1>       mov    edi, invalid_fname_chars
2855                              <1> loc_scan_invalid_filename_char:
2856 000083C0 AE                  <1>       scasb
```

```
2857 000083C1 741F              <1>       je    short loc_invalid_filename_stc
2858 000083C3 E2FB              <1>       loop  loc_scan_invalid_filename_char
2859 000083C5 AC                <1>       lodsb
2860 000083C6 3C1F              <1>       cmp   al, 1Fh  ; 20h and above
2861 000083C8 77EC              <1>       ja    short check_filename_next_char
2862                            <1>
2863                            <1> check_filename_dot:
2864 000083CA 8B3424            <1>       mov   esi, [esp]
2865                            <1>
2866 000083CD B421              <1>       mov   ah, 21h
2867 000083CF B908000000        <1>       mov   ecx, 8
2868                            <1> loc_check_filename_next_char:
2869 000083D4 AC                <1>       lodsb
2870 000083D5 3C2E              <1>       cmp   al, 2Eh
2871 000083D7 7511              <1>       jne   short pass_check_fn_dot_check
2872                            <1> loc_check_filename_ext_0:
2873 000083D9 AC                <1>       lodsb
2874 000083DA 38E0              <1>       cmp   al, ah ; 21h
2875 000083DC 7205              <1>       jb    short loc_invalid_filename
2876 000083DE 3C2E              <1>       cmp   al, 2Eh
2877 000083E0 7519              <1>       jne   short loc_check_filename_ext_1
2878                            <1>
2879                            <1> loc_invalid_filename_stc:
2880                            <1> loc_check_fn_stc_rtn:
2881 000083E2 F9                <1>       stc
2882                            <1> loc_invalid_filename:
2883                            <1>       ; 10/10/2016 (0Bh -> 26)
2884 000083E3 B81A000000        <1>       mov   eax, ERR_INV_FILE_NAME ; (=26)
2885                            <1>       ; Invalid file name chars
2886                            <1> loc_check_fn_rtn:
2887 000083E8 5E                <1>       pop   esi
2888 000083E9 C3                <1>       retn
2889                            <1>
2890                            <1> pass_check_fn_dot_check:
2891 000083EA 38E0              <1>       cmp   al, ah ; 21h
2892 000083EC 7224              <1>       jb    short loc_check_fn_clc_rtn
2893 000083EE E2E4              <1>       loop  loc_check_filename_next_char
2894 000083F0 AC                <1>       lodsb
2895 000083F1 38E0              <1>       cmp   al, ah ; 21h
2896 000083F3 721D              <1>       jb    short loc_check_fn_clc_rtn
2897 000083F5 3C2E              <1>       cmp   al, 2Eh
2898 000083F7 75E9              <1>       jne   short loc_check_fn_stc_rtn
2899 000083F9 EBDE              <1>       jmp   short loc_check_filename_ext_0
2900                            <1>
2901                            <1> loc_check_filename_ext_1:
2902 000083FB AC                <1>       lodsb
2903 000083FC 38E0              <1>       cmp   al, ah ; 21h
2904 000083FE 7212              <1>       jb    short loc_check_fn_clc_rtn
2905 00008400 3C2E              <1>       cmp   al, 2Eh
2906 00008402 74DE              <1>       je    short loc_check_fn_stc_rtn
2907 00008404 AC                <1>       lodsb
2908 00008405 38E0              <1>       cmp   al, ah ; 21h
2909 00008407 7209              <1>       jb    short loc_check_fn_clc_rtn
2910 00008409 3C2E              <1>       cmp   al, 2Eh
2911 0000840B 74D5              <1>       je    short loc_check_fn_stc_rtn
2912 0000840D AC                <1>       lodsb
2913 0000840E 38E0              <1>       cmp   al, ah ; 21h
2914 00008410 73D0              <1>       jnb   short loc_check_fn_stc_rtn
2915                            <1>
2916                            <1> loc_check_fn_clc_rtn:
2917 00008412 5E                <1>       pop   esi
2918 00008413 F8                <1>       clc
2919 00008414 C3                <1>       retn
2920                            <1>
2921                            <1> loc_print_deleted_message:
2922 00008415 BE[990B0100]      <1>       mov   esi, Msg_Deleted
2923 0000841A E83EDFFFFF        <1>       call  print_msg
2924                            <1>
2925                            <1>       ;clc
2926                            <1>
2927                            <1> loc_file_rw_restore_retn:
2928                            <1>       ; 15/02/2016 (TRDOS 386 =  TRDOS v2.0)
2929                            <1>       ; 28/02/2010 (CMD_INTR.ASM)
2930                            <1> loc_file_rw_cmd_failed:
2931 0000841F 9C                <1>       pushf
2932 00008420 E84DF7FFFF        <1>       call  restore_cdir_after_cmd_fail
2933 00008425 9D                <1>       popf
2934 00008426 720D              <1>       jc    short loc_file_rw_check_write_fault
2935 00008428 C3                <1>       retn
2936                            <1>
2937                            <1> loc_permission_denied:
2938                            <1>       ; 27/02/2016
2939 00008429 BE[A60B0100]      <1>       mov   esi, Msg_Permission_Denied
2940 0000842E E82ADFFFFF        <1>       call  print_msg
2941 00008433 EBEA              <1>       jmp   short loc_file_rw_restore_retn
2942                            <1>
2943                            <1> loc_file_rw_check_write_fault:
2944                            <1>       ;cmp  al, 1Dh ; Write Fault
2945 00008435 3C12              <1>         cmp  al, 18 ; 05/11/2016
2946 00008437 0F85E0F6FFFF      <1>       jne    loc_run_cmd_failed_cmp_al
2947 0000843D BE[8D090100]      <1>       mov   esi, Msg_Not_Ready_Write_Err
2948                            <1>       ;call print_msg
2949                            <1>       ;retn
2950 00008442 E916DFFFFF        <1>       jmp   print_msg
2951                            <1>
2952                            <1> make_directory:
2953                            <1>       ; 21/02/2016 (TRDOS 386 =  TRDOS v2.0)
2954                            <1>       ; 12/03/2011 (CMD_INTR.ASM, 'cmp_cmd_mkdir')
2955                            <1>       ; 14/08/2010
2956                            <1>       ; 10/07/2010
2957                            <1>       ; 29/11/2009
2958                            <1>       ;
```

```
                            <1> get_mkdir_fchar:
                            <1>       ; esi = directory name
00008447 803E20             <1>       cmp    byte [esi], 20h
0000844A 7701               <1>         ja   short loc_mkdir_parse_path_name
                            <1>
                            <1> loc_mkdir_nodirname_retn:
0000844C C3                 <1>       retn
                            <1>
                            <1> loc_mkdir_parse_path_name:
0000844D BF[6A5C0100]       <1>       mov    edi, FindFile_Drv
00008452 E8AF1D0000         <1>         call    parse_path_name
00008457 0F8290F6FFFF       <1>       jc     loc_cmd_failed
                            <1>
                            <1> loc_mkdir_check_dirname_exists:
0000845D BE[AC5C0100]       <1>       mov    esi, FindFile_Name
00008462 803E20             <1>       cmp    byte [esi], 20h
00008465 0F8682F6FFFF       <1>       jna    loc_cmd_failed
0000846B 8935[285D0100]     <1>       mov    [DelFile_FNPointer], esi
00008471 E839FFFFFF         <1>       call   check_filename
00008476 7259               <1>       jc     short loc_mkdir_invalid_dir_name_chars
                            <1>
                            <1> loc_mkdir_drv:
00008478 8A35[C6520100]     <1>       mov    dh, [Current_Drv]
0000847E 8835[265B0100]     <1>       mov    [RUN_CDRV], dh
                            <1>
00008484 8A15[6A5C0100]     <1>       mov    dl, [FindFile_Drv]
0000848A 38F2               <1>       cmp    dl, dh
0000848C 7407               <1>       je     short loc_mkdir_change_directory
                            <1>
0000848E E8CAE7FFFF         <1>       call   change_current_drive
00008493 728A               <1>       jc     loc_file_rw_cmd_failed
                            <1>
                            <1> loc_mkdir_change_directory:
00008495 803D[6B5C0100]20   <1>       cmp    byte [FindFile_Directory], 20h
0000849C 7614               <1>       jna    short loc_mkdir_find_directory
                            <1>
0000849E FE05[AD060100]     <1>       inc    byte [Restore_CDIR]
000084A4 BE[6B5C0100]       <1>       mov    esi, FindFile_Directory
000084A9 30E4               <1>       xor    ah, ah ; CD_COMMAND sign -> 0
000084AB E842170000         <1>       call   change_current_directory
000084B0 722E               <1>       jc     short loc_mkdir_check_error_code
                            <1>
                            <1> ;loc_mkdir_change_prompt_dir_string:
                            <1>       ;call  change_prompt_dir_string
                            <1>
                            <1> loc_mkdir_find_directory:
                            <1>       ;mov   esi, FindFile_Name
000084B2 8B35[285D0100]     <1>       mov    esi, [DelFile_FNPointer]
                            <1>       ;xor   eax, eax
000084B8 6631C0             <1>       xor    ax, ax ; any name (dir, file, volume)
000084BB E82FFBFFFF         <1>       call   find_first_file
000084C0 721E               <1>       jc     short loc_mkdir_check_error_code
                            <1>
                            <1> loc_mkdir_directory_found:
000084C2 BE[F10A0100]       <1>       mov    esi, Msg_Name_Exists
000084C7 E891DEFFFF         <1>       call   print_msg
                            <1>
000084CC E94EFFFFFF         <1>         jmp     loc_file_rw_restore_retn
                            <1>
                            <1> loc_mkdir_invalid_dir_name_chars:
000084D1 BE[C40A0100]       <1>       mov    esi, Msg_invalid_name_chars
000084D6 E882DEFFFF         <1>       call   print_msg
                            <1>
000084DB E93FFFFFFF         <1>         jmp     loc_file_rw_restore_retn
                            <1>
                            <1> loc_mkdir_check_error_code:
000084E0 3C02               <1>       cmp    al, 2
                            <1>       ;je    short loc_mkdir_directory_not_found
000084E2 7406               <1>       je     short loc_mkdir_ask_for_yes_no
000084E4 F9                 <1>       stc
000084E5 E935FFFFFF         <1>         jmp     loc_file_rw_cmd_failed
                            <1>
                            <1> loc_mkdir_directory_not_found:
                            <1> loc_mkdir_ask_for_yes_no:
000084EA BE[120B0100]       <1>       mov    esi, Msg_DoYouWantMkdir
000084EF E869DEFFFF         <1>       call   print_msg
000084F4 8B35[285D0100]     <1>       mov    esi, [DelFile_FNPointer]
000084FA E85EDEFFFF         <1>       call   print_msg
000084FF BE[310B0100]       <1>       mov    esi, Msg_YesNo
00008504 E854DEFFFF         <1>       call   print_msg
                            <1>
00008509 C605[3B0B0100]20   <1>       mov    byte [Y_N_nextline], 20h
                            <1>
                            <1> loc_mkdir_ask_again:
00008510 30E4               <1>       xor    ah, ah
00008512 E8FF86FFFF         <1>       call   int16h
00008517 3C1B               <1>       cmp    al, 1Bh
                            <1>       ;je    short loc_do_not_make_directory
00008519 7447               <1>       je     short loc_mkdir_y_n_escape
0000851B 24DF               <1>       and    al, 0DFh ; y -> Y, n -> N
0000851D 3C59               <1>       cmp    al, 'Y' ; 'yes'
0000851F 7404               <1>       je     short loc_mkdir_yes_make_directory
00008521 3C4E               <1>       cmp    al, 'N' ; 'no'
00008523 75EB               <1>       jne    short loc_mkdir_ask_again
                            <1>
                            <1> loc_do_not_make_directory:
                            <1> loc_mkdir_yes_make_directory:
00008525 A2[3B0B0100]       <1>       mov    [Y_N_nextline], al
0000852A 6650               <1>       push   ax
0000852C BE[3B0B0100]       <1>       mov    esi, Y_N_nextline
00008531 E827DEFFFF         <1>       call   print_msg
00008536 6658               <1>       pop    ax
```

```
3061                                    <1>       ;cmp   al, 'Y' ; 'yes'
3062                                    <1>       ;cmc
3063                                    <1>         ;jnc loc_file_rw_restore_retn
3064 00008538 3C4E                     <1>       cmp    al, 'N' ; 'no'
3065 0000853A 0F84DFFEFFFF             <1>         je   loc_file_rw_restore_retn
3066                                    <1>
3067                                    <1> loc_mkdir_call_make_sub_directory:
3068 00008540 8B35[285D0100]           <1>       mov    esi, [DelFile_FNPointer]
3069 00008546 B110                     <1>       mov    cl, 10h ; Directory attributes
3070 00008548 E8B81D0000               <1>       call   make_sub_directory
3071                                    <1> loc_rename_file_ok: ; 06/03/2016
3072 0000854D 0F82CCFEFFFF             <1>         jc   loc_file_rw_cmd_failed
3073                                    <1> move_source_file_to_destination_OK:
3074 00008553 BE[3F0B0100]             <1>       mov    esi, Msg_OK
3075 00008558 E800DEFFFF               <1>       call   print_msg
3076 0000855D E9BDFEFFFF               <1>       jmp    loc_file_rw_restore_retn
3077                                    <1>
3078                                    <1> loc_mkdir_y_n_escape:
3079 00008562 B04E                     <1>       mov    al, 'N' ; 'no'
3080 00008564 EBBF                     <1>       jmp    short loc_do_not_make_directory
3081                                    <1>
3082                                    <1> delete_directory:
3083                                    <1>       ; 15/10/2016
3084                                    <1>       ; 06/03/2016
3085                                    <1>       ; 01/03/2016
3086                                    <1>       ; 29/02/2016
3087                                    <1>       ; 28/02/2016
3088                                    <1>       ; 27/02/2016
3089                                    <1>       ; 26/02/2016 (TRDOS 386 =  TRDOS v2.0)
3090                                    <1>       ; 16/10/2010 (CMD_INTR.ASM, 'cmp_cmd_rmdir')
3091                                    <1>       ; 05/06/2010
3092                                    <1>       ;
3093                                    <1> get_rmdir_fchar:
3094                                    <1>       ; esi = directory name
3095 00008566 803E20                   <1>       cmp    byte [esi], 20h
3096 00008569 7701                     <1>         ja   short loc_rmdir_parse_path_name
3097                                    <1>
3098                                    <1> loc_rmdir_nodirname_retn:
3099 0000856B C3                       <1>       retn
3100                                    <1>
3101                                    <1> loc_rmdir_parse_path_name:
3102 0000856C BF[6A5C0100]             <1>       mov    edi, FindFile_Drv
3103 00008571 E8901C0000               <1>       call   parse_path_name
3104 00008576 0F8271F5FFFF             <1>       jc     loc_cmd_failed
3105                                    <1>
3106                                    <1> loc_rmdir_check_dirname_exists:
3107 0000857C BE[AC5C0100]             <1>       mov    esi, FindFile_Name
3108 00008581 803E20                   <1>       cmp    byte [esi], 20h
3109 00008584 0F8663F5FFFF             <1>       jna    loc_cmd_failed
3110 0000858A 8935[285D0100]           <1>       mov    [DelFile_FNPointer], esi
3111                                    <1>
3112                                    <1> loc_rmdir_drv:
3113 00008590 8A35[C6520100]           <1>       mov    dh, [Current_Drv]
3114 00008596 8835[265B0100]           <1>       mov    [RUN_CDRV], dh
3115                                    <1>
3116 0000859C 8A15[6A5C0100]           <1>       mov    dl, [FindFile_Drv]
3117 000085A2 38F2                     <1>       cmp    dl, dh
3118 000085A4 740B                     <1>       je     short loc_rmdir_change_directory
3119                                    <1>
3120 000085A6 E8B2E6FFFF               <1>       call   change_current_drive
3121 000085AB 0F826EFEFFFF             <1>       jc     loc_file_rw_cmd_failed
3122                                    <1>
3123                                    <1> loc_rmdir_change_directory:
3124 000085B1 803D[6B5C0100]20         <1>       cmp    byte [FindFile_Directory], 20h
3125 000085B8 7614                     <1>       jna    short loc_rmdir_find_directory
3126                                    <1>
3127 000085BA FE05[AD060100]           <1>       inc    byte [Restore_CDIR]
3128 000085C0 BE[6B5C0100]             <1>       mov    esi, FindFile_Directory
3129 000085C5 30E4                     <1>       xor    ah, ah ; CD_COMMAND sign -> 0
3130 000085C7 E826160000               <1>       call   change_current_directory
3131 000085CC 7211                     <1>       jc     short loc_rmdir_check_error_code
3132                                    <1>
3133                                    <1> ;loc_rmdir_change_prompt_dir_string:
3134                                    <1>       ;call  change_prompt_dir_string
3135                                    <1>
3136                                    <1> loc_rmdir_find_directory:
3137                                    <1>       ;mov   esi, FindFile_Name
3138 000085CE 8B35[285D0100]           <1>       mov    esi, [DelFile_FNPointer]
3139 000085D4 66B81008                 <1>       mov    ax, 0810h ; Only directories
3140 000085D8 E812FAFFFF               <1>       call   find_first_file
3141 000085DD 730A                     <1>       jnc    short loc_rmdir_ambgfn_check
3142                                    <1>
3143                                    <1> loc_rmdir_check_error_code:
3144 000085DF 3C02                     <1>       cmp    al, 2
3145 000085E1 740B                     <1>       je     short loc_rmdir_directory_not_found
3146 000085E3 F9                       <1>       stc
3147 000085E4 E936FEFFFF               <1>       jmp    loc_file_rw_cmd_failed
3148                                    <1>
3149                                    <1> loc_rmdir_ambgfn_check:
3150 000085E9 6621D2                   <1>       and    dx, dx ; Ambiguous filename chars used sign (DX>0)
3151 000085EC 740F                     <1>       jz     short loc_rmdir_directory_found
3152                                    <1>
3153                                    <1> loc_rmdir_directory_not_found:
3154 000085EE BE[AF090100]             <1>       mov    esi, Msg_Dir_Not_Found
3155 000085F3 E865DDFFFF               <1>       call   print_msg
3156                                    <1>
3157 000085F8 E922FEFFFF               <1>       jmp    loc_file_rw_restore_retn
3158                                    <1>
3159                                    <1> loc_rmdir_directory_found:
3160 000085FD 80E307                   <1>       and    bl, 07h ; Attributes
3161 00008600 0F8523FEFFFF             <1>       jnz    loc_permission_denied
3162                                    <1>
```

```
                            <1> loc_rmdir_save_lnel: ; 28/02/2016
                            <1>        ;mov  bh, [LongName_EntryLength]
00008606 883D[325D0100]     <1>       mov    [DelFile_LNEL], bh ; Long name entry length (if > 0)
                            <1>       ; edi = Directory Entry Offset (DirBuff)
                            <1>       ; esi = Directory Entry (FFF Structure)
                            <1>       ;mov   [DelFile_DirEntryAddr], edi ; not required
                            <1>       ;mov   ax, [edi+20] ; First Cluster High Word
                            <1>         ;shl eax, 16
                            <1>       ;mov   ax, [edi+26] ; First Cluster Low Word
                            <1>       ; ROOT Dir First Cluster = 0
                            <1>         ;cmp eax, 2
                            <1>       ;jb    loc_update_direntry_1
                            <1>
                            <1> pass_rmdir_fc_check:
0000860C 57                 <1>       push   edi ; * (29/02/2016)
                            <1>
0000860D BE[450B0100]       <1>       mov    esi, Msg_DoYouWantRmDir
00008612 E846DDFFFF         <1>       call   print_msg
00008617 8B35[285D0100]     <1>       mov    esi, [DelFile_FNPointer]
0000861D E83BDDFFFF         <1>       call   print_msg
00008622 BE[310B0100]       <1>       mov    esi, Msg_YesNo
00008627 E831DDFFFF         <1>       call   print_msg
                            <1>
                            <1> loc_rmdir_ask_again:
0000862C 30E4               <1>       xor    ah, ah
0000862E E8E385FFFF         <1>       call   int16h
00008633 3C1B               <1>       cmp    al, 1Bh
                            <1>       ;je    short loc_do_not_delete_directory
00008635 0F8498000000       <1>         je      loc_rmdir_y_n_escape ; 06/03/2016
0000863B 24DF               <1>       and    al, 0DFh
0000863D A2[3B0B0100]       <1>       mov    [Y_N_nextline], al
00008642 3C59               <1>       cmp    al, 'Y'
00008644 7404               <1>       je     short loc_rmdir_yes_delete_directory
00008646 3C4E               <1>       cmp    al, 'N'
00008648 75E2               <1>       jne    short loc_rmdir_ask_again
                            <1>
                            <1> loc_do_not_delete_directory:
                            <1> loc_rmdir_yes_delete_directory:
0000864A A2[3B0B0100]       <1>       mov    [Y_N_nextline], al
0000864F 6650               <1>       push   ax
00008651 BE[3B0B0100]       <1>       mov    esi, Y_N_nextline
00008656 E802DDFFFF         <1>       call   print_msg
0000865B 6658               <1>       pop    ax
0000865D 5F                 <1>       pop    edi ; * (29/02/2016)
                            <1>       ;cmp   al, 'Y' ; 'yes'
                            <1>       ;cmc
                            <1>         ;jnc loc_file_rw_restore_retn
0000865E 3C4E               <1>       cmp    al, 'N' ; 'no'
00008660 0F84B9FDFFFF       <1>         je   loc_file_rw_restore_retn
                            <1>
                            <1> loc_rmdir_delete_short_name_check_dir_empty:
                            <1>       ; EDI = Directory buffer entry offset/address
00008666 668B4714           <1>       mov    ax, [edi+20] ; First Cluster High Word
0000866A C1E010             <1>         shl  eax, 16
0000866D 668B471A           <1>       mov    ax, [edi+26] ; First Cluster Low Word
                            <1>
00008671 A3[2C5D0100]       <1>       mov    [DelFile_FCluster], eax
                            <1>
                            <1>       ;mov   bx, [DirBuff_EntryCounter]
00008676 668B1D[E45C0100]   <1>       mov    bx, [FindFile_DirEntryNumber] ; 27/02/2016
0000867D 66891D[305D0100]   <1>       mov    [DelFile_EntryCounter], bx
                            <1>
00008684 29DB               <1>       sub    ebx, ebx
00008686 8A3D[6A5C0100]     <1>       mov    bh, [FindFile_Drv]
0000868C BE00010900         <1>       mov    esi, Logical_DOSDisks
00008691 01DE               <1>       add    esi, ebx
                            <1>
00008693 66817F0CA101       <1>       cmp    word [edi+DirEntry_NTRes], 01A1h
00008699 743F               <1>       je     short loc_rmdir_delete_fs_directory
                            <1>
                            <1>       ;cmp   byte [esi+LD_FATType], 1
                            <1>       ;jnb   short loc_rmdir_get__last_cluster_0
                            <1>       ;mov   eax, 0Bh ; Invalid Format
                            <1>       ;jmp   loc_file_rw_cmd_failed
                            <1>
                            <1> ;loc_rmdir_get_last_cluster_0:
0000869B 8B15[F55A0100]     <1>       mov    edx, [DirBuff_Cluster]
000086A1 8915[5C5D0100]     <1>       mov    [RmDir_ParentDirCluster], edx
                            <1>
000086A7 893D[585D0100]     <1>       mov    [RmDir_DirEntryOffset], edi
                            <1>
                            <1>       ; 01/03/2016
000086AD C705[E65A0100]0000- <1>      mov    dword [FAT_ClusterCounter], 0 ; Reset
000086B5 0000               <1>
                            <1>
                            <1> loc_rmdir_get_last_cluster:
000086B7 E85C3A0000         <1>       call   get_last_cluster
000086BC 0F82B8000000       <1>         jc      loc_rmdir_cmd_failed
                            <1>
000086C2 3B05[2C5D0100]     <1>       cmp    eax, [DelFile_FCluster]
000086C8 752F               <1>       jne    short loc_rmdir_multi_dir_clusters
                            <1>
000086CA C605[575D0100]00   <1>       mov    byte [RmDir_MultiClusters], 0
000086D1 EB2D               <1>       jmp    short pass_rmdir_multi_dir_clusters
                            <1>
                            <1> loc_rmdir_y_n_escape:
000086D3 B04E               <1>       mov    al, 'N' ; 'no'
000086D5 E970FFFFFF         <1>         jmp     loc_do_not_delete_directory
                            <1>
                            <1> loc_rmdir_delete_fs_directory:
000086DA 807E04A1           <1>       cmp    byte [esi+LD_FSType], 0A1h
000086DE 0F8545FDFFFF       <1>       jne    loc_permission_denied
```

```
3264                                    <1>
3265 000086E4 E826140000                <1>       call   delete_fs_directory
3266 000086E9 0F8326FDFFFF              <1>       jnc    loc_print_deleted_message
3267                                    <1>
3268 000086EF 09C0                      <1>       or     eax, eax
3269 000086F1 745D                      <1>       jz     loc_rmdir_directory_not_empty_2
3270 000086F3 F9                        <1>       stc
3271 000086F4 E926FDFFFF                <1>       jmp    loc_file_rw_cmd_failed
3272                                    <1>
3273                                    <1> loc_rmdir_multi_dir_clusters:
3274 000086F9 C605[575D0100]01          <1>       mov    byte [RmDir_MultiClusters], 1
3275                                    <1>
3276                                    <1> pass_rmdir_multi_dir_clusters:
3277 00008700 A3[605D0100]              <1>       mov    [RmDir_DirLastCluster], eax
3278 00008705 890D[645D0100]            <1>       mov    [RmDir_PreviousCluster], ecx
3279                                    <1>
3280                                    <1> loc_rmdir_load_fat_sub_directory:
3281 0000870B E8F3330000                <1>       call   load_FAT_sub_directory
3282 00008710 7268                      <1>       jc     loc_rmdir_cmd_failed
3283                                    <1>
3284                                    <1> loc_rmdir_find_last_dir_entry:
3285 00008712 56                        <1>       push   esi
3286 00008713 BE[4E5C0100]              <1>       mov    esi, Dir_File_Name
3287 00008718 C6062A                    <1>       mov    byte [esi], '*'
3288 0000871B C646082A                  <1>       mov    byte [esi+8], '*'
3289 0000871F 31DB                      <1>       xor    ebx, ebx ; Entry offset  = 0
3290                                    <1> loc_rmdir_find_last_dir_entry_next:
3291 00008721 66B80008                  <1>       mov    ax, 0800h ; Except volume/long names
3292 00008725 6631C9                    <1>       xor    cx, cx ; 0 = Find a valid file or dir name
3293 00008728 E81B180000                <1>       call   find_directory_entry
3294 0000872D 7271                      <1>       jc     short loc_rmdir_empty_dir_cluster
3295 0000872F 83FB01                    <1>       cmp    ebx, 1
3296 00008732 771B                      <1>       ja     short loc_rmdir_directory_not_empty_1
3297                                    <1> loc_rmdir_dot_entry_check:
3298 00008734 80FD2E                    <1>       cmp    ch, '.' ; The first char of the dir entry
3299 00008737 7516                      <1>       jne    short loc_rmdir_directory_not_empty_1
3300 00008739 08DB                      <1>       or     bl, bl
3301 0000873B 7506                      <1>       jnz    short loc_rmdir_dotdot_entry_check
3302 0000873D 807F0120                  <1>       cmp    byte [edi+1], 20h
3303 00008741 EB06                      <1>       jmp    short pass_rmdir_dot_entry_check
3304                                    <1>
3305                                    <1> loc_rmdir_dotdot_entry_check:
3306 00008743 66817F012E20              <1>       cmp    word [edi+1], '. '
3307                                    <1> pass_rmdir_dot_entry_check:
3308 00008749 7504                      <1>       jne    short loc_rmdir_directory_not_empty_1
3309 0000874B FEC3                      <1>       inc    bl
3310 0000874D EBD2                      <1>       jmp    short loc_rmdir_find_last_dir_entry_next
3311                                    <1>
3312                                    <1>
3313                                    <1> loc_rmdir_directory_not_empty_1:
3314 0000874F 58                        <1>       pop    eax ; pushed esi
3315                                    <1>
3316                                    <1> loc_rmdir_directory_not_empty_2:
3317 00008750 BE[660B0100]              <1>       mov    esi, Msg_Dir_Not_Empty
3318 00008755 E803DCFFFF                <1>       call   print_msg
3319                                    <1>       ; 01/03/2016
3320 0000875A A1[E65A0100]              <1>       mov    eax, [FAT_ClusterCounter]
3321 0000875F 09C0                      <1>       or     eax, eax ; 0 ?
3322 00008761 0F84B8FCFFFF              <1>       jz     loc_file_rw_restore_retn
3323                                    <1>       ; ESI = Logical DOS Drive Description Table address
3324                                    <1>
3325 00008767 66BB01FF                  <1>       mov    bx, 0FF01h ; BH = FFh -> use ESI for Drive parameters
3326                                    <1>                   ; BL = 1 -> add free clusters
3327 0000876B E829380000                <1>       call   calculate_fat_freespace
3328 00008770 09C9                      <1>       or     ecx, ecx
3329 00008772 0F84A7FCFFFF              <1>         jz      loc_file_rw_restore_retn ; ecx = 0 -> OK
3330                                    <1>       ; ecx > 0 -> Error (Recalculation is neeeded)
3331 00008778 EB0E                      <1>       jmp    short loc_rmdir_cmd_return
3332                                    <1>
3333                                    <1>
3334                                    <1> loc_rmdir_cmd_failed:
3335 0000877A 833D[E65A0100]01          <1>       cmp    dword [FAT_ClusterCounter], 1
3336 00008781 0F8298FCFFFF              <1>       jb     loc_file_rw_cmd_failed
3337 00008787 F9                        <1>       stc
3338                                    <1> loc_rmdir_cmd_return:
3339                                    <1>       ; 01/03/2016
3340 00008788 9C                        <1>       pushf
3341                                    <1>       ; ESI = Logical DOS Drive Description Table address
3342 00008789 66BB00FF                  <1>       mov    bx, 0FF00h ; BH = FFh -> use ESI for Drive parameters
3343                                    <1>                   ; BL = 0 -> Recalculate free cluster count
3344 0000878D 50                        <1>       push   eax
3345 0000878E E806380000                <1>       call   calculate_fat_freespace
3346 00008793 58                        <1>       pop    eax
3347 00008794 9D                        <1>       popf
3348 00008795 0F8284FCFFFF              <1>       jc     loc_file_rw_cmd_failed
3349 0000879B E97FFCFFFF                <1>       jmp    loc_file_rw_restore_retn
3350                                    <1>
3351                                    <1>
3352                                    <1> loc_rmdir_empty_dir_cluster:
3353 000087A0 5E                        <1>       pop    esi
3354                                    <1>
3355                                    <1> loc_rmdir_set_prev_cluster_dir_last_cluster:
3356 000087A1 803D[575D0100]00          <1>       cmp    byte [RmDir_MultiClusters], 0
3357 000087A8 761D                      <1>       jna    short loc_rmdir_unlink_dir_last_cluster
3358                                    <1>
3359 000087AA A1[645D0100]              <1>       mov    eax, [RmDir_PreviousCluster]
3360                                    <1>       ;xor   ecx, ecx
3361 000087AF 49                        <1>       dec    ecx ; FFFFFFFFh
3362 000087B0 E892340000                <1>       call   update_cluster
3363 000087B5 7310                      <1>       jnc    short loc_rmdir_unlink_dir_last_cluster
3364                                    <1>
3365                                    <1> loc_rmdir_unlink_stc_retn:
```

```
                                <1>       ; 01/03/2016
000087B7 83F801                 <1>       cmp    eax, 1  ; eax = 0 -> end of cluster chain
000087BA F5                     <1>       cmc
000087BB 72BD                   <1>       jc     short loc_rmdir_cmd_failed
000087BD EB1D                   <1>       jmp    short loc_rmdir_save_fat_buffer
                                <1>
                                <1> loc_rmdir_unlink_stc_retn_0Bh:
000087BF B81C000000             <1>       mov    eax, 28 ; Invalid format ; 15/10/2016
000087C4 F9                     <1>       stc
000087C5 EBB3                   <1>       jmp    short loc_rmdir_cmd_failed
                                <1>
                                <1> loc_rmdir_unlink_dir_last_cluster:
000087C7 A1[605D0100]           <1>       mov    eax, [RmDir_DirLastCluster]
000087CC 31C9                   <1>       xor    ecx, ecx ; 0
000087CE E874340000             <1>       call   update_cluster
000087D3 73EA                   <1>       jnc    short loc_rmdir_unlink_stc_retn_0Bh
                                <1>       ; Because of it is the last cluster
                                <1>       ; 'update_cluster' must return with eocc error
000087D5 09C0                   <1>       or     eax, eax
000087D7 7403                   <1>       jz     short loc_rmdir_save_fat_buffer ; eocc
000087D9 F9                     <1>       stc
000087DA EB9E                   <1>         jmp     short loc_rmdir_cmd_failed
                                <1>
                                <1> loc_rmdir_save_fat_buffer:
000087DC 803D[DE5A0100]02       <1>       cmp    byte [FAT_BuffValidData], 2
000087E3 7525                   <1>       jne    short loc_rmdir_calculate_FAT_freespace
000087E5 E81A370000             <1>       call   save_fat_buffer
000087EA 728E                   <1>       jc     short loc_rmdir_cmd_failed
                                <1>
                                <1>       ; 01/03/2016
000087EC 803D[575D0100]00       <1>       cmp    byte [RmDir_MultiClusters], 0
000087F3 7615                   <1>       jna    short loc_rmdir_calculate_FAT_freespace
                                <1>
000087F5 A1[2C5D0100]           <1>       mov    eax, [DelFile_FCluster]
000087FA E9B8FEFFFF             <1>         jmp     loc_rmdir_get_last_cluster
                                <1>
                                <1> loc_rmdir_delete_short_name_invalid_data:
000087FF B81D000000             <1>       mov    eax, 29 ; Invalid data (15/10/2016)
00008804 F9                     <1>       stc
00008805 E970FFFFFF             <1>         jmp     loc_rmdir_cmd_failed
                                <1>
                                <1> loc_rmdir_calculate_FAT_freespace:
0000880A A1[E65A0100]           <1>       mov    eax, [FAT_ClusterCounter]
0000880F 66BB01FF               <1>       mov    bx, 0FF01h
                                <1>       ; BL = 1 -> Add EAX to free space count
                                <1>       ; BH = FFh ->
                                <1>       ; ESI = Logical DOS Drive Description Table address
00008813 E881370000             <1>       call   calculate_fat_freespace
                                <1>
00008818 21C9                   <1>       and    ecx, ecx ; ecx = 0 -> valid free sector count
0000881A 7409                   <1>       jz     short loc_rmdir_delete_short_name_continue
                                <1>
                                <1> loc_rmdir_recalculate_FAT_freespace:
0000881C 66BB00FF               <1>         mov     bx, 0FF00h ; BL = 0 -> Recalculate free space
00008820 E874370000             <1>       call   calculate_fat_freespace
                                <1>
                                <1> loc_rmdir_delete_short_name_continue:
00008825 A1[5C5D0100]           <1>       mov    eax, [RmDir_ParentDirCluster]
0000882A 83F802                 <1>       cmp    eax, 2
0000882D 730D                   <1>       jnb    short loc_rmdir_del_short_name_load_sub_dir
0000882F E844320000             <1>       call   load_FAT_root_directory
00008834 0F82E5FBFFFF           <1>       jc     loc_file_rw_cmd_failed
0000883A EB0B                   <1>       jmp    short loc_rmdir_del_short_name_ld_chk_fclust
                                <1>
                                <1> loc_rmdir_del_short_name_load_sub_dir:
0000883C E8C2320000             <1>       call   load_FAT_sub_directory
00008841 0F82D8FBFFFF           <1>       jc     loc_file_rw_cmd_failed
                                <1>
                                <1> loc_rmdir_del_short_name_ld_chk_fclust:
00008847 0FB73D[585D0100]       <1>       movzx  edi, word [RmDir_DirEntryOffset]
0000884E 81C700000800           <1>       add    edi, Directory_Buffer
                                <1>
00008854 668B4714               <1>       mov    ax, [edi+20] ; First Cluster High Word
00008858 C1E010                 <1>       shl    eax, 16
0000885B 668B471A               <1>       mov    ax, [edi+26] ; First Cluster Low Word
                                <1>         ; Not necessary...
0000885F 3B05[2C5D0100]         <1>       cmp    eax, [DelFile_FCluster]
00008865 7598                   <1>       jne    short loc_rmdir_delete_short_name_invalid_data
                                <1>       ;
00008867 C607E5                 <1>       mov    byte [edi], 0E5h ; 'Deleted' sign
                                <1>       ; 27/02/2016
                                <1>       ; TRDOS v1 has a bug here! it does not set
                                <1>       ; 'DirBuff_ValidData' to 2; as result of this bug,
                                <1>       ; 'save_directory_buffer' would not save the change !
0000886A C605[F05A0100]02       <1>       mov    byte [DirBuff_ValidData], 2 ; change sign
                                <1>       ;
00008871 E8F41D0000             <1>       call   save_directory_buffer
00008876 0F82A3FBFFFF           <1>       jc     loc_file_rw_cmd_failed
                                <1>
                                <1> loc_rmdir_del_long_name:
0000887C 0FB615[325D0100]       <1>       movzx  edx, byte [DelFile_LNEL]
00008883 08D2                   <1>       or     dl, dl
00008885 7414                   <1>       jz     short loc_rmdir_update_parent_dir_lmdt
                                <1>
00008887 0FB705[305D0100]       <1>       movzx  eax, word [DelFile_EntryCounter]
0000888E 29D0                   <1>       sub    eax, edx
00008890 0F8289FBFFFF           <1>       jc     loc_file_rw_cmd_failed
                                <1>
                                <1>       ; EAX = Directory Entry Number of the long name last entry
00008896 E82F1F0000             <1>       call   delete_longname
                                <1>       ;jc    short loc_file_rw_cmd_failed
                                <1>
```

```
                                <1> loc_rmdir_update_parent_dir_lmdt:
0000889B E8651E0000             <1>       call   update_parent_dir_lmdt
                                <1>       ;jc    short loc_file_rw_cmd_failed
                                <1>
                                <1> loc_rmdir_ok:
000088A0 BE[3F0B0100]           <1>       mov    esi, Msg_OK
000088A5 E8B3DAFFFF             <1>       call   print_msg
000088AA E970FBFFFF             <1>       jmp    loc_file_rw_restore_retn
                                <1>
                                <1>
                                <1> delete_file:
                                <1>       ; 29/02/2016
                                <1>       ; 28/02/2016 (TRDOS 386 =  TRDOS v2.0)
                                <1>       ; 09/08/2010 (CMD_INTR.ASM, 'cmp_cmd_del')
                                <1>       ; 28/02/2010
                                <1>
                                <1> get_delfile_fchar:
                                <1>       ; esi = file name
000088AF 803E20                 <1>       cmp    byte [esi], 20h
000088B2 7701                   <1>         ja   short loc_delfile_parse_path_name
                                <1>
                                <1> loc_delfile_nofilename_retn:
000088B4 C3                     <1>       retn
                                <1>
                                <1> loc_delfile_parse_path_name:
000088B5 BF[6A5C0100]           <1>       mov    edi, FindFile_Drv
000088BA E847190000             <1>       call   parse_path_name
000088BF 0F8228F2FFFF           <1>       jc     loc_cmd_failed
                                <1>
                                <1> loc_delfile_check_filename_exists:
000088C5 BE[AC5C0100]           <1>       mov    esi, FindFile_Name
000088CA 803E20                 <1>       cmp    byte [esi], 20h
000088CD 0F861AF2FFFF           <1>       jna    loc_cmd_failed
000088D3 8935[285D0100]         <1>       mov    [DelFile_FNPointer], esi
                                <1>
                                <1> loc_delfile_drv:
000088D9 8A15[6A5C0100]         <1>       mov    dl, [FindFile_Drv]
000088DF 8A35[C6520100]         <1>       mov    dh, [Current_Drv]
000088E5 8835[265B0100]         <1>       mov    [RUN_CDRV], dh
000088EB 38F2                   <1>       cmp    dl, dh
000088ED 740B                   <1>       je     short loc_delfile_change_directory
                                <1>
000088EF E869E3FFFF             <1>       call   change_current_drive
000088F4 0F8225FBFFFF           <1>       jc     loc_file_rw_cmd_failed
                                <1>
                                <1> loc_delfile_change_directory:
000088FA 803D[6B5C0100]20       <1>       cmp    byte [FindFile_Directory], 20h
00008901 7618                   <1>       jna    short loc_delfile_find
                                <1>
00008903 FE05[AD060100]         <1>       inc    byte [Restore_CDIR]
00008909 BE[6B5C0100]           <1>       mov    esi, FindFile_Directory
0000890E 30E4                   <1>       xor    ah, ah ; CD_COMMAND sign -> 0
00008910 E8DD120000             <1>       call   change_current_directory
00008915 0F8204FBFFFF           <1>       jc     loc_file_rw_cmd_failed
                                <1>
                                <1> ;loc_delfile_change_prompt_dir_string:
                                <1>       ;call  change_prompt_dir_string
                                <1>
                                <1> loc_delfile_find:
                                <1>       ;mov   esi, FindFile_Name
0000891B 8B35[285D0100]         <1>       mov    esi, [DelFile_FNPointer]
00008921 66B80018               <1>       mov    ax, 1800h ; Except volume label and dirs
00008925 E8C5F6FFFF             <1>       call   find_first_file
0000892A 0F82EFFAFFFF           <1>       jc     loc_file_rw_cmd_failed
                                <1>
                                <1> loc_delfile_ambgfn_check:
00008930 6621D2                 <1>       and    dx, dx ; Ambiguous filename chars used sign (DX>0)
00008933 740B                   <1>       jz     short loc_delfile_found
                                <1>
                                <1> loc_file_not_found:
00008935 B802000000             <1>       mov    eax, 2 ; File not found sign
0000893A F9                     <1>       stc
0000893B E9DFFAFFFF             <1>       jmp    loc_file_rw_cmd_failed
                                <1>
                                <1> loc_delfile_found:
00008940 80E307                 <1>       and    bl, 07h ; Attributes
00008943 0F85E0FAFFFF           <1>         jnz     loc_permission_denied
                                <1>
                                <1> ;loc_delfile_found_save_lnel:
                                <1> ;     mov    [DelFile_LNEL], bh ; Long name entry length (if > 0)
                                <1>
                                <1> loc_delfile_ask_for_delete:
00008949 57                     <1>       push   edi ; * (29/02/2016)
                                <1>
0000894A BE[7D0B0100]           <1>       mov    esi, Msg_DoYouWantDelete
0000894F E809DAFFFF             <1>       call   print_msg
00008954 8B35[285D0100]         <1>       mov    esi, [DelFile_FNPointer]
0000895A E8FED9FFFF             <1>       call   print_msg
0000895F BE[310B0100]           <1>       mov    esi, Msg_YesNo
00008964 E8F4D9FFFF             <1>       call   print_msg
                                <1>
                                <1> loc_delfile_ask_again:
00008969 30E4                   <1>       xor    ah, ah
0000896B E8A682FFFF             <1>       call   int16h
00008970 3C1B                   <1>       cmp    al, 1Bh
                                <1>       ;je    short loc_do_not_delete_file
00008972 7457                   <1>       je     short loc_delfile_y_n_escape ; 06/03/2016
00008974 24DF                   <1>       and    al, 0DFh
00008976 A2[3B0B0100]           <1>       mov    [Y_N_nextline], al
0000897B 3C59                   <1>       cmp    al, 'Y'
0000897D 7404                   <1>       je     short loc_yes_delete_file
0000897F 3C4E                   <1>       cmp    al, 'N'
```

```
3570 00008981 75E6               <1>       jne    short loc_delfile_ask_again
3571                             <1>
3572                             <1> loc_do_not_delete_file:
3573                             <1> loc_yes_delete_file:
3574 00008983 A2[3B0B0100]       <1>       mov    [Y_N_nextline], al
3575 00008988 6650               <1>       push   ax
3576 0000898A BE[3B0B0100]       <1>       mov    esi, Y_N_nextline
3577 0000898F E8C9D9FFFF         <1>       call   print_msg
3578 00008994 6658               <1>       pop    ax
3579 00008996 5F                 <1>       pop    edi ; * (29/02/2016)
3580                             <1>       ;cmp   al, 'Y' ; 'yes'
3581                             <1>       ;cmc
3582                             <1>         ;jnc loc_file_rw_restore_retn
3583 00008997 3C4E               <1>       cmp    al, 'N' ; 'no'
3584 00008999 0F8480FAFFFF       <1>         je   loc_file_rw_restore_retn
3585                             <1>
3586                             <1> loc_delete_file:
3587 0000899F 8A3D[6A5C0100]     <1>       mov    bh, [FindFile_Drv]
3588                             <1>       ;mov   bl, [DelFile_LNEL]
3589 000089A5 8A1D[B95C0100]     <1>       mov    bl, [FindFile_LongNameEntryLength]
3590                             <1>       ;mov   cx, [DirBuff_EntryCounter]
3591 000089AB 668B0D[E45C0100]   <1>       mov    cx, [FindFile_DirEntryNumber]
3592                             <1>       ; (*) EDI = Directory buffer entry offset/address
3593 000089B2 E8FD1F0000         <1>       call   remove_file ; (FILE.ASM, 'proc_delete_file')
3594 000089B7 0F8358FAFFFF       <1>       jnc    loc_print_deleted_message
3595                             <1>
3596 000089BD 3C05               <1>       cmp    al, 05h
3597 000089BF 0F8464FAFFFF       <1>       je     loc_permission_denied
3598 000089C5 F9                 <1>       stc
3599 000089C6 E954FAFFFF         <1>       jmp    loc_file_rw_cmd_failed
3600                             <1>
3601                             <1> loc_delfile_y_n_escape:
3602 000089CB B04E               <1>       mov    al, 'N' ; 'no'
3603 000089CD EBB4               <1>       jmp    short loc_do_not_delete_file
3604                             <1>
3605                             <1> set_file_attributes:
3606                             <1>       ; 06/03/2016
3607                             <1>       ; 04/03/2016 (TRDOS 386 =  TRDOS v2.0)
3608                             <1>       ; 10/07/2010 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_attrib')
3609                             <1>       ; 23/05/2010
3610                             <1>       ; 17/12/2000 (P2000.ASM)
3611                             <1>
3612                             <1>       ; esi = file or directory name
3613 000089CF 6631C0             <1>       xor    ax, ax
3614 000089D2 66A3[CE0B0100]     <1>       mov    [Attr_Chars], ax
3615 000089D8 A2[805D0100]       <1>       mov    [Attributes], al
3616                             <1>
3617                             <1> get_attrib_fchar:
3618                             <1>       ; esi = file name
3619 000089DD 8A06               <1>       mov    al, [esi]
3620 000089DF 3C20               <1>       cmp    al, 20h
3621 000089E1 7623               <1>       jna    short loc_attr_file_nofilename_retn
3622                             <1>
3623                             <1> loc_scan_attrib_params:
3624 000089E3 3C2D               <1>       cmp    al, '-'
3625 000089E5 0F871C010000       <1>       ja     loc_attr_file_parse_path_name
3626 000089EB 7408               <1>       je     short loc_attr_space
3627                             <1>
3628 000089ED 3C2B               <1>       cmp    al, '+'
3629 000089EF 0F85F8F0FFFF       <1>       jne    loc_cmd_failed
3630                             <1>
3631                             <1> loc_attr_space:
3632 000089F5 8A6601             <1>       mov    ah, [esi+1]
3633 000089F8 80FC20             <1>       cmp    ah, 20h
3634 000089FB 770A               <1>       ja     short pass_attr_space
3635 000089FD 0F82EAF0FFFF       <1>       jb     loc_cmd_failed
3636 00008A03 46                 <1>       inc    esi
3637 00008A04 EBEF               <1>       jmp    short loc_attr_space
3638                             <1>
3639                             <1> loc_attr_file_nofilename_retn:
3640 00008A06 C3                 <1>       retn
3641                             <1>
3642                             <1> pass_attr_space:
3643 00008A07 80E4DF             <1>       and    ah, 0DFh
3644 00008A0A 80FC53             <1>       cmp    ah, 'S'
3645 00008A0D 0F87DAF0FFFF       <1>       ja     loc_cmd_failed
3646 00008A13 7204               <1>       jb     short pass_attr_system
3647 00008A15 B404               <1>       mov    ah, 04h   ; System
3648 00008A17 EB21               <1>       jmp    short pass_attr_archive
3649                             <1>
3650                             <1> pass_attr_system:
3651 00008A19 80FC48             <1>       cmp    ah, 'H'
3652 00008A1C 7706               <1>       ja     short pass_attr_hidden
3653 00008A1E 7213               <1>       jb     short pass_attr_read_only
3654 00008A20 B402               <1>       mov    ah, 02h    ; Hidden
3655 00008A22 EB16               <1>       jmp    short pass_attr_archive
3656                             <1>
3657                             <1> pass_attr_hidden:
3658 00008A24 80FC52             <1>       cmp    ah, 'R'
3659 00008A27 0F87C0F0FFFF       <1>       ja     loc_cmd_failed
3660 00008A2D 7204               <1>       jb     short pass_attr_read_only ; Read only
3661 00008A2F B401               <1>       mov    ah, 01h
3662 00008A31 EB07               <1>       jmp    short pass_attr_archive
3663                             <1>
3664                             <1> pass_attr_read_only:
3665 00008A33 80FC41             <1>       cmp    ah, 'A'
3666 00008A36 753B               <1>       jne    short loc_chk_attr_enter
3667 00008A38 B420               <1>       mov    ah, 20h    ; Archive
3668                             <1>
3669                             <1> pass_attr_archive:
3670 00008A3A 3C2D               <1>       cmp    al, '-'
3671 00008A3C 7508               <1>       jne    short pass_reducing_attributes
```

```
3672 00008A3E 0825[CE0B0100]      <1>       or    [Attr_Chars], ah
3673 00008A44 EB06                <1>       jmp   short loc_change_attributes_inc
3674                              <1>
3675                              <1> pass_reducing_attributes:
3676 00008A46 0825[CF0B0100]      <1>       or    [Attr_Chars+1], ah
3677                              <1>
3678                              <1> loc_change_attributes_inc:
3679 00008A4C 46                  <1>       inc   esi
3680 00008A4D 8A6601              <1>       mov   ah, [esi+1]
3681 00008A50 80FC20              <1>       cmp   ah, 20h
3682 00008A53 7227                <1>       jb    short pass_change_attr
3683 00008A55 74F5                <1>       je    short loc_change_attributes_inc
3684 00008A57 80FC2D              <1>       cmp   ah, '-'
3685 00008A5A 770D                <1>       ja    short loc_chk_next_attr_char1
3686 00008A5C 7405                <1>       je    short loc_chk_next_attr_char0
3687 00008A5E 80FC2B              <1>       cmp   ah, '+'
3688 00008A61 7506                <1>       jne   short loc_chk_next_attr_char1
3689                              <1>
3690                              <1> loc_chk_next_attr_char0:
3691 00008A63 46                  <1>       inc   esi
3692 00008A64 668B06              <1>       mov   ax, [esi]
3693 00008A67 EB9E                <1>       jmp   short pass_attr_space
3694                              <1>
3695                              <1> loc_chk_next_attr_char1:
3696 00008A69 803E2D              <1>       cmp   byte [esi], '-'
3697 00008A6C 7799                <1>       ja    short pass_attr_space
3698 00008A6E E988000000          <1>         jmp     loc_attr_file_check_fname_fchar
3699                              <1>
3700                              <1> loc_chk_attr_enter:
3701 00008A73 80FC0D              <1>       cmp   ah, 0Dh
3702 00008A76 0F8571F0FFFF        <1>       jne   loc_cmd_failed
3703                              <1>
3704                              <1> pass_change_attr:
3705 00008A7C A0[CE0B0100]        <1>       mov   al, [Attr_Chars]
3706 00008A81 F6D0                <1>       not   al
3707 00008A83 2005[805D0100]      <1>       and   [Attributes], al
3708 00008A89 A0[CF0B0100]        <1>       mov   al, [Attr_Chars+1]
3709 00008A8E 0805[805D0100]      <1>       or    [Attributes], al
3710                              <1>
3711                              <1> loc_show_attributes:
3712 00008A94 BE[4B130100]        <1>       mov   esi, nextline
3713 00008A99 E8BFD8FFFF          <1>       call  print_msg
3714                              <1>
3715                              <1> loc_show_attributes_no_nextline:
3716 00008A9E C705[CE0B0100]4E4F- <1>       mov   dword [Attr_Chars], 'NORM'
3716 00008AA6 524D                <1>
3717 00008AA8 66C705[D20B0100]41- <1>       mov   word [Attr_Chars+4], 'AL'
3717 00008AB0 4C                  <1>
3718 00008AB1 BE[CE0B0100]        <1>       mov   esi, Attr_Chars
3719 00008AB6 A0[805D0100]        <1>       mov   al, [Attributes]
3720 00008ABB A804                <1>       test  al, 04h
3721 00008ABD 7406                <1>       jz    short pass_put_attr_s
3722 00008ABF 66C7065300          <1>       mov   word [esi], 0053h     ; S
3723 00008AC4 46                  <1>       inc   esi
3724                              <1>
3725                              <1> pass_put_attr_s:
3726 00008AC5 A802                <1>       test  al, 02h
3727 00008AC7 7406                <1>       jz    short pass_put_attr_h
3728 00008AC9 66C7064800          <1>       mov   word [esi], 0048h     ; H
3729 00008ACE 46                  <1>       inc   esi
3730                              <1>
3731                              <1> pass_put_attr_h:
3732 00008ACF A801                <1>       test  al, 01h
3733 00008AD1 7406                <1>       jz    short pass_put_attr_r
3734 00008AD3 66C7065200          <1>       mov   word [esi], 0052h     ; R
3735 00008AD8 46                  <1>       inc   esi
3736                              <1>
3737                              <1> pass_put_attr_r:
3738 00008AD9 3C20                <1>       cmp   al, 20h
3739 00008ADB 7205                <1>       jb    short pass_put_attr_a
3740 00008ADD 66C7064100          <1>       mov   word [esi], 0041h     ; A
3741                              <1>
3742                              <1> pass_put_attr_a:
3743 00008AE2 BE[C10B0100]        <1>       mov   esi, Str_Attributes
3744 00008AE7 E871D8FFFF          <1>       call  print_msg
3745 00008AEC BE[4B130100]        <1>       mov   esi, nextline
3746 00008AF1 E867D8FFFF          <1>       call  print_msg
3747 00008AF6 E924F9FFFF          <1>       jmp   loc_file_rw_restore_retn
3748                              <1>
3749                              <1> loc_attr_file_check_fname_fchar:
3750 00008AFB 46                  <1>       inc   esi
3751 00008AFC 803E20              <1>       cmp   byte [esi], 20h
3752 00008AFF 74FA                <1>       je    short loc_attr_file_check_fname_fchar
3753 00008B01 0F8275FFFFFF        <1>         jb      pass_change_attr
3754                              <1>
3755                              <1> loc_attr_file_parse_path_name:
3756 00008B07 BF[6A5C0100]        <1>       mov   edi, FindFile_Drv
3757 00008B0C E8F5160000          <1>       call  parse_path_name
3758 00008B11 0F82D6EFFFFF        <1>       jc    loc_cmd_failed
3759                              <1>
3760                              <1> loc_attr_file_check_filename_exists:
3761 00008B17 BE[AC5C0100]        <1>       mov   esi, FindFile_Name
3762 00008B1C 803E20              <1>       cmp   byte [esi], 20h
3763 00008B1F 0F86C8EFFFFF        <1>       jna   loc_cmd_failed
3764 00008B25 8935[285D0100]      <1>       mov   [DelFile_FNPointer], esi
3765                              <1>
3766                              <1> loc_attr_file_drv:
3767 00008B2B 8A35[C6520100]      <1>       mov   dh, [Current_Drv]
3768 00008B31 8835[265B0100]      <1>       mov   [RUN_CDRV], dh
3769                              <1>
3770 00008B37 8A15[6A5C0100]      <1>       mov   dl, [FindFile_Drv]
3771 00008B3D 38F2                <1>       cmp   dl, dh
```

```
3772 00008B3F 740B                <1>       je     short loc_attr_file_change_directory
3773                              <1>
3774 00008B41 E817E1FFFF          <1>       call   change_current_drive
3775 00008B46 0F82D3F8FFFF        <1>       jc     loc_file_rw_cmd_failed
3776                              <1>
3777                              <1> loc_attr_file_change_directory:
3778 00008B4C 803D[6B5C0100]20    <1>         cmp     byte [FindFile_Directory], 20h
3779 00008B53 7618                <1>       jna    short loc_attr_file_find
3780                              <1>
3781 00008B55 FE05[AD060100]      <1>       inc    byte [Restore_CDIR]
3782                              <1>
3783 00008B5B BE[6B5C0100]        <1>       mov    esi, FindFile_Directory
3784 00008B60 30E4                <1>       xor    ah, ah ; CD_COMMAND sign -> 0
3785 00008B62 E88B100000          <1>       call   change_current_directory
3786 00008B67 0F82B2F8FFFF        <1>       jc     loc_file_rw_cmd_failed
3787                              <1>
3788                              <1> ;loc_attr_file_change_prompt_dir_string:
3789                              <1>       ;call  change_prompt_dir_string
3790                              <1>
3791                              <1> loc_attr_file_find:
3792                              <1>       ;mov   esi, FindFile_Name
3793 00008B6D 8B35[285D0100]      <1>       mov    esi, [DelFile_FNPointer]
3794 00008B73 66B80008            <1>       mov    ax, 0800h ; Except volume labels
3795 00008B77 E873F4FFFF          <1>       call   find_first_file
3796 00008B7C 0F829DF8FFFF        <1>       jc     loc_file_rw_cmd_failed
3797                              <1>
3798                              <1> loc_attr_file_ambgfn_check:
3799 00008B82 6609D2              <1>       or     dx, dx ; Ambiguous filename chars used sign (DX>0)
3800                              <1>       ;      (Note: It was BX in TRDOS v1)
3801                              <1>       ;jz    short loc_attr_file_found
3802 00008B85 0F85AAFDFFFF        <1>         jnz     loc_file_not_found ; 06/03/2016
3803                              <1>
3804                              <1>       ;mov   eax, 2 ; File not found sign
3805                              <1>       ;stc
3806                              <1>       ;jmp   loc_file_rw_cmd_failed
3807                              <1>
3808                              <1> loc_attr_file_found:
3809                              <1>       ; EDI = Directory buffer entry offset/address
3810                              <1>       ; BL = File (or Directory) Attributes
3811                              <1>       ;      (Note: It was 'CL' in TRDOS v1)
3812                              <1>       ; mov  bl, [EDI+0Bh]
3813                              <1>
3814 00008B8B 66833D[CE0B0100]00  <1>       cmp    word [Attr_Chars], 0
3815 00008B93 770B                <1>       ja     short loc_attr_file_change_attributes
3816 00008B95 881D[805D0100]      <1>       mov    [Attributes], bl
3817 00008B9B E9F4FEFFFF          <1>       jmp    loc_show_attributes
3818                              <1>
3819                              <1> loc_attr_file_change_attributes:
3820 00008BA0 A0[CE0B0100]        <1>       mov    al, [Attr_Chars]
3821 00008BA5 F6D0                <1>       not    al
3822 00008BA7 20C3                <1>       and    bl, al
3823 00008BA9 A0[CF0B0100]        <1>       mov    al, [Attr_Chars+1]
3824 00008BAE 08C3                <1>       or     bl, al
3825                              <1>
3826 00008BB0 66817F0CA101        <1>       cmp    word [edi+DirEntry_NTRes], 01A1h ; Singlix FS
3827 00008BB6 741D                <1>       je     short loc_attr_file_fs_check
3828                              <1>
3829 00008BB8 881D[805D0100]      <1>       mov    [Attributes], bl
3830 00008BBE 885F0B              <1>       mov    [edi+0Bh], bl    ; Attributes (New!)
3831                              <1>
3832                              <1>       ; 04/03/2016
3833                              <1>       ; TRDOS v1 has a bug here! it does not set
3834                              <1>       ; 'DirBuff_ValidData' to 2; as result of this bug,
3835                              <1>       ; 'save_directory_buffer' would not save the new attributes !
3836                              <1>
3837 00008BC1 C605[F05A0100]02    <1>       mov    byte [DirBuff_ValidData], 2
3838                              <1>
3839 00008BC8 E89D1A0000          <1>       call   save_directory_buffer
3840 00008BCD 0F824CF8FFFF        <1>       jc     loc_file_rw_cmd_failed
3841                              <1>
3842 00008BD3 EB33                <1>       jmp    short loc_print_attr_changed_message
3843                              <1>
3844                              <1> loc_attr_file_fs_check:
3845 00008BD5 29C0                <1>       sub    eax, eax
3846 00008BD7 8A25[EE5A0100]      <1>         mov     ah, [DirBuff_DRV]
3847 00008BDD BE00010900          <1>       mov    esi, Logical_DOSDisks
3848 00008BE2 01C6                <1>         add     esi, eax
3849 00008BE4 807E04A1            <1>         cmp     byte [esi+LD_FSType], 0A1h
3850 00008BE8 7309                <1>       jnc    short loc_attr_file_change_fs_file_attributes
3851 00008BEA 66B80D00            <1>       mov    ax, 0Dh ; Invalid Data
3852 00008BEE E92CF8FFFF          <1>       jmp    loc_file_rw_cmd_failed
3853                              <1>
3854                              <1> loc_attr_file_change_fs_file_attributes:
3855                              <1>       ; BL = New MS-DOS File Attributes
3856 00008BF3 88D8                <1>       mov    al, bl ; File/Directory Attributes
3857 00008BF5 30E4                <1>       xor    ah, ah ; Attributes in MS-DOS format sign
3858 00008BF7 E8A7050000          <1>       call   change_fs_file_attributes
3859 00008BFC 0F821DF8FFFF        <1>       jc     loc_file_rw_cmd_failed
3860                              <1>
3861 00008C02 881D[805D0100]      <1>       mov    [Attributes], bl
3862                              <1>
3863                              <1> loc_print_attr_changed_message:
3864 00008C08 BE[BC0B0100]        <1>       mov    esi, Msg_New
3865 00008C0D E84BD7FFFF          <1>       call   print_msg
3866 00008C12 E987FEFFFF          <1>       jmp    loc_show_attributes_no_nextline
3867                              <1>
3868                              <1> rename_file:
3869                              <1>       ; 06/11/2016
3870                              <1>       ; 05/11/2016
3871                              <1>       ; 16/10/2016
3872                              <1>       ; 08/03/2016
3873                              <1>       ; 06/03/2016 (TRDOS 386 =  TRDOS v2.0)
```

```
3874                                    <1>       ; 20/11/2010 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_rename')
3875                                    <1>       ; 16/11/2010
3876                                    <1>
3877                                    <1> get_rename_source_fchar:
3878                                    <1>       ; esi = file name
3879 00008C17 803E20                   <1>       cmp    byte [esi], 20h
3880 00008C1A 7614                     <1>         jna  short loc_rename_nofilename_retn
3881                                    <1>
3882 00008C1C 8935[A85D0100]           <1>       mov    [SourceFilePath], esi
3883                                    <1>
3884                                    <1> rename_scan_source_file:
3885 00008C22 46                       <1>       inc    esi
3886 00008C23 803E20                   <1>       cmp    byte [esi], 20h
3887 00008C26 7409                     <1>       je     short rename_scan_destination_file_1
3888                                    <1>       ;jb    short loc_rename_nofilename_retn
3889 00008C28 0F82BFEEFFFF             <1>       jb     loc_cmd_failed
3890 00008C2E EBF2                     <1>       jmp    short rename_scan_source_file
3891                                    <1>
3892                                    <1> loc_rename_nofilename_retn: ; 08/03/2016
3893 00008C30 C3                       <1>       retn
3894                                    <1>
3895                                    <1> rename_scan_destination_file_1:
3896 00008C31 C60600                   <1>       mov    byte [esi], 0
3897                                    <1>
3898                                    <1> rename_scan_destination_file_2:
3899 00008C34 46                       <1>       inc    esi
3900 00008C35 803E20                   <1>       cmp    byte [esi], 20h
3901 00008C38 74FA                     <1>       je     short rename_scan_destination_file_2
3902                                    <1>       ;jb    short loc_rename_nofilename_retn
3903 00008C3A 0F82ADEEFFFF             <1>       jb     loc_cmd_failed
3904                                    <1>
3905 00008C40 8935[AC5D0100]           <1>       mov    [DestinationFilePath], esi
3906                                    <1>
3907                                    <1> rename_scan_destination_file_3:
3908 00008C46 46                       <1>       inc    esi
3909 00008C47 803E20                   <1>       cmp    byte [esi], 20h
3910 00008C4A 77FA                     <1>       ja     short rename_scan_destination_file_3
3911                                    <1>
3912 00008C4C C60600                   <1>       mov    byte [esi], 0
3913                                    <1>
3914                                    <1> loc_rename_save_current_drive:
3915 00008C4F 8A35[C6520100]           <1>       mov    dh, [Current_Drv]
3916 00008C55 8835[265B0100]           <1>       mov    byte [RUN_CDRV], dh
3917                                    <1>
3918                                    <1> loc_rename_sf_parse_path_name:
3919 00008C5B 8B35[A85D0100]           <1>       mov    esi, [SourceFilePath]
3920 00008C61 BF[6A5C0100]             <1>       mov    edi, FindFile_Drv
3921 00008C66 E89B150000               <1>       call   parse_path_name
3922 00008C6B 0F827CEEFFFF             <1>       jc     loc_cmd_failed
3923                                    <1>
3924                                    <1> loc_rename_sf_check_filename_exists:
3925 00008C71 BE[AC5C0100]             <1>       mov    esi, FindFile_Name
3926 00008C76 803E20                   <1>       cmp    byte [esi], 20h
3927 00008C79 0F866EEEFFFF             <1>       jna    loc_cmd_failed
3928                                    <1>
3929                                    <1>       ;mov   [DelFile_FNPointer], esi
3930                                    <1>
3931                                    <1> loc_rename_sf_drv:
3932                                    <1>       ;mov   dh, [Current_Drv]
3933                                    <1>       ;mov   [RUN_CDRV], dh
3934                                    <1>
3935 00008C7F 8A15[6A5C0100]           <1>       mov    dl, [FindFile_Drv]
3936 00008C85 38F2                     <1>       cmp    dl, dh ; dh = [Current_Drv]
3937 00008C87 740B                     <1>       je     short rename_sf_change_directory
3938                                    <1>
3939 00008C89 E8CFDFFFFF               <1>       call   change_current_drive
3940 00008C8E 0F828BF7FFFF             <1>       jc     loc_file_rw_cmd_failed
3941                                    <1>
3942                                    <1> rename_sf_change_directory:
3943 00008C94 803D[6B5C0100]20         <1>       cmp    byte [FindFile_Directory], 20h
3944 00008C9B 7618                     <1>       jna    short rename_sf_find
3945                                    <1>
3946 00008C9D FE05[AD060100]           <1>       inc    byte [Restore_CDIR]
3947 00008CA3 BE[6B5C0100]             <1>       mov    esi, FindFile_Directory
3948 00008CA8 30E4                     <1>       xor    ah, ah ; CD_COMMAND sign -> 0
3949 00008CAA E8430F0000               <1>       call   change_current_directory
3950 00008CAF 0F826AF7FFFF             <1>       jc     loc_file_rw_cmd_failed
3951                                    <1>
3952                                    <1> ;rename_sf_change_prompt_dir_string:
3953                                    <1>       ;call  change_prompt_dir_string
3954                                    <1>
3955                                    <1> rename_sf_find:
3956                                    <1>       ;mov   esi, [DelFile_FNPointer]
3957 00008CB5 BE[AC5C0100]             <1>       mov    esi, FindFile_Name
3958                                    <1>
3959 00008CBA 66B80008                 <1>       mov    ax, 0800h ; Except volume labels
3960 00008CBE E82CF3FFFF               <1>       call   find_first_file
3961 00008CC3 0F8256F7FFFF             <1>       jc     loc_file_rw_cmd_failed
3962                                    <1>
3963                                    <1> loc_rename_sf_ambgfn_check:
3964 00008CC9 6621D2                   <1>       and    dx, dx ; Ambiguous filename chars used sign (DX>0)
3965                                    <1>       ;      (Note: It was BX in TRDOS v1)
3966                                    <1>       ;jz    short loc_rename_sf_found
3967 00008CCC 0F8563FCFFFF             <1>       jnz    loc_file_not_found
3968                                    <1>
3969                                    <1>       ;mov   eax, 2 ; File not found sign
3970                                    <1>       ;stc
3971                                    <1>       ;jmp   loc_file_rw_cmd_failed
3972                                    <1>
3973                                    <1> loc_rename_sf_found:
3974                                    <1>       ; EDI = Directory buffer entry offset/address
3975                                    <1>       ; BL = File (or Directory) Attributes
```

```
3976                                    <1>       ;     (Note: It was 'CL' in TRDOS v1)
3977                                    <1>       ; mov  bl, [EDI+0Bh]
3978                                    <1>
3979 00008CD2 F6C307                   <1>       test   bl, 07h ; Attributes, S-H-R
3980 00008CD5 0F854EF7FFFF             <1>       jnz    loc_permission_denied
3981                                    <1>
3982 00008CDB BE[6A5C0100]             <1>         mov     esi, FindFile_Drv
3983 00008CE0 BF[B05D0100]             <1>         mov     edi, SourceFile_Drv
3984 00008CE5 B920000000               <1>       mov    ecx, 32
3985 00008CEA F3A5                     <1>       rep    movsd
3986                                    <1>
3987                                    <1> loc_rename_df_parse_path_name:
3988 00008CEC 8B35[AC5D0100]           <1>       mov    esi, [DestinationFilePath]
3989 00008CF2 BF[6A5C0100]             <1>       mov    edi, FindFile_Drv
3990 00008CF7 E80A150000               <1>       call   parse_path_name
3991 00008CFC 7219                     <1>       jc     short loc_rename_df_cmd_failed
3992                                    <1>
3993                                    <1>       ;mov   dh, [RUN_CDRV]
3994 00008CFE 8A35[C6520100]           <1>       mov    dh, [Current_Drv]
3995                                    <1>
3996                                    <1>       ; 'rename' command is valid only for same dos drive and same dir!
3997                                    <1>       ; ('move' command must be used if source file and destination file
3998                                    <1>       ; directories are not same!)
3999 00008D04 8A15[6A5C0100]           <1>       mov    dl, [FindFile_Drv]
4000 00008D0A 38F2                     <1>       cmp    dl, dh ; are source and destination drives different ?!
4001 00008D0C 7509                     <1>       jne    short loc_rename_df_cmd_failed ; yes!
4002                                    <1>
4003                                    <1> rename_df_check_dirname_exists:
4004 00008D0E 803D[6B5C0100]00         <1>       cmp    byte [FindFile_Directory], 0
4005 00008D15 760B                     <1>       jna    short rename_df_check_filename_exists
4006                                    <1>
4007                                    <1>       ; different source file and destination file directories !
4008                                    <1> loc_rename_df_cmd_failed:
4009 00008D17 B801000000               <1>       mov    eax, 1 ; TRDOS 'Bad command or file name' error
4010 00008D1C F9                       <1>       stc
4011 00008D1D E9FDF6FFFF               <1>       jmp    loc_file_rw_cmd_failed
4012                                    <1>
4013                                    <1> rename_df_check_filename_exists:
4014 00008D22 BE[AC5C0100]             <1>       mov    esi, FindFile_Name
4015 00008D27 E883F6FFFF               <1>       call   check_filename
4016 00008D2C 0F829FF7FFFF             <1>       jc     loc_mkdir_invalid_dir_name_chars
4017                                    <1>
4018                                    <1>       ;mov   [DelFile_FNPointer], esi
4019                                    <1>       ;cmp   byte [esi], 20h
4020                                    <1>       ;ja    short loc_rename_df_find
4021                                    <1>
4022                                    <1>       ;mov   dh, [Current_Drv] ; dh has not been changed
4023                                    <1>
4024                                    <1> rename_df_drv_check_writable:
4025 00008D32 0FB6F6                   <1>       movzx  esi, dh
4026                                    <1>       ;movzx esi, byte [Current_Drv]
4027 00008D35 81C600010900             <1>       add    esi, Logical_DOSDisks
4028                                    <1>
4029 00008D3B 88F2                     <1>       mov    dl, dh ; dl = [Current_Drv]
4030 00008D3D 8A7601                   <1>       mov    dh, [esi+LD_DiskType]
4031                                    <1>
4032 00008D40 80FE01                   <1>       cmp    dh, 1 ; 0 = Invalid
4033 00008D43 7310                     <1>       jnb    short rename_df_compare_sf_df_name
4034                                    <1>
4035                                    <1>       ; 16/10/2016 (13h -> 30)
4036 00008D45 B81E000000               <1>       mov    eax, 30 ; 'Disk write-protected' error
4037 00008D4A 8B1D[AC5D0100]           <1>       mov    ebx, [DestinationFilePath]
4038 00008D50 E9CAF6FFFF               <1>       jmp    loc_file_rw_cmd_failed
4039                                    <1>
4040                                    <1> rename_df_compare_sf_df_name:
4041 00008D55 BE[AC5C0100]             <1>       mov    esi, FindFile_Name
4042 00008D5A BF[F25D0100]             <1>       mov    edi, SourceFile_Name
4043 00008D5F B90C000000               <1>       mov    ecx, 12
4044                                    <1> rename_df_compare_sf_df_name_next:
4045 00008D64 AC                       <1>       lodsb
4046 00008D65 AE                       <1>       scasb
4047 00008D66 7506                     <1>       jne    short loc_rename_df_find
4048 00008D68 08C0                     <1>       or     al, al
4049 00008D6A 74AB                     <1>       jz     short loc_rename_df_cmd_failed
4050 00008D6C E2F6                     <1>       loop   rename_df_compare_sf_df_name_next
4051                                    <1>
4052                                    <1> loc_rename_df_find:
4053                                    <1>       ;mov   esi, [DelFile_FNPointer]
4054 00008D6E BE[AC5C0100]             <1>       mov    esi, FindFile_Name
4055                                    <1>
4056 00008D73 6631C0                   <1>       xor    ax, ax ; Any
4057 00008D76 E874F2FFFF               <1>       call   find_first_file
4058 00008D7B 730A                     <1>       jnc    short loc_rename_df_found
4059                                    <1>
4060                                    <1> loc_rename_df_check_error_code:
4061                                    <1>       ;cmp   eax, 2
4062 00008D7D 3C02                     <1>       cmp    al, 2 ; Not found error
4063 00008D7F 7411                     <1>       je     short rename_df_move_find_struct_to_dest
4064 00008D81 F9                       <1>       stc
4065 00008D82 E998F6FFFF               <1>       jmp loc_file_rw_cmd_failed
4066                                    <1>
4067                                    <1> loc_rename_df_found:
4068                                    <1>       ; 05/11/2016
4069 00008D87 B805000000               <1>       mov    eax, 05h ; permission denied error
4070 00008D8C F9                       <1>       stc
4071 00008D8D E997F6FFFF               <1>       jmp    loc_permission_denied  ; 06/11/2016
4072                                    <1>
4073                                    <1> rename_df_move_find_struct_to_dest:
4074 00008D92 BE[6A5C0100]             <1>         mov     esi, FindFile_Drv
4075 00008D97 BF[305E0100]             <1>         mov     edi, DestinationFile_Drv
4076 00008D9C B920000000               <1>       mov    ecx, 32
4077 00008DA1 F3A5                     <1>       rep    movsd
```

```
4078                            <1>
4079                            <1> loc_rename_df_process_q_sf:
4080                            <1>       ;mov   ecx, 12
4081 00008DA3 B10C             <1>       mov    cl, 12
4082 00008DA5 BE[F25D0100]     <1>       mov    esi, SourceFile_Name
4083 00008DAA BF[FD0B0100]     <1>       mov    edi, Rename_OldName
4084                            <1> rename_df_process_q_nml_1_sf:
4085 00008DAF AC               <1>       lodsb
4086 00008DB0 3C20             <1>         cmp  al, 20h
4087 00008DB2 7603             <1>         jna  short rename_df_process_q_nml_2_sf
4088 00008DB4 AA               <1>       stosb
4089 00008DB5 E2F8             <1>       loop   rename_df_process_q_nml_1_sf
4090                            <1>
4091                            <1> rename_df_process_q_nml_2_sf:
4092 00008DB7 C60700           <1>       mov    byte [edi], 0
4093                            <1>
4094                            <1> loc_rename_df_process_q_df:
4095                            <1>       ;mov   ecx, 12
4096 00008DBA B10C             <1>       mov    cl, 12
4097 00008DBC BE[725E0100]     <1>       mov    esi, DestinationFile_Name
4098 00008DC1 BF[0E0C0100]     <1>       mov    edi, Rename_NewName
4099                            <1> rename_df_process_q_nml_1_df:
4100 00008DC6 AC               <1>       lodsb
4101 00008DC7 3C20             <1>       cmp    al, 20h
4102 00008DC9 7603             <1>       jna    short loc_rename_df_process_q_nml_2_df
4103 00008DCB AA               <1>       stosb
4104 00008DCC E2F8             <1>       loop   rename_df_process_q_nml_1_df
4105                            <1>
4106                            <1> loc_rename_df_process_q_nml_2_df:
4107 00008DCE C60700           <1>       mov    byte [edi], 0
4108                            <1>
4109                            <1> loc_rename_confirmation_question:
4110 00008DD1 BE[D50B0100]     <1>       mov    esi, Msg_DoYouWantRename
4111 00008DD6 E882D5FFFF       <1>       call   print_msg
4112                            <1>
4113 00008DDB A0[0D5E0100]     <1>       mov    al, [SourceFile_DirEntry+11] ; Attributes
4114 00008DE0 2410             <1>       and    al, 10h
4115 00008DE2 750C             <1>       jnz    short rename_confirmation_question_dir
4116                            <1>
4117                            <1> rename_confirmation_question_file:
4118 00008DE4 BE[EC0B0100]     <1>       mov    esi, Rename_File
4119 00008DE9 E86FD5FFFF       <1>       call   print_msg
4120 00008DEE EB0A             <1>       jmp    short rename_confirmation_question_as
4121                            <1>
4122                            <1> rename_confirmation_question_dir:
4123 00008DF0 BE[F20B0100]     <1>       mov    esi, Rename_Directory
4124 00008DF5 E863D5FFFF       <1>       call   print_msg
4125                            <1>
4126                            <1> rename_confirmation_question_as:
4127 00008DFA BE[FD0B0100]     <1>       mov    esi, Rename_OldName
4128 00008DFF E859D5FFFF       <1>       call   print_msg
4129 00008E04 BE[0A0C0100]     <1>       mov    esi, Msg_File_rename_as
4130 00008E09 E84FD5FFFF       <1>       call   print_msg
4131 00008E0E BE[310B0100]     <1>       mov    esi, Msg_YesNo
4132 00008E13 E845D5FFFF       <1>       call   print_msg
4133                            <1>
4134                            <1> loc_rename_ask_again:
4135 00008E18 30E4             <1>       xor    ah, ah
4136 00008E1A E8F77DFFFF       <1>       call   int16h
4137 00008E1F 3C1B             <1>       cmp    al, 1Bh
4138 00008E21 740F             <1>       je     short loc_do_not_rename_file
4139 00008E23 24DF             <1>       and    al, 0DFh
4140 00008E25 A2[3B0B0100]     <1>       mov    [Y_N_nextline], al
4141 00008E2A 3C59             <1>       cmp    al, 'Y'
4142 00008E2C 7404             <1>       je     short loc_yes_rename_file
4143 00008E2E 3C4E             <1>       cmp    al, 'N'
4144 00008E30 75E6             <1>       jne    short loc_rename_ask_again
4145                            <1>
4146                            <1> loc_do_not_rename_file:
4147                            <1> loc_yes_rename_file:
4148 00008E32 A2[3B0B0100]     <1>       mov    [Y_N_nextline], al
4149 00008E37 6650             <1>       push   ax
4150 00008E39 BE[3B0B0100]     <1>       mov    esi, Y_N_nextline
4151 00008E3E E81AD5FFFF       <1>       call   print_msg
4152 00008E43 6658             <1>       pop    ax
4153                            <1>       ;cmp   al, 'Y' ; 'yes'
4154                            <1>       ;cmc
4155                            <1>         ;jnc loc_file_rw_restore_retn
4156 00008E45 3C4E             <1>       cmp    al, 'N' ; 'no'
4157 00008E47 0F84D2F5FFFF     <1>         je   loc_file_rw_restore_retn
4158                            <1>
4159 00008E4D BE[0E0C0100]     <1>       mov    esi, Rename_NewName
4160 00008E52 668B0D[2A5E0100] <1>       mov    cx, [SourceFile_DirEntryNumber]
4161 00008E59 66A1[165E0100]   <1>       mov    ax, [SourceFile_DirEntry+20] ; First Cluster, HW
4162 00008E5F 66C1E010         <1>       shl    ax, 16
4163 00008E63 66A1[1C5E0100]   <1>       mov    ax, [SourceFile_DirEntry+26] ; First Cluster, LW
4164                            <1>
4165 00008E69 0FB61D[FF5D0100] <1>       movzx  ebx, byte [SourceFile_LongNameEntryLength]
4166 00008E70 E8DB1B0000       <1>       call   rename_directory_entry
4167 00008E75 E9D3F6FFFF       <1>       jmp    loc_rename_file_ok
4168                            <1> ;loc_rename_file_ok:
4169                            <1> ;     jc     loc_run_cmd_failed
4170                            <1> ;     mov    esi, Msg_OK
4171                            <1> ;     call   proc_printmsg
4172                            <1> ;     jmp    loc_file_rw_restore_retn
4173                            <1>
4174                            <1> move_file:
4175                            <1>       ; 11/03/2016
4176                            <1>       ; 09/03/2016
4177                            <1>       ; 08/03/2016 (TRDOS 386 =  TRDOS v2.0)
4178                            <1>       ; 21/05/2011 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_move')
4179                            <1>       ; 23/04/2011
```

```
4180                            <1>
4181                            <1> get_move_source_fchar:
4182                            <1>       ; esi = file name
4183 00008E7A 803E20            <1>       cmp    byte [esi], 20h
4184 00008E7D 7614              <1>         jna  short loc_move_nofilename_retn
4185                            <1>
4186 00008E7F 8935[A85D0100]    <1>       mov    [SourceFilePath], esi
4187                            <1>
4188                            <1> move_scan_source_file:
4189 00008E85 46                <1>       inc    esi
4190 00008E86 803E20            <1>       cmp    byte [esi], 20h
4191 00008E89 7409              <1>         je      short move_scan_destination_1
4192                            <1>       ;jb    short loc_move_nofilename_retn
4193 00008E8B 0F825CECFFFF      <1>       jb     loc_cmd_failed
4194 00008E91 EBF2              <1>       jmp    short move_scan_source_file
4195                            <1>
4196                            <1> loc_move_nofilename_retn:
4197 00008E93 C3                <1>       retn
4198                            <1>
4199                            <1> move_scan_destination_1:
4200 00008E94 C60600            <1>       mov    byte [esi], 0
4201                            <1>
4202                            <1> move_scan_destination_2:
4203 00008E97 46                <1>       inc    esi
4204 00008E98 803E20            <1>       cmp    byte [esi], 20h
4205 00008E9B 74FA              <1>       je     short move_scan_destination_2
4206                            <1>       ;jb    short loc_move_nofilename_retn
4207 00008E9D 0F824AECFFFF      <1>       jb     loc_cmd_failed
4208                            <1>
4209 00008EA3 8935[AC5D0100]    <1>       mov    [DestinationFilePath], esi
4210                            <1>
4211                            <1> move_scan_destination_3:
4212 00008EA9 46                <1>       inc    esi
4213 00008EAA 803E20            <1>       cmp    byte [esi], 20h
4214 00008EAD 77FA              <1>       ja     short move_scan_destination_3
4215 00008EAF C60600            <1>       mov    byte [esi], 0
4216                            <1>
4217                            <1> loc_move_scan_destination_OK:
4218 00008EB2 8B35[A85D0100]    <1>       mov    esi, [SourceFilePath]
4219 00008EB8 8B3D[AC5D0100]    <1>       mov    edi, [DestinationFilePath]
4220                            <1>
4221 00008EBE B001              <1>       mov    al, 1  ; move procedure Phase 1
4222 00008EC0 E8081C0000        <1>       call   move_source_file_to_destination_file
4223 00008EC5 7328              <1>       jnc    short move_source_file_to_destination_question
4224                            <1>
4225                            <1> loc_move_cmd_failed_1:
4226 00008EC7 08C0              <1>       or     al, al
4227 00008EC9 0F841EECFFFF      <1>       jz     loc_cmd_failed
4228 00008ECF 3C11              <1>       cmp    al, 11h
4229 00008ED1 740D              <1>       je     short loc_msg_not_same_device
4230 00008ED3 3C05              <1>       cmp    al, 05h
4231 00008ED5 0F853DECFFFF      <1>       jne    loc_run_cmd_failed
4232                            <1>
4233 00008EDB E949F5FFFF        <1>       jmp    loc_permission_denied
4234                            <1>
4235                            <1>       ;mov   esi, Msg_Permission_denied
4236                            <1>       ;call  print_msg
4237                            <1>       ;jmp   loc_file_rw_restore_retn
4238                            <1>
4239                            <1> loc_msg_not_same_device:
4240 00008EE0 BE[1B0C0100]      <1>       mov    esi, msg_not_same_drv
4241 00008EE5 E873D4FFFF        <1>       call   print_msg
4242 00008EEA E930F5FFFF        <1>       jmp    loc_file_rw_restore_retn
4243                            <1>
4244                            <1> move_source_file_to_destination_question:
4245 00008EEF A0[B05D0100]      <1>         mov     al, [SourceFile_Drv]
4246 00008EF4 0441              <1>       add    al, 'A'
4247 00008EF6 A2[7D0C0100]      <1>       mov    [msg_source_file_drv], al
4248 00008EFB A0[305E0100]      <1>         mov     al, [DestinationFile_Drv]
4249 00008F00 0441              <1>       add    al, 'A'
4250 00008F02 A2[9C0C0100]      <1>       mov    [msg_destination_file_drv], al
4251                            <1>
4252 00008F07 57                <1>       push   edi ; *
4253                            <1>
4254 00008F08 BE[610C0100]      <1>       mov    esi, msg_source_file
4255 00008F0D E84BD4FFFF        <1>       call   print_msg
4256 00008F12 BE[B15D0100]      <1>       mov    esi, SourceFile_Directory
4257 00008F17 803E20            <1>       cmp    byte [esi], 20h
4258 00008F1A 7605              <1>       jna    short msftdfq_sfn
4259 00008F1C E83CD4FFFF        <1>       call   print_msg
4260                            <1> msftdfq_sfn:
4261 00008F21 BE[F25D0100]      <1>       mov    esi, SourceFile_Name
4262 00008F26 E832D4FFFF        <1>       call   print_msg
4263 00008F2B BE[800C0100]      <1>       mov    esi, msg_destination_file
4264 00008F30 E828D4FFFF        <1>       call   print_msg
4265 00008F35 BE[315E0100]      <1>       mov    esi, DestinationFile_Directory
4266 00008F3A 803E20            <1>       cmp    byte [esi], 20h
4267 00008F3D 7605              <1>       jna    short msftdfq_dfn
4268 00008F3F E819D4FFFF        <1>       call   print_msg
4269                            <1> msftdfq_dfn:
4270 00008F44 BE[725E0100]      <1>       mov    esi, DestinationFile_Name
4271 00008F49 E80FD4FFFF        <1>       call   print_msg
4272 00008F4E BE[9F0C0100]      <1>       mov    esi, msg_copy_nextline
4273 00008F53 E805D4FFFF        <1>       call   print_msg
4274 00008F58 BE[9F0C0100]      <1>       mov    esi, msg_copy_nextline
4275 00008F5D E8FBD3FFFF        <1>       call   print_msg
4276                            <1>
4277                            <1> loc_move_ask_for_new_file_yes_no:
4278 00008F62 BE[2D0C0100]      <1>       mov    esi, Msg_DoYouWantMoveFile
4279 00008F67 E8F1D3FFFF        <1>       call   print_msg
4280 00008F6C BE[310B0100]      <1>       mov    esi, Msg_YesNo
4281 00008F71 E8E7D3FFFF        <1>       call   print_msg
```

```
4282                            <1> loc_move_ask_for_new_file_again:
4283 00008F76 30E4              <1>       xor    ah, ah
4284 00008F78 E8997CFFFF        <1>       call   int16h
4285 00008F7D 3C1B              <1>       cmp    al, 1Bh
4286                            <1>       ;je    short loc_do_not_move_file
4287 00008F7F 744F              <1>       je     short loc_move_y_n_escape
4288 00008F81 24DF              <1>       and    al, 0DFh
4289 00008F83 A2[3B0B0100]      <1>         mov     [Y_N_nextline], al
4290 00008F88 3C59              <1>       cmp    al, 'Y'
4291 00008F8A 7404              <1>       je     short loc_yes_move_file
4292 00008F8C 3C4E              <1>       cmp    al, 'N'
4293 00008F8E 75E6              <1>       jne    short loc_move_ask_for_new_file_again
4294                            <1>
4295                            <1> loc_do_not_move_file:
4296                            <1> loc_yes_move_file:
4297 00008F90 A2[3B0B0100]      <1>       mov    [Y_N_nextline], al
4298 00008F95 6650              <1>       push   ax
4299 00008F97 BE[3B0B0100]      <1>       mov    esi, Y_N_nextline
4300 00008F9C E8BCD3FFFF        <1>       call   print_msg
4301 00008FA1 6658              <1>       pop    ax
4302 00008FA3 5F                <1>       pop    edi ; *
4303                            <1>       ;cmp   al, 'Y' ; 'yes'
4304                            <1>       ;cmc
4305                            <1>         ;jnc loc_file_rw_restore_retn
4306 00008FA4 3C4E              <1>       cmp    al, 'N' ; 'no'
4307 00008FA6 0F8473F4FFFF      <1>         je   loc_file_rw_restore_retn
4308                            <1>
4309                            <1> loc_move_yes_move_file:
4310 00008FAC B002              <1>       mov    al, 2 ; move procedure Phase 2
4311 00008FAE E81A1B0000        <1>       call   move_source_file_to_destination_file
4312                            <1>       ;jc    short loc_move_cmd_failed_2
4313 00008FB3 0F839AF5FFFF      <1>         jnc     move_source_file_to_destination_OK
4314                            <1>
4315                            <1> ;move_source_file_to_destination_OK:
4316                            <1> ;     mov    esi, Msg_OK
4317                            <1> ;     call   print_msg
4318                            <1> ;     jmp    loc_file_rw_restore_retn
4319                            <1>
4320                            <1> loc_move_cmd_failed_2:
4321 00008FB9 3C27              <1>       cmp    al, 27h
4322 00008FBB 0F8557EBFFFF      <1>       jne    loc_run_cmd_failed
4323                            <1>
4324 00008FC1 BE[460C0100]      <1>       mov    esi, msg_insufficient_disk_space
4325 00008FC6 E892D3FFFF        <1>       call   print_msg
4326                            <1>
4327 00008FCB E94FF4FFFF        <1>       jmp    loc_file_rw_restore_retn
4328                            <1>
4329                            <1> loc_move_y_n_escape:
4330 00008FD0 B04E              <1>       mov    al, 'N' ; 'no'
4331 00008FD2 EBBC              <1>       jmp    short loc_do_not_move_file
4332                            <1>
4333                            <1> copy_file:
4334                            <1>       ; 15/10/2016
4335                            <1>       ; 24/03/2016
4336                            <1>       ; 21/03/2016
4337                            <1>       ; 15/03/2016 (TRDOS 386 =  TRDOS v2.0)
4338                            <1>       ; 21/05/2011 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_copy')
4339                            <1>       ; 01/08/2010
4340                            <1>
4341                            <1> get_copy_source_fchar:
4342                            <1>       ; esi = file name
4343 00008FD4 803E20            <1>       cmp    byte [esi], 20h
4344 00008FD7 7614              <1>         jna     short loc_copy_nofilename_retn
4345                            <1>
4346 00008FD9 8935[A85D0100]    <1>       mov    [SourceFilePath], esi
4347                            <1>
4348                            <1> copy_scan_source_file:
4349 00008FDF 46                <1>       inc    esi
4350 00008FE0 803E20            <1>       cmp    byte [esi], 20h
4351 00008FE3 7409              <1>       je     short copy_scan_destination_1
4352                            <1>       ;jb    short loc_copy_nofilename_retn
4353 00008FE5 0F8202EBFFFF      <1>       jb     loc_cmd_failed
4354 00008FEB EBF2              <1>       jmp    short copy_scan_source_file
4355                            <1>
4356                            <1> loc_copy_nofilename_retn:
4357 00008FED C3                <1>       retn
4358                            <1>
4359                            <1> copy_scan_destination_1:
4360 00008FEE C60600            <1>       mov    byte [esi], 0
4361                            <1>
4362                            <1> copy_scan_destination_2:
4363 00008FF1 46                <1>       inc    esi
4364 00008FF2 803E20            <1>       cmp    byte [esi], 20h
4365 00008FF5 74FA              <1>       je     short copy_scan_destination_2
4366                            <1>       ;jb    short loc_copy_nofilename_retn
4367 00008FF7 0F82F0EAFFFF      <1>       jb     loc_cmd_failed
4368                            <1>
4369 00008FFD 8935[AC5D0100]    <1>       mov    [DestinationFilePath], esi
4370                            <1>
4371                            <1> copy_scan_destination_3:
4372 00009003 46                <1>       inc    esi
4373 00009004 803E20            <1>       cmp    byte [esi], 20h
4374 00009007 77FA              <1>       ja     short copy_scan_destination_3
4375 00009009 C60600            <1>       mov    byte [esi], 0
4376                            <1>
4377                            <1> loc_copy_save_current_drive:
4378 0000900C 8A35[C6520100]    <1>       mov    dh, [Current_Drv]
4379 00009012 8835[265B0100]    <1>       mov    [RUN_CDRV], dh
4380                            <1>
4381                            <1> copy_source_file_to_destination_phase_1:
4382 00009018 8B35[A85D0100]    <1>       mov    esi, [SourceFilePath]
4383 0000901E 8B3D[AC5D0100]    <1>       mov    edi, [DestinationFilePath]
```

```
4384                                    <1>
4385 00009024 B001                      <1>       mov    al, 1  ; copy procedure Phase 1
4386 00009026 E83F1D0000                <1>       call   copy_source_file_to_destination_file
4387 0000902B 732B                      <1>       jnc    short copy_source_file_to_destination_question
4388                                    <1>
4389                                    <1> loc_copy_cmd_failed_1:
4390                                    <1>       ; 18/03/2016 (restore current drive and directory)
4391 0000902D 08C0                      <1>       or     al, al
4392 0000902F 7507                      <1>       jnz    short loc_copy_cmd_failed_2
4393                                    <1>
4394 00009031 FEC0                      <1>         inc     al ; mov al, 1 ; Bad command or file name !
4395 00009033 E9E0EAFFFF                <1>       jmp    loc_run_cmd_failed
4396                                    <1>
4397                                    <1> loc_copy_cmd_failed_2:
4398 00009038 3C27                      <1>       cmp    al, 27h ; Insufficient disk space
4399 0000903A 740D                      <1>       je     short loc_file_write_insuff_disk_space_msg
4400                                    <1>
4401 0000903C 3C05                      <1>       cmp    al, 05h
4402 0000903E 0F85D4EAFFFF              <1>       jne    loc_run_cmd_failed
4403                                    <1>
4404 00009044 E9E0F3FFFF                <1>       jmp    loc_permission_denied
4405                                    <1>
4406                                    <1> loc_file_write_insuff_disk_space_msg:
4407 00009049 BE[460C0100]              <1>       mov    esi, msg_insufficient_disk_space
4408 0000904E E80AD3FFFF                <1>       call   print_msg
4409 00009053 E9C7F3FFFF                <1>         jmp     loc_file_rw_restore_retn
4410                                    <1>
4411                                    <1> copy_source_file_to_destination_question:
4412 00009058 57                        <1>       push   edi ; *
4413                                    <1>
4414                                    <1>       ; dh = source file attributes
4415                                    <1>       ; dl > 0 -> destination file found
4416 00009059 20D2                      <1>       and    dl, dl
4417 0000905B 7449                      <1>       jz     short copy_source_file_to_destination_pass_owrq
4418                                    <1>
4419                                    <1> loc_copy_ask_for_owr_yes_no:
4420 0000905D BE[A20C0100]              <1>       mov    esi, Msg_DoYouWantOverWriteFile
4421 00009062 E8F6D2FFFF                <1>       call   print_msg
4422 00009067 BE[725E0100]              <1>       mov    esi, DestinationFile_Name
4423 0000906C E8ECD2FFFF                <1>       call   print_msg
4424 00009071 BE[310B0100]              <1>       mov    esi, Msg_YesNo
4425 00009076 E8E2D2FFFF                <1>       call   print_msg
4426                                    <1>
4427                                    <1> loc_copy_ask_for_owr_again:
4428 0000907B 30E4                      <1>       xor    ah, ah
4429 0000907D E8947BFFFF                <1>       call   int16h
4430 00009082 3C1B                      <1>       cmp    al, 1Bh
4431                                    <1>         ;je     loc_do_not_copy_file
4432 00009084 7419                      <1>         je      short loc_copy_y_n_escape
4433 00009086 24DF                      <1>       and    al, 0DFh
4434 00009088 A2[3B0B0100]              <1>         mov     [Y_N_nextline], al
4435 0000908D 3C59                      <1>       cmp    al, 'Y'
4436 0000908F 0F84B1000000              <1>         je      loc_yes_copy_file
4437 00009095 3C4E                      <1>       cmp    al, 'N'
4438 00009097 0F84A9000000              <1>         je      loc_do_not_copy_file
4439 0000909D EBDC                      <1>       jmp    short loc_copy_ask_for_owr_again
4440                                    <1>
4441                                    <1> loc_copy_y_n_escape:
4442 0000909F B04E                      <1>       mov    al, 'N' ; 'no'
4443 000090A1 E9A0000000                <1>         jmp     loc_do_not_copy_file
4444                                    <1>
4445                                    <1> copy_source_file_to_destination_pass_owrq:
4446 000090A6 A0[B05D0100]              <1>       mov     al, [SourceFile_Drv]
4447 000090AB 0441                      <1>       add    al, 'A'
4448 000090AD A2[7D0C0100]              <1>       mov    [msg_source_file_drv], al
4449 000090B2 A0[305E0100]              <1>         mov     al, [DestinationFile_Drv]
4450 000090B7 0441                      <1>       add    al, 'A'
4451 000090B9 A2[9C0C0100]              <1>       mov    [msg_destination_file_drv], al
4452                                    <1>
4453 000090BE BE[610C0100]              <1>       mov    esi, msg_source_file
4454 000090C3 E895D2FFFF                <1>       call   print_msg
4455 000090C8 BE[B15D0100]              <1>       mov    esi, SourceFile_Directory
4456 000090CD 803E20                    <1>       cmp    byte [esi], 20h
4457 000090D0 7605                      <1>       jna    short csftdfq_sfn
4458 000090D2 E886D2FFFF                <1>       call   print_msg
4459                                    <1> csftdfq_sfn:
4460 000090D7 BE[F25D0100]              <1>       mov    esi, SourceFile_Name
4461 000090DC E87CD2FFFF                <1>       call   print_msg
4462 000090E1 BE[800C0100]              <1>       mov    esi, msg_destination_file
4463 000090E6 E872D2FFFF                <1>       call   print_msg
4464 000090EB BE[315E0100]              <1>       mov    esi, DestinationFile_Directory
4465 000090F0 803E20                    <1>       cmp    byte [esi], 20h
4466 000090F3 7605                      <1>       jna    short csftdfq_dfn
4467 000090F5 E863D2FFFF                <1>       call   print_msg
4468                                    <1> csftdfq_dfn:
4469 000090FA BE[725E0100]              <1>       mov    esi, DestinationFile_Name
4470 000090FF E859D2FFFF                <1>       call   print_msg
4471 00009104 BE[9F0C0100]              <1>       mov    esi, msg_copy_nextline
4472 00009109 E84FD2FFFF                <1>       call   print_msg
4473 0000910E BE[9F0C0100]              <1>       mov    esi, msg_copy_nextline
4474 00009113 E845D2FFFF                <1>       call   print_msg
4475                                    <1>
4476                                    <1> loc_copy_ask_for_new_file_yes_no:
4477 00009118 BE[C10C0100]              <1>       mov    esi, Msg_DoYouWantCopyFile
4478 0000911D E83BD2FFFF                <1>       call   print_msg
4479 00009122 BE[310B0100]              <1>       mov    esi, Msg_YesNo
4480 00009127 E831D2FFFF                <1>       call   print_msg
4481                                    <1>
4482                                    <1> loc_copy_ask_for_new_file_again:
4483 0000912C 30E4                      <1>       xor    ah, ah
4484 0000912E E8E37AFFFF                <1>       call   int16h
4485 00009133 3C1B                      <1>       cmp    al, 1Bh
```

```
4486 00009135 740F              <1>       je    short loc_do_not_copy_file
4487 00009137 24DF              <1>       and   al, 0DFh
4488 00009139 A2[3B0B0100]      <1>         mov     [Y_N_nextline], al
4489 0000913E 3C59              <1>       cmp   al, 'Y'
4490 00009140 7404              <1>       je    short loc_yes_copy_file
4491 00009142 3C4E              <1>       cmp   al, 'N'
4492 00009144 75E6              <1>       jne   short loc_copy_ask_for_new_file_again
4493                            <1>
4494                            <1> loc_do_not_copy_file:
4495                            <1> loc_yes_copy_file:
4496 00009146 A2[3B0B0100]      <1>       mov   [Y_N_nextline], al
4497 0000914B 6650              <1>       push  ax
4498 0000914D BE[3B0B0100]      <1>       mov   esi, Y_N_nextline
4499 00009152 E806D2FFFF        <1>       call  print_msg
4500 00009157 6658              <1>       pop   ax
4501 00009159 5F                <1>       pop   edi ; *
4502                            <1>       ;cmp  al, 'Y' ; 'yes'
4503                            <1>       ;cmc
4504                            <1>         ;jnc loc_file_rw_restore_retn
4505 0000915A 3C4E              <1>       cmp   al, 'N' ; 'no'
4506 0000915C 0F84BDF2FFFF      <1>         je  loc_file_rw_restore_retn
4507                            <1>
4508                            <1> copy_source_file_to_destination_pass_q:
4509 00009162 B002              <1>       mov   al, 2  ; copy procedure Phase 2
4510 00009164 E8011C0000        <1>       call  copy_source_file_to_destination_file
4511                            <1>       ;jc   short loc_file_write_check_disk_space_err
4512                            <1>
4513                            <1>       ; 24/03/2016
4514 00009169 6651              <1>       push  cx
4515 0000916B BE[9F0C0100]      <1>       mov   esi, msg_copy_nextline
4516 00009170 E8E8D1FFFF        <1>       call  print_msg
4517                            <1>       ;pop  cx
4518 00009175 6658              <1>       pop   ax
4519                            <1>
4520                            <1>       ;or   cl, cl
4521 00009177 08C0              <1>       or    al, al
4522 00009179 7419              <1>       jz    short copy_source_file_to_destination_OK
4523                            <1>
4524                            <1>       ; 15/10/2016 (1Dh -> 18)
4525                            <1>       ; 18/03/2016 (1Dh)
4526                            <1>       ;cmp  cl, 18 ; write error
4527 0000917B 3C12              <1>       cmp   al, 18
4528 0000917D 7506              <1>       jne   short copy_source_file_to_destination_not_OK
4529                            <1>       ;
4530                            <1>       ;mov  al, cl ; error number (write fault!)
4531 0000917F F9                <1>       stc
4532 00009180 E99AF2FFFF        <1>       jmp   loc_file_rw_cmd_failed
4533                            <1>
4534                            <1> copy_source_file_to_destination_not_OK:
4535 00009185 BE[DA0C0100]      <1>       mov   esi, Msg_read_file_error_before_EOF
4536 0000918A E8CED1FFFF        <1>       call  print_msg
4537 0000918F E98BF2FFFF        <1>       jmp   loc_file_rw_restore_retn
4538                            <1>
4539                            <1> copy_source_file_to_destination_OK:
4540 00009194 BE[3F0B0100]      <1>       mov   esi, Msg_OK
4541 00009199 E8BFD1FFFF        <1>       call  print_msg
4542                            <1>
4543 0000919E E97CF2FFFF        <1>       jmp   loc_file_rw_restore_retn
4544                            <1>
4545                            <1> ;loc_file_write_check_disk_space_err:
4546                            <1>       ;cmp  al, 27h ; Insufficient disk space
4547                            <1>       ;je   loc_file_write_insuff_disk_space_msg
4548                            <1>         ;jb  loc_file_rw_cmd_failed
4549                            <1>
4550                            <1>       ;call  print_misc_error_msg ; 15/03/2016
4551                            <1>         ;jmp loc_file_rw_restore_retn
4552                            <1>
4553                            <1> change_fs_file_attributes:
4554                            <1>       ; 04/03/2016 ; Temporary
4555                            <1>       ; AL = File or directory attributes
4556                            <1>       ; AH = 0 -> Attributes are in MS-DOS format
4557                            <1>       ; AH > 0 -> Attributes are in SINGLIX format
4558                            <1>       ;push  ebx
4559                            <1>       ; ... do somethings here ...
4560                            <1>       ;pop   ebx
4561                            <1>       ; BL = File or directory attributes
4562 000091A3 C3                <1>       retn
4563                            <1>
4564                            <1> set_get_env:
4565                            <1>       ; 11/04/2016 (TRDOS 386 =  TRDOS v2.0)
4566                            <1>       ; 02/09/2011 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_set')
4567                            <1>       ; 2005 - 28/08/2011
4568                            <1> get_setenv_fchar:
4569                            <1>       ; esi = environment variable/string
4570 000091A4 8A06              <1>       mov   al, [esi]
4571 000091A6 3C20              <1>       cmp   al, 20h
4572 000091A8 771E              <1>       ja    short loc_find_env
4573                            <1>
4574 000091AA BE00300900        <1>       mov   esi, Env_Page
4575                            <1> loc_print_setline:
4576 000091AF 803E00            <1>       cmp   byte [esi], 0
4577 000091B2 7613              <1>       jna   short loc_setenv_retn
4578 000091B4 E8A4D1FFFF        <1>       call  print_msg
4579 000091B9 56                <1>       push  esi
4580 000091BA BE[4B130100]      <1>       mov   esi, nextline
4581 000091BF E899D1FFFF        <1>       call  print_msg
4582 000091C4 5E                <1>       pop   esi
4583 000091C5 EBE8              <1>       jmp   short loc_print_setline
4584                            <1>
4585                            <1> loc_setenv_retn:
4586 000091C7 C3                <1>       retn
4587                            <1>
```

```
                                <1> loc_find_env:
000091C8 3C3D                   <1>       cmp    al, '='
000091CA 0F841DE9FFFF           <1>       je     loc_cmd_failed
                                <1>
000091D0 56                     <1>       push   esi
                                <1> loc_repeat_env_equal_check:
000091D1 46                     <1>       inc    esi
000091D2 803E3D                 <1>       cmp    byte [esi], '='
000091D5 7431                   <1>       je     short pass_env_equal_check
000091D7 803E20                 <1>       cmp    byte [esi], 20h
000091DA 73F5                   <1>       jnb    short loc_repeat_env_equal_check
000091DC C60600                 <1>       mov    byte [esi], 0
000091DF 5E                     <1>       pop    esi
000091E0 BF[C6530100]           <1>       mov    edi, TextBuffer ; out buffer
000091E5 B9FF000000             <1>       mov    ecx, 255 ; maximum size (limit)
000091EA 30C0                   <1>       xor    al, al ; 0 -> use [ESI]
000091EC E89E000000             <1>       call   get_environment_string
000091F1 72D4                   <1>       jc     short loc_setenv_retn
                                <1>
000091F3 BE[C6530100]           <1>       mov    esi, TextBuffer
000091F8 E860D1FFFF             <1>       call   print_msg
000091FD BE[4B130100]           <1>       mov    esi, nextline
00009202 E856D1FFFF             <1>       call   print_msg
                                <1>
00009207 C3                     <1>       retn
                                <1>
                                <1> pass_env_equal_check:
00009208 46                     <1>       inc    esi
00009209 803E20                 <1>       cmp    byte [esi], 20h
0000920C 73FA                   <1>       jnb    short pass_env_equal_check
0000920E C60600                 <1>       mov    byte [esi], 0
                                <1>
                                <1> loc_call_set_env_string:
00009211 5E                     <1>       pop    esi
00009212 E83B010000             <1>       call   set_environment_string
00009217 73AE                   <1>       jnc    short loc_setenv_retn
                                <1>
                                <1> loc_set_cmd_failed:
00009219 3C08                   <1>       cmp    al, 08h
0000921B 0F85CCE8FFFF           <1>       jne    loc_cmd_failed
                                <1>
00009221 BE[1A0D0100]           <1>       mov    esi, Msg_No_Set_Space
00009226 E832D1FFFF             <1>       call   print_msg
                                <1>
0000922B C3                     <1>       retn
                                <1>
                                <1> set_get_path:
                                <1>       ; 11/04/2016 (TRDOS 386 =  TRDOS v2.0)
                                <1>       ; 03/09/2011 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_path')
                                <1>       ; 2005
                                <1> get_path_fchar:
                                <1>       ; esi = path
0000922C 803E20                 <1>       cmp    byte [esi], 20h
0000922F 7737                   <1>       ja     short loc_set_path
                                <1>
00009231 BE00300900             <1>       mov    esi, Env_Page
                                <1> loc_print_path:
00009236 803E00                 <1>       cmp    byte [esi], 0
00009239 762C                   <1>       jna    short loc_path_retn
                                <1>
0000923B BE[79070100]           <1>       mov    esi, Cmd_Path ; 'PATH' address
00009240 BF[C6530100]           <1>       mov    edi, TextBuffer ; oout buffer
00009245 30C0                   <1>       xor    al, al  ; use [ESI]
00009247 B9FF000000             <1>       mov    ecx, 255 ; maximum size (limit)
0000924C E83E000000             <1>       call   get_environment_string
00009251 7214                   <1>       jc     short loc_path_retn
                                <1>
00009253 BE[C6530100]           <1>       mov    esi, TextBuffer
00009258 E800D1FFFF             <1>       call   print_msg
0000925D BE[4B130100]           <1>       mov    esi, nextline
00009262 E8F6D0FFFF             <1>       call   print_msg
                                <1>
                                <1> loc_path_retn:
00009267 C3                     <1>       retn
                                <1>
                                <1> loc_set_path:
00009268 56                     <1>       push   esi
                                <1> loc_set_path_find_end:
00009269 46                     <1>       inc    esi
0000926A 803E20                 <1>       cmp    byte [esi], 20h
0000926D 73FA                   <1>       jnb    short loc_set_path_find_end
0000926F C60600                 <1>       mov    byte [esi], 0
                                <1> loc_set_path_header:
00009272 5E                     <1>       pop    esi
00009273 4E                     <1>       dec    esi
00009274 C6063D                 <1>       mov    byte [esi], '='
00009277 4E                     <1>       dec    esi
00009278 C60648                 <1>       mov    byte [esi], 'H'
0000927B 4E                     <1>       dec    esi
0000927C C60654                 <1>       mov    byte [esi], 'T'
0000927F 4E                     <1>       dec    esi
00009280 C60641                 <1>       mov    byte [esi], 'A'
00009283 4E                     <1>       dec    esi
00009284 C60650                 <1>       mov    byte [esi], 'P'
                                <1>
                                <1> loc_path_call_set_env_string:
00009287 E8C6000000             <1>       call   set_environment_string
0000928C 728B                   <1>         jc   short loc_set_cmd_failed
                                <1>
0000928E 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>
0000928F BA00300900             <1>       mov   edx, Env_Page
00009294 803A00                 <1>       cmp   byte [edx], 0
00009297 7474                   <1>       jz    short get_env_string_with_word_stc_retn
                                <1>
00009299 66890D[345F0100]       <1>       mov   [env_var_length], cx
                                <1>
000092A0 51                     <1>       push  ecx ; *
000092A1 56                     <1>       push  esi ; **
                                <1>
000092A2 08C0                   <1>       or    al, al
000092A4 7449                   <1>       jz    short get_env_string_with_word
                                <1>
                                <1> get_env_string_with_seq_number:
000092A6 B101                   <1>       mov   cl, 1
000092A8 88C5                   <1>       mov   ch, al
000092AA 31C0                   <1>       xor   eax, eax
000092AC 89D6                   <1>       mov   esi, edx ; Env_Page
                                <1>
                                <1> get_env_string_seq_number_check:
000092AE 38CD                   <1>       cmp   ch, cl
000092B0 7726                   <1>       ja    short get_env_string_seq_number_next
                                <1>
                                <1> get_env_string_move_to_buff:
000092B2 57                     <1>       push  edi ; ***
                                <1>
000092B3 29D2                   <1>       sub   edx, edx
                                <1>
                                <1> get_env_string_seq_number_repeat1:
000092B5 42                     <1>       inc   edx
000092B6 AC                     <1>       lodsb
000092B7 AA                     <1>       stosb
                                <1>
000092B8 66FF0D[345F0100]       <1>       dec   word [env_var_length]
000092BF 7508                   <1>       jnz   short get_env_string_seq_number_repeat3
                                <1>
                                <1> get_env_string_seq_number_repeat2:
000092C1 20C0                   <1>       and   al, al
000092C3 7408                   <1>       jz    short get_env_string_seq_number_ok
000092C5 42                     <1>       inc   edx
000092C6 AC                     <1>       lodsb
000092C7 EBF8                   <1>       jmp   short get_env_string_seq_number_repeat2
                                <1>
                                <1> get_env_string_seq_number_repeat3:
000092C9 08C0                   <1>       or    al, al
000092CB 75E8                   <1>       jnz   short get_env_string_seq_number_repeat1
                                <1>
                                <1> get_env_string_seq_number_ok:
000092CD 5F                     <1>       pop   edi ; ***
000092CE 89D0                   <1>       mov   eax, edx ; Length of the environment string
                                <1>                      ; (ASCIIZ, includes ZERO tail)
000092D0 BA00300900             <1>       mov   edx, Env_Page
                                <1>
                                <1> get_env_string_stc_retn:
000092D5 5E                     <1>       pop   esi ; **
000092D6 59                     <1>       pop   ecx ; *
000092D7 C3                     <1>       retn
                                <1>
                                <1> get_env_string_seq_number_next:
000092D8 AC                     <1>       lodsb
000092D9 08C0                   <1>       or    al, al
000092DB 75FB                   <1>       jnz   short get_env_string_seq_number_next
                                <1>
000092DD 81FE00320900           <1>       cmp   esi, Env_Page + Env_Page_Size ; +512 (+4096)
000092E3 F5                     <1>       cmc
000092E4 72EF                   <1>       jc    short get_env_string_stc_retn
                                <1>
000092E6 AC                     <1>       lodsb
000092E7 3C01                   <1>       cmp   al, 1
000092E9 72EA                   <1>       jb    short get_env_string_stc_retn
000092EB FEC1                   <1>       inc   cl
000092ED EBBF                   <1>       jmp   short get_env_string_seq_number_check
                                <1>
                                <1> get_env_string_with_word:
000092EF 31C9                   <1>       xor   ecx, ecx
                                <1>
                                <1> get_env_string_calc_word_length:
000092F1 AC                     <1>       lodsb
000092F2 3C20                   <1>       cmp   al, 20h
000092F4 7211                   <1>       jb    short get_env_string_calc_word_length_ok
                                <1>       ;inc  cx
000092F6 FEC1                   <1>       inc   cl
```

```
4792                                    <1>
4793 000092F8 3C61                     <1>       cmp   al, 'a'
4794 000092FA 72F5                     <1>       jb    short get_env_string_calc_word_length
4795 000092FC 3C7A                     <1>       cmp   al, 'z'
4796 000092FE 77F1                     <1>       ja    short get_env_string_calc_word_length
4797 00009300 24DF                     <1>       and   al, 0DFh
4798 00009302 8846FF                   <1>       mov   [esi-1], al
4799 00009305 EBEA                     <1>       jmp   short get_env_string_calc_word_length
4800                                    <1>
4801                                    <1> get_env_string_calc_word_length_ok:
4802 00009307 08C9                     <1>       or    cl, cl
4803 00009309 7506                     <1>       jnz   short get_env_string_calc_word_length_save
4804                                    <1>
4805 0000930B 5E                       <1>       pop   esi ; **
4806                                    <1>
4807                                    <1> get_env_string_stc_retn1:
4808 0000930C 59                       <1>       pop   ecx ; *
4809                                    <1>
4810                                    <1> get_env_string_with_word_stc_retn:
4811 0000930D 31C0                     <1>       xor   eax, eax
4812 0000930F F9                       <1>       stc
4813 00009310 C3                       <1>       retn
4814                                    <1>
4815                                    <1> get_env_string_calc_word_length_save:
4816 00009311 871C24                   <1>       xchg  ebx, [esp] ; **
4817 00009314 89DE                     <1>       mov   esi, ebx
4818                                    <1>             ; Start of the env string (to be searched)
4819                                    <1>
4820 00009316 57                       <1>       push  edi ; ***
4821 00009317 89D7                     <1>       mov   edi, edx ; Env_Page
4822                                    <1>
4823                                    <1> get_env_string_compare:
4824 00009319 57                       <1>       push  edi ; ****
4825 0000931A 51                       <1>       push  ecx ; ***** ; Variable name length
4826                                    <1>
4827                                    <1> get_env_string_compare_rep:
4828 0000931B AC                       <1>       lodsb
4829 0000931C AE                       <1>       scasb
4830 0000931D 7511                     <1>       jne   short get_env_string_compare_next1
4831 0000931F E2FA                     <1>       loop  get_env_string_compare_rep
4832                                    <1>
4833 00009321 803F3D                   <1>       cmp   byte [edi], '='
4834 00009324 750A                     <1>       jne   short get_env_string_compare_next1
4835                                    <1>
4836 00009326 59                       <1>       pop   ecx ; *****
4837 00009327 5F                       <1>       pop   edi ; ****
4838 00009328 89FE                     <1>       mov   esi, edi
4839 0000932A 5F                       <1>       pop   edi ; ***
4840 0000932B 871C24                   <1>       xchg  ebx, [esp] ; **
4841 0000932E EB82                     <1>       jmp   short get_env_string_move_to_buff
4842                                    <1>
4843                                    <1> get_env_string_compare_next1:
4844 00009330 89FE                     <1>       mov   esi, edi
4845 00009332 59                       <1>       pop   ecx ; *****
4846 00009333 5F                       <1>       pop   edi ; ****
4847                                    <1> get_env_string_compare_next2:
4848 00009334 81FEFF310900             <1>       cmp   esi, Env_Page + Env_Page_Size - 1 ; +511 (+4095)
4849 0000933A 7310                     <1>       jnb   short get_env_string_compare_not_ok
4850 0000933C 20C0                     <1>       and   al, al
4851 0000933E AC                       <1>       lodsb
4852 0000933F 75F3                     <1>       jnz   short get_env_string_compare_next2
4853 00009341 08C0                     <1>       or    al, al
4854 00009343 7407                     <1>       jz    short get_env_string_compare_not_ok
4855 00009345 4E                       <1>       dec   esi ; 12/04/2016
4856 00009346 89F7                     <1>       mov   edi, esi
4857 00009348 89DE                     <1>       mov   esi, ebx
4858 0000934A EBCD                     <1>       jmp   short get_env_string_compare
4859                                    <1>
4860                                    <1> get_env_string_compare_not_ok:
4861 0000934C 5F                       <1>       pop   edi ; ***
4862 0000934D 89DE                     <1>       mov   esi, ebx
4863 0000934F 5B                       <1>       pop   ebx ; **
4864 00009350 EBBA                     <1>       jmp   short get_env_string_stc_retn1
4865                                    <1>
4866                                    <1> set_environment_string:
4867                                    <1>       ; 13/04/2016
4868                                    <1>       ; 12/04/2016
4869                                    <1>       ; 11/04/2016
4870                                    <1>       ; 06/04/2016
4871                                    <1>       ; 05/04/2016 (TRDOS 386 = TRDOS v2.0)
4872                                    <1>       ; 02/09/2011 (TRDOS v1, MAINPROG.ASM)
4873                                    <1>       ; 29/08/2011
4874                                    <1>       ; 29/08/2011
4875                                    <1>       ; INPUT->
4876                                    <1>       ;     ESI = ASCIIZ environment string
4877                                    <1>       ; OUTPUT ->
4878                                    <1>       ;     ESI is not changed
4879                                    <1>       ;     CF = 1 -> Could not set,
4880                                    <1>       ;          insufficient environment space
4881                                    <1>       ;
4882                                    <1>       ; (EAX, EDX will be changed)
4883                                    <1>       ;
4884                                    <1>       ;    (EAX = Start address of the env string if > 0)
4885                                    <1>       ;    (EDX = Environment string length)
4886                                    <1>
4887 00009352 56                       <1>       push  esi ; *
4888                                    <1>
4889 00009353 31C0                     <1>       xor   eax, eax
4890                                    <1>
4891                                    <1> set_env_chk_validation1:
4892 00009355 FEC4                     <1>       inc   ah ; variable (string) length
4893 00009357 AC                       <1>       lodsb
```

```
4894 00009358 3C3D                <1>       cmp    al, '='
4895 0000935A 7415                <1>       je     short set_env_chk_validation2
4896 0000935C 3C20                <1>       cmp    al, 20h
4897 0000935E 720F                <1>       jb     short set_env_string_stc
4898                              <1>
4899                              <1>       ; 06/04/2016
4900 00009360 3C61                <1>       cmp    al, 'a'
4901 00009362 72F1                <1>       jb     short set_env_chk_validation1
4902 00009364 3C7A                <1>       cmp    al, 'z'
4903 00009366 77ED                <1>       ja     short set_env_chk_validation1
4904 00009368 2C20                <1>       sub    al, 'a'-'A'
4905 0000936A 8846FF              <1>       mov    [esi-1], al
4906 0000936D EBE6                <1>       jmp    short set_env_chk_validation1
4907                              <1>
4908                              <1> set_env_string_stc:
4909 0000936F 5E                  <1>       pop    esi ; *
4910                              <1>       ;stc
4911 00009370 C3                  <1>       retn
4912                              <1>
4913                              <1> set_env_chk_validation2:
4914 00009371 51                  <1>       push   ecx ; **
4915 00009372 53                  <1>       push   ebx ; ***
4916 00009373 57                  <1>       push   edi ; ****
4917                              <1>
4918                              <1>       ; 12/04/2016
4919 00009374 8B5C240C            <1>       mov    ebx, [esp+12]
4920                              <1>
4921                              <1> set_env_chk_validation2w:
4922 00009378 89F7                <1>       mov    edi, esi
4923 0000937A 4F                  <1>       dec    edi
4924                              <1>
4925 0000937B 807FFF20            <1>       cmp    byte [edi-1], 20h
4926 0000937F 771A                <1>       ja     short set_env_chk_validation2z
4927                              <1>
4928 00009381 56                  <1>       push   esi
4929 00009382 89FE                <1>       mov    esi, edi
4930 00009384 4E                  <1>       dec    esi
4931                              <1>
4932                              <1> set_env_chk_validation2x:
4933 00009385 4E                  <1>       dec    esi
4934                              <1>
4935 00009386 39DE                <1>       cmp    esi, ebx
4936 00009388 7207                <1>       jb     short set_env_chk_validation2y
4937                              <1>
4938 0000938A 4F                  <1>       dec    edi
4939                              <1>
4940 0000938B 8A06                <1>       mov    al, [esi]
4941 0000938D 8807                <1>       mov    [edi], al
4942                              <1>
4943 0000938F EBF4                <1>       jmp    short set_env_chk_validation2x
4944                              <1>
4945                              <1> set_env_chk_validation2y:
4946 00009391 5E                  <1>       pop    esi
4947                              <1>
4948                              <1>       ;mov   byte [ebx], 20h
4949                              <1>
4950 00009392 43                  <1>       inc    ebx
4951 00009393 895C240C            <1>       mov    [esp+12], ebx
4952                              <1>
4953 00009397 FECC                <1>       dec    ah ; 13/04/2016
4954                              <1>
4955 00009399 EBDD                <1>       jmp    short set_env_chk_validation2w
4956                              <1>
4957                              <1> set_env_chk_validation2z:
4958 0000939B BA00300900          <1>       mov    edx, Env_Page
4959 000093A0 89D7                <1>       mov    edi, edx
4960                              <1>
4961                              <1> set_env_chk_validation3:
4962 000093A2 AC                  <1>       lodsb
4963 000093A3 3C20                <1>       cmp    al, 20h
4964 000093A5 74FB                <1>       je     short set_env_chk_validation3
4965                              <1>
4966 000093A7 9C                  <1>       pushf
4967                              <1>
4968                              <1>       ; 12/04/2016
4969                              <1> set_env_chk_validation3n:
4970 000093A8 3C61                <1>       cmp    al, 'a'
4971 000093AA 720C                <1>       jb     short set_env_chk_validation3c
4972 000093AC 3C7A                <1>       cmp    al, 'z'
4973 000093AE 7705                <1>       ja     short set_env_chk_validation3x
4974 000093B0 2C20                <1>       sub    al, 'a'-'A'
4975 000093B2 8846FF              <1>       mov    [esi-1], al
4976                              <1>
4977                              <1> set_env_chk_validation3x:
4978 000093B5 AC                  <1>       lodsb
4979 000093B6 EBF0                <1>       jmp    short set_env_chk_validation3n
4980                              <1>
4981                              <1> set_env_chk_validation3c:
4982 000093B8 3C20                <1>       cmp    al, 20h
4983 000093BA 73F9                <1>       jnb    short set_env_chk_validation3x
4984                              <1>
4985 000093BC 803F00              <1>       cmp    byte [edi], 0
4986 000093BF 7731                <1>       ja     short set_env_chk_validation4
4987                              <1>
4988 000093C1 9D                  <1>       popf
4989 000093C2 7228                <1>       jb     short set_env_string_nothing
4990                              <1>
4991 000093C4 B900020000          <1>       mov    ecx, Env_Page_Size ; 512 (4096)
4992                              <1>
4993 000093C9 89DE                <1>       mov    esi, ebx ; 12/04/2016
4994                              <1>
4995                              <1> set_env_string_copy_to_envb:
```

```
4996 000093CB AC                <1>       lodsb
4997 000093CC 3C20              <1>       cmp    al, 20h
4998 000093CE 720A              <1>       jb     short set_env_string_copy_to_envb_z
4999 000093D0 AA                <1>       stosb
5000 000093D1 E2F8              <1>       loop   set_env_string_copy_to_envb
5001                            <1>
5002                            <1>       ; 11/04/2016
5003 000093D3 89D7              <1>       mov    edi, edx ; Env_Page
5004 000093D5 B900020000        <1>       mov    ecx, Env_Page_Size
5005                            <1>
5006                            <1> set_env_string_copy_to_envb_z:
5007 000093DA 52                <1>       push   edx  ; Start address of the variable
5008 000093DB BA00020000        <1>       mov    edx, Env_Page_Size
5009 000093E0 29CA              <1>       sub    edx, ecx ; variable (string) length
5010                            <1>
5011 000093E2 28C0              <1>       sub    al, al ; 0
5012 000093E4 F3AA              <1>       rep    stosb ; clear remain bytes of the env page
5013                            <1>
5014 000093E6 58                <1>       pop    eax  ; Start address of the variable
5015                            <1>
5016                            <1> set_env_string_allocate_envb_retn:  ; stc or clc return
5017 000093E7 5F                <1>       pop    edi ; ****
5018 000093E8 5B                <1>       pop    ebx ; ***
5019 000093E9 59                <1>       pop    ecx ; **
5020 000093EA 5E                <1>       pop    esi ; *
5021 000093EB C3                <1>       retn
5022                            <1>
5023                            <1> set_env_string_nothing:
5024 000093EC 31C0              <1>       xor    eax, eax
5025 000093EE 31D2              <1>       xor    edx, edx ; 11/04/2016
5026 000093F0 EBF5              <1>       jmp    short set_env_string_allocate_envb_retn
5027                            <1>
5028                            <1> set_env_chk_validation4:
5029                            <1>       ; 11/04/2016
5030 000093F2 9D                <1>       popf
5031                            <1>
5032 000093F3 89D6              <1>       mov    esi, edx  ; Env_Page
5033                            <1>
5034                            <1> set_env_chk_validation5:
5035 000093F5 89DF              <1>       mov    edi, ebx  ; ASCIIZ environment string address
5036 000093F7 0FB6CC            <1>       movzx  ecx, ah ; Variable (string) length (with '=')
5037                            <1>
5038                            <1> set_env_chk_validation5_loop:
5039 000093FA AC                <1>       lodsb
5040 000093FB AE                <1>       scasb
5041 000093FC 750A              <1>       jne    short set_env_chk_validation6
5042 000093FE E2FA              <1>       loop   set_env_chk_validation5_loop
5043                            <1>
5044 00009400 3C3D              <1>       cmp    al, '='
5045 00009402 0F8483000000      <1>         je      set_env_change_variable
5046                            <1>
5047                            <1> set_env_chk_validation6:
5048 00009408 08C0              <1>       or     al, al ; 0
5049 0000940A 7403              <1>       jz     short set_env_chk_validation7
5050                            <1>
5051 0000940C AC                <1>       lodsb
5052 0000940D EBF9              <1>       jmp    short set_env_chk_validation6
5053                            <1>
5054                            <1> set_env_chk_validation7:
5055 0000940F 88E1              <1>       mov    cl, ah
5056 00009411 01F1              <1>       add    ecx, esi
5057 00009413 81F9FF310900      <1>       cmp    ecx, Env_Page + Env_Page_Size - 1
5058                            <1>              ; 511 (4095)
5059                            <1>              ; strlen + '=' + 0
5060 00009419 72DA              <1>       jb     short set_env_chk_validation5
5061                            <1>
5062                            <1> set_env_chk_validation8: ; variable not found
5063 0000941B 0FB6F4            <1>       movzx  esi, ah  ; variable name length (with '=')
5064 0000941E 01DE              <1>       add    esi, ebx ; position just after of the '='
5065                            <1>
5066                            <1> set_env_chk_validation8_loop:
5067 00009420 AC                <1>       lodsb
5068 00009421 3C20              <1>       cmp    al, 20h
5069 00009423 74FB              <1>       je     short set_env_chk_validation8_loop
5070 00009425 72C5              <1>       jb     short set_env_string_nothing
5071                            <1>
5072                            <1> set_env_chk_validation9:
5073 00009427 AC                <1>       lodsb
5074 00009428 3C20              <1>       cmp    al, 20h
5075 0000942A 73FB              <1>       jnb    short set_env_chk_validation9
5076                            <1>
5077                            <1>       ; End of ASCIIZ environment string
5078                            <1>
5079                            <1> set_env_add_variable:
5080 0000942C 29DE              <1>       sub    esi, ebx ; variable+definition length
5081                            <1>
5082 0000942E 56                <1>       push   esi ; *****
5083                            <1>
5084 0000942F 89D6              <1>       mov    esi, edx ; Environment page address
5085                            <1>
5086 00009431 B900020000        <1>       mov    ecx, Env_Page_Size ; 512 (4096)
5087                            <1>
5088                            <1> set_env_add_variable_loop:
5089 00009436 AC                <1>       lodsb
5090 00009437 20C0              <1>       and    al, al
5091 00009439 7406              <1>       jz     short set_env_add_variable_chk1 ; 0
5092 0000943B E2F9              <1>       loop   set_env_add_variable_loop
5093                            <1>
5094                            <1>       ; 11/04/2016
5095 0000943D 884EFF            <1>       mov    [esi-1], cl ; 0
5096 00009440 41                <1>       inc    ecx
5097                            <1>
```

```
                                <1> set_env_add_variable_chk1:
00009441 49                     <1>       dec    ecx
00009442 7408                   <1>       jz     short set_env_add_variable_nspc
00009444 AC                     <1>       lodsb
00009445 08C0                   <1>       or     al, al
00009447 740C                   <1>       jz     short set_env_add_variable_chk2 ; 00
00009449 49                     <1>       dec    ecx
0000944A 75EA                   <1>       jnz    short set_env_add_variable_loop
                                <1>
                                <1> set_env_add_variable_nspc: ; no space on environment page
0000944C 58                     <1>       pop    eax ; *****
0000944D B808000000             <1>       mov    eax, 8 ; No space for new environment string
00009452 F9                     <1>       stc
00009453 EB92                   <1>         jmp     short set_env_string_allocate_envb_retn
                                <1>
                                <1> set_env_add_variable_chk2:
00009455 8B0C24                 <1>       mov    ecx, [esp] ; *****
00009458 4E                     <1>       dec    esi ; beginning address of the new variable
00009459 89F0                   <1>       mov    eax, esi
0000945B 01C8                   <1>       add    eax, ecx ; string length (with CR)
0000945D 81C200020000           <1>       add    edx, Env_Page_Size ; 512 (4096)
00009463 39D0                   <1>       cmp    eax, edx
00009465 77E5                   <1>       ja     short set_env_add_variable_nspc
00009467 49                     <1>       dec    ecx ; except CR at the end
00009468 89CA                   <1>       mov    edx, ecx ; 12/04/2016
0000946A 89F7                   <1>       mov    edi, esi
0000946C 893C24                 <1>       mov    [esp], edi ; ***** ; Start address of new variable
0000946F 89DE                   <1>       mov    esi, ebx ; ASCIIZ environment string address
00009471 F3A4                   <1>       rep    movsb
00009473 28C0                   <1>       sub    al, al
00009475 AA                     <1>       stosb
00009476 58                     <1>       pop    eax ; ***** ; Beginning address of new variable
00009477 81FF00320900           <1>         cmp     edi, Env_Page + Env_Page_Size ; 12/04/2016
0000947D 0F8364FFFFFF           <1>         jnb     set_env_string_allocate_envb_retn ; OK !
00009483 880F                   <1>       mov    [edi], cl ; 0
00009485 F8                     <1>       clc    ; 13/04/2016
00009486 E95CFFFFFF             <1>         jmp     set_env_string_allocate_envb_retn ; OK !
                                <1>
                                <1> set_env_change_variable:
                                <1>       ; 06/04/2016
                                <1>       ; esi = Variable's address in environment page (after '=')
                                <1>       ; edi = ASCIIZ environment string address (after '=')
                                <1>
                                <1>       ; ah = variable length from start to the '='
0000948B 8825[345F0100]         <1>       mov    [env_var_length], ah
                                <1>
00009491 28C9                   <1>       sub    cl, cl ; ecx = 0
                                <1>
00009493 57                     <1>       push   edi ; *****
                                <1>
00009494 89F7                   <1>       mov    edi, esi ; 11/04/2016
                                <1>
                                <1> set_env_change_variable_calc1:
00009496 AC                     <1>       lodsb
00009497 08C0                   <1>       or     al, al
00009499 7403                   <1>       jz     short set_env_change_variable_calc2
                                <1>
0000949B 41                     <1>       inc    ecx ; length of environment string (after the '=')
                                <1>
0000949C EBF8                   <1>       jmp    short set_env_change_variable_calc1
                                <1>
                                <1> set_env_change_variable_calc2:
0000949E 8B3424                 <1>       mov    esi, [esp] ; ASCIIZ environment string address
                                <1>
000094A1 29D2                   <1>       sub    edx, edx
                                <1>
                                <1> set_env_change_variable_calc3:
000094A3 AC                     <1>       lodsb
000094A4 3C20                   <1>       cmp    al, 20h
000094A6 7203                   <1>       jb     short set_env_change_variable_calc4
                                <1>
000094A8 42                     <1>       inc    edx ; length of ASCIIZ string (after the '=')
                                <1>
000094A9 EBF8                   <1>       jmp    short set_env_change_variable_calc3
                                <1>
                                <1> set_env_change_variable_calc4:
000094AB C646FF00               <1>       mov    byte [esi-1], 0  ; put ZERO instead of CR
                                <1>
000094AF 5E                     <1>       pop    esi ; ***** ; ASCIIZ string address (after '=')
                                <1>
                                <1>       ; EDI = Old variable's address (after '=')
                                <1>
                                <1>       ; compare the new string with the old string
000094B0 39CA                   <1>       cmp    edx, ecx
000094B2 7717                   <1>       ja     short set_env_change_variable_calc5 ; longer
000094B4 0F828F000000           <1>         jb      set_env_change_variable_calc9 ; shorter
                                <1>
                                <1>       ;same length (simple copy)
000094BA 0FB6C4                 <1>       movzx  eax, ah
000094BD 01C2                   <1>       add    edx, eax
000094BF F7D8                   <1>       neg    eax
000094C1 01F8                   <1>       add    eax, edi
                                <1>       ; EAX = Start address of the variable
                                <1>       ; EDX = Variable length (without ZERO at the end of variable)
                                <1>
000094C3 F3A4                   <1>       rep    movsb
000094C5 F8                     <1>       clc    ; 13/04/2016
000094C6 E91CFFFFFF             <1>         jmp     set_env_string_allocate_envb_retn ; OK !
                                <1>
                                <1> set_env_change_variable_calc5:
                                <1>       ; 11/04/2016
000094CB 52                     <1>       push   edx ; *****
```

```
5200 000094CC 29CA                <1>       sub    edx, ecx ; difference ; (the new string is longer)
5201 000094CE 89F3                <1>       mov    ebx, esi
5202 000094D0 89FE                <1>       mov    esi, edi
5203                              <1>
5204                              <1> set_env_change_variable_calc6:
5205 000094D2 AC                  <1>       lodsb
5206 000094D3 20C0                <1>       and    al, al
5207 000094D5 75FB                <1>       jnz    short set_env_change_variable_calc6
5208                              <1>
5209 000094D7 81FE00320900        <1>       cmp    esi, Env_Page + Env_Page_Size ; 512 (4096)
5210 000094DD 0F8369FFFFFF        <1>         jnb     set_env_add_variable_nspc
5211                              <1>
5212 000094E3 89F9                <1>       mov    ecx, edi  ; current (old) variable's address
5213 000094E5 89F7                <1>       mov    edi, esi  ; next variable's address
5214                              <1>
5215 000094E7 AC                  <1>       lodsb
5216 000094E8 08C0                <1>       or     al, al
5217 000094EA 7416                <1>       jz     short set_env_change_variable_calc8 ; 00
5218                              <1>
5219                              <1> set_env_change_variable_calc7:
5220 000094EC AC                  <1>       lodsb
5221 000094ED 20C0                <1>       and    al, al
5222 000094EF 75FB                <1>       jnz    short set_env_change_variable_calc7
5223                              <1>
5224 000094F1 81FE00320900        <1>       cmp    esi, Env_Page + Env_Page_Size ; 512 (4096)
5225 000094F7 0F834FFFFFFF        <1>         jnb     set_env_add_variable_nspc
5226                              <1>
5227 000094FD AC                  <1>       lodsb
5228 000094FE 08C0                <1>       or     al, al
5229 00009500 75EA                <1>       jnz    short set_env_change_variable_calc7
5230                              <1>
5231                              <1> set_env_change_variable_calc8:
5232 00009502 4E                  <1>       dec    esi ; address of the second (last) 0 of the 00
5233                              <1>
5234 00009503 01F2                <1>       add    edx, esi ; final position of the last 0
5235                              <1>
5236 00009505 81FA00320900        <1>       cmp    edx, Env_Page + Env_Page_Size ; 512 (4096)
5237 0000950B 0F833BFFFFFF        <1>         jnb     set_env_add_variable_nspc
5238                              <1>
5239 00009511 89C8                <1>       mov    eax, ecx ; old variable's address (after '=')
5240                              <1>
5241 00009513 89F1                <1>       mov    ecx, esi
5242 00009515 29F9                <1>       sub    ecx, edi ; count of bytes to move forward
5243                              <1>
5244                              <1>       ; 13/04/2016
5245 00009517 C60200              <1>       mov    byte [edx], 0
5246 0000951A 89D7                <1>       mov    edi, edx
5247 0000951C 29F2                <1>       sub    edx, esi ; difference (additional byte count)
5248 0000951E 4F                  <1>       dec    edi ; the last zero address (first byte of the 00)
5249 0000951F 89FE                <1>       mov    esi, edi
5250 00009521 29D6                <1>       sub    esi, edx ; - displacement
5251                              <1>
5252 00009523 FA                  <1>       cli    ; disable interrupts
5253 00009524 FD                  <1>       std    ; backward
5254                              <1>
5255 00009525 F3A4                <1>       rep    movsb ; move ECX bytes from DS:ESI to ES:EDI
5256                              <1>
5257 00009527 FC                  <1>       cld    ; forward (default)
5258 00009528 FB                  <1>       sti    ; enable interrupts
5259                              <1>
5260 00009529 89C7                <1>       mov    edi, eax
5261 0000952B 59                  <1>       pop    ecx ; ***** ; byte count (after '=')
5262 0000952C 89CA                <1>       mov    edx, ecx
5263 0000952E 89DE                <1>       mov    esi, ebx ; ASCIIZ string address (after '=')
5264 00009530 89FB                <1>       mov    ebx, edi
5265                              <1>
5266 00009532 F3A4                <1>       rep    movsb
5267                              <1>
5268 00009534 880F                <1>       mov    [edi], cl ; 0 ; end of variable
5269                              <1>
5270 00009536 0FB605[345F0100]    <1>       movzx  eax, byte [env_var_length]
5271 0000953D 01C2                <1>       add    edx, eax ; variable length (total)
5272 0000953F F7D8                <1>       neg    eax
5273 00009541 01D8                <1>       add    eax, ebx ; start address of the variable
5274 00009543 F8                  <1>       clc    ; 13/04/2016
5275 00009544 E99EFEFFFF          <1>         jmp     set_env_string_allocate_envb_retn ; OK !
5276                              <1>
5277                              <1> set_env_change_variable_calc9:
5278                              <1>       ; 11/04/2016
5279 00009549 21D2                <1>       and    edx, edx ; is empty ?
5280 0000954B 753B                <1>       jnz    short set_env_change_variable_calc15
5281                              <1>
5282 0000954D 0FB6DC              <1>       movzx  ebx, ah
5283 00009550 F7DB                <1>       neg    ebx
5284 00009552 01FB                <1>       add    ebx, edi
5285                              <1>
5286                              <1>       ; EBX = Start address of the variable (in env page)
5287                              <1>       ; EDX = Variable length = 0
5288                              <1>
5289 00009554 89FE                <1>       mov    esi, edi
5290                              <1>
5291                              <1> set_env_change_variable_calc10:
5292 00009556 AC                  <1>       lodsb
5293 00009557 08C0                <1>       or     al, al
5294 00009559 75FB                <1>       jnz    short set_env_change_variable_calc10
5295                              <1>
5296 0000955B B9FF310900          <1>       mov    ecx, Env_Page + Env_Page_Size - 1
5297                              <1>
5298 00009560 39CE                <1>       cmp    esi, ecx ; +511 (+4095)
5299 00009562 7604                <1>       jna    short set_env_change_variable_calc11
5300                              <1>
5301 00009564 89CE                <1>       mov    esi, ecx
```

```
5302 00009566 8806              <1>       mov    [esi], al ; 0
5303                            <1>
5304                            <1> set_env_change_variable_calc11:
5305 00009568 89DF              <1>       mov    edi, ebx ; old variable's start address
5306                            <1>
5307                            <1> set_env_change_variable_calc12:
5308 0000956A AC                <1>       lodsb
5309 0000956B AA                <1>       stosb
5310 0000956C 20C0              <1>       and    al, al
5311 0000956E 75FA              <1>       jnz    short set_env_change_variable_calc12
5312 00009570 39CE              <1>       cmp    esi, ecx
5313 00009572 7706              <1>       ja     short set_env_change_variable_calc13
5314 00009574 AC                <1>       lodsb
5315 00009575 AA                <1>       stosb
5316 00009576 20C0              <1>       and    al, al
5317 00009578 75F0              <1>       jnz    short set_env_change_variable_calc12
5318                            <1>
5319                            <1> set_env_change_variable_calc13:
5320 0000957A 29F9              <1>       sub    ecx, edi
5321 0000957C 7203              <1>       jb     short set_env_change_variable_calc14
5322 0000957E 41                <1>       inc    ecx ; 1-512 (1-4096)
5323 0000957F F3AA              <1>       rep    stosb ; al = 0
5324                            <1>
5325                            <1> set_env_change_variable_calc14:
5326 00009581 29C0              <1>       sub    eax, eax ; Start address of the variable
5327                            <1>       ; EAX = 0 -> Variable is removed
5328                            <1>       ; EDX = Variable length = 0
5329                            <1>
5330 00009583 E95FFEFFFF        <1>         jmp     set_env_string_allocate_envb_retn ; OK !
5331                            <1>
5332                            <1> set_env_change_variable_calc15:
5333 00009588 52                <1>       push   edx ; *****
5334 00009589 F7DA              <1>       neg    edx
5335 0000958B 01CA              <1>       add    edx, ecx ; difference (the old string is longer)
5336 0000958D 89F3              <1>       mov    ebx, esi
5337 0000958F 89FE              <1>       mov    esi, edi
5338                            <1>
5339                            <1> set_env_change_variable_calc16:
5340 00009591 AC                <1>       lodsb
5341 00009592 20C0              <1>       and    al, al
5342 00009594 75FB              <1>       jnz    short set_env_change_variable_calc16
5343                            <1>
5344 00009596 B900320900        <1>       mov    ecx, Env_Page + Env_Page_Size
5345                            <1>
5346 0000959B 39CE              <1>       cmp    esi, ecx ; +512 (+4096)
5347 0000959D 7605              <1>       jna    short set_env_change_variable_calc17
5348                            <1>
5349 0000959F 89CE              <1>       mov    esi, ecx
5350 000095A1 8846FF            <1>       mov    [esi-1], al ; 0
5351                            <1>
5352                            <1> set_env_change_variable_calc17:
5353 000095A4 89F9              <1>       mov    ecx, edi  ; current (old) variable's address
5354 000095A6 89F7              <1>       mov    edi, esi  ; next variable's address
5355                            <1>
5356 000095A8 AC                <1>       lodsb
5357 000095A9 08C0              <1>       or     al, al
5358 000095AB 741D              <1>       jz     short set_env_change_variable_calc20
5359                            <1>
5360                            <1> set_env_change_variable_calc18:
5361 000095AD AC                <1>       lodsb
5362 000095AE 20C0              <1>       and    al, al
5363 000095B0 75FB              <1>       jnz    short set_env_change_variable_calc18
5364                            <1>
5365 000095B2 81FE00320900      <1>       cmp    esi, Env_Page + Env_Page_Size
5366 000095B8 720B              <1>       jb     short set_env_change_variable_calc19
5367 000095BA 740E              <1>       je     short set_env_change_variable_calc20
5368                            <1>
5369 000095BC BEFF310900        <1>       mov    esi, Env_Page + Env_Page_Size - 1
5370 000095C1 8806              <1>       mov    [esi], al ; 0
5371 000095C3 EB06              <1>       jmp    short set_env_change_variable_calc21
5372                            <1>
5373                            <1> set_env_change_variable_calc19:
5374 000095C5 AC                <1>       lodsb
5375 000095C6 08C0              <1>       or     al, al
5376 000095C8 75E3              <1>       jnz    short set_env_change_variable_calc18
5377                            <1>
5378                            <1> set_env_change_variable_calc20:
5379 000095CA 4E                <1>       dec    esi ; address of the second (last) 0 of the 00
5380                            <1>
5381                            <1> set_env_change_variable_calc21:
5382                            <1>       ; edx = difference (byte count)
5383                            <1>
5384 000095CB 89C8              <1>       mov    eax, ecx ; old variable's address (after '=')
5385                            <1>
5386 000095CD 89F1              <1>       mov    ecx, esi
5387 000095CF 29F9              <1>       sub    ecx, edi ; count of bytes to move backward
5388                            <1>
5389 000095D1 89FE              <1>       mov    esi, edi ; next variable's address
5390 000095D3 29D7              <1>       sub    edi, edx ; (displacement)
5391                            <1>
5392 000095D5 F3A4              <1>       rep    movsb
5393                            <1>
5394 000095D7 880F              <1>       mov    [edi], cl ; 0 ; 00 ; end of environment variables
5395                            <1>
5396 000095D9 89C7              <1>       mov    edi, eax
5397 000095DB 5A                <1>       pop    edx ; ***** ; byte count (after '=')
5398 000095DC 89D1              <1>       mov    ecx, edx
5399 000095DE 89DE              <1>       mov    esi, ebx ; ASCIIZ string address (after '=')
5400 000095E0 89FB              <1>       mov    ebx, edi
5401                            <1>
5402 000095E2 F3A4              <1>       rep    movsb
5403                            <1>
```

```
5404 000095E4 880F                <1>       mov    [edi], cl ; 0 ; end of variable
5405                              <1>
5406 000095E6 0FB605[345F0100]    <1>       movzx  eax, byte [env_var_length]
5407 000095ED 01C2                <1>       add    edx, eax ; variable length (total)
5408 000095EF F7D8                <1>       neg    eax
5409 000095F1 01D8                <1>       add    eax, ebx ; start address of the variable
5410 000095F3 F8                  <1>       clc    ; 13/04/2016
5411 000095F4 E9EEFDFFFF          <1>         jmp     set_env_string_allocate_envb_retn ; OK !
5412                              <1>
5413                              <1> mainprog_startup_configuration:
5414                              <1>       ; 06/05/2016
5415                              <1>       ; 14/04/2016 (TRDOS 386 = TRDOS v2.0)
5416                              <1>       ; 17/09/2011 (TRDOS v1, MAINPROG.ASM)
5417                              <1>       ;
5418                              <1> loc_load_mainprog_cfg_file:
5419 000095F9 BE[F3060100]        <1>       mov    esi, MainProgCfgFile
5420 000095FE 66B80018            <1>       mov    ax, 1800h ; Except volume label and dirs
5421 00009602 E8E8E9FFFF          <1>       call   find_first_file
5422 00009607 7256                <1>       jc     short loc_load_mainprog_cfg_exit
5423                              <1>
5424                              <1>       ;or    eax, eax
5425                              <1>       ;jz    short loc_load_mainprog_cfg_exit
5426                              <1>
5427                              <1> loc_start_mainprog_configuration:
5428                              <1>       ; ESI = FindFile_DirEntry Location
5429                              <1>       ; EAX = File Size
5430                              <1>
5431 00009609 A3[B4520100]        <1>       mov    [MainProgCfg_FileSize], eax
5432                              <1>
5433 0000960E 668B5614            <1>       mov    dx, [esi+DirEntry_FstClusHI]
5434 00009612 C1E210              <1>       shl    edx, 16
5435 00009615 668B561A            <1>       mov    dx, [esi+DirEntry_FstClusLO]
5436 00009619 8915[E85E0100]      <1>       mov    [csftdf_sf_cluster], edx
5437                              <1>
5438 0000961F 89C1                <1>       mov    ecx, eax
5439 00009621 29C0                <1>       sub    eax, eax
5440                              <1>
5441                              <1>       ; TRDOS 386 (TRDOS v2.0)
5442                              <1>       ; Allocate contiguous memory block for loading the file
5443                              <1>
5444                              <1>       ; eax = 0 (Allocate memory from the beginning)
5445                              <1>       ; ecx = File (Allocation) size in bytes
5446                              <1>
5447 00009623 E8FBBDFFFF          <1>       call   allocate_memory_block
5448 00009628 7235                <1>       jc     short loc_load_mainprog_cfg_exit
5449                              <1>
5450 0000962A A3[E05E0100]        <1>       mov    [csftdf_sf_mem_addr], eax ; loading address
5451 0000962F 890D[E45E0100]      <1>       mov    [csftdf_sf_mem_bsize], ecx ; block size
5452                              <1>
5453 00009635 31DB                <1>       xor    ebx, ebx
5454                              <1>       ;mov   [csftdf_sf_rbytes], ebx ; 0, reset
5455                              <1>
5456 00009637 8A3D[C6520100]      <1>       mov    bh, [Current_Drv] ; [FindFile_Drv]
5457 0000963D BE00010900          <1>       mov    esi, Logical_DOSDisks
5458 00009642 01DE                <1>       add    esi, ebx
5459                              <1>
5460 00009644 8B1D[E05E0100]      <1>       mov    ebx, [csftdf_sf_mem_addr] ; memory block address
5461                              <1>
5462 0000964A 807E0300            <1>       cmp    byte [esi+LD_FATType], 0
5463 0000964E 7710                <1>         ja   short loc_mcfg_load_fat_file
5464                              <1>
5465 00009650 C705[F05E0100]0000- <1>       mov    dword [csftdf_r_size], 65536
5465 00009658 0100                <1>
5466 0000965A E992010000          <1>         jmp     loc_mcfg_load_fs_file
5467                              <1>
5468                              <1> loc_load_mainprog_cfg_exit:
5469 0000965F C3                  <1>       retn
5470                              <1>
5471                              <1> loc_mcfg_load_fat_file:
5472 00009660 0FB74611            <1>       movzx  eax, word [esi+LD_BPB+BytesPerSec]
5473 00009664 0FB64E13            <1>       movzx  ecx, byte [esi+LD_BPB+SecPerClust]
5474 00009668 F7E1                <1>       mul    ecx
5475 0000966A A3[F05E0100]        <1>       mov    [csftdf_r_size], eax
5476                              <1>
5477                              <1> loc_mcfg_load_fat_file_next:
5478 0000966F E813010000          <1>       call   mcfg_read_fat_file_sectors
5479 00009674 0F82F7000000        <1>         jc      mcfg_deallocate_mem
5480                              <1>
5481 0000967A 09D2                <1>       or     edx, edx ; edx > 0 -> EOF
5482 0000967C 74F1                <1>       jz     short loc_mcfg_load_fat_file_next
5483                              <1>
5484                              <1> loc_mcfg_load_fat_file_ok:
5485                              <1>       ; 06/05/2016
5486 0000967E C705[845F0100]-     <1>       mov    dword [mainprog_return_addr], loc_mcfg_ci_return_addr
5486 00009684 [32970000]          <1>
5487                              <1>       ;
5488 00009688 8B35[E05E0100]      <1>       mov    esi, [csftdf_sf_mem_addr]
5489 0000968E 8935[B8520100]      <1>       mov    [MainProgCfg_LineOffset], esi
5490                              <1>
5491 00009694 A1[B4520100]        <1>       mov    eax, [MainProgCfg_FileSize]
5492 00009699 89C2                <1>       mov    edx, eax
5493 0000969B 01F2                <1>       add    edx, esi
5494                              <1>
5495                              <1> loc_mcfg_process_next_line_check:
5496 0000969D 89C1                <1>       mov    ecx, eax
5497                              <1>
5498 0000969F 803E2A              <1>       cmp    byte [esi], "*" ; Remark sign
5499 000096A2 7503                <1>       jne    short loc_mcfg_process_next_line
5500 000096A4 46                  <1>       inc    esi
5501 000096A5 EB17                <1>       jmp    short loc_move_mainprog_cfg_nl1
5502                              <1>
5503                              <1> loc_mcfg_process_next_line:
```

```
5504 000096A7 83F94F              <1>       cmp    ecx, 79
5505 000096AA 7605                <1>       jna    short loc_start_mainprog_cfg_process
5506                              <1>
5507 000096AC B94F000000          <1>       mov    ecx, 79
5508                              <1>
5509                              <1> loc_start_mainprog_cfg_process:
5510 000096B1 BF[76530100]        <1>       mov    edi, CommandBuffer
5511                              <1>
5512                              <1> loc_move_mainprog_cfg_line:
5513 000096B6 AC                  <1>       lodsb
5514 000096B7 3C20                <1>       cmp    al, 20h
5515 000096B9 720C                <1>       jb     short loc_move_mainprog_cfg_nl2
5516 000096BB AA                  <1>       stosb
5517 000096BC E2F8                <1>       loop   loc_move_mainprog_cfg_line
5518                              <1>
5519                              <1> loc_move_mainprog_cfg_nl1:
5520 000096BE 39D6                <1>       cmp    esi, edx ; + configuration file size
5521 000096C0 7312                <1>       jnb    short loc_end_of_mainprog_cfg_line
5522 000096C2 AC                  <1>       lodsb
5523 000096C3 3C20                <1>       cmp    al, 20h
5524 000096C5 73F7                <1>       jnb    short loc_move_mainprog_cfg_nl1
5525                              <1>
5526                              <1> loc_move_mainprog_cfg_nl2:
5527 000096C7 39D6                <1>       cmp    esi, edx
5528 000096C9 7309                <1>       jnb    short loc_end_of_mainprog_cfg_line
5529 000096CB 8A06                <1>       mov    al, [esi]
5530 000096CD 3C20                <1>       cmp    al, 20h
5531 000096CF 7703                <1>       ja     short loc_end_of_mainprog_cfg_line
5532 000096D1 46                  <1>       inc    esi
5533 000096D2 EBF3                <1>       jmp    short loc_move_mainprog_cfg_nl2
5534                              <1>
5535                              <1> loc_end_of_mainprog_cfg_line:
5536 000096D4 C60700              <1>       mov    byte [edi], 0
5537                              <1>
5538 000096D7 8935[B8520100]      <1>       mov    [MainProgCfg_LineOffset], esi
5539                              <1>
5540                              <1> loc_move_mainprog_cfg_command:
5541 000096DD BE[76530100]        <1>       mov    esi, CommandBuffer
5542 000096E2 89F7                <1>       mov    edi, esi
5543 000096E4 31DB                <1>       xor    ebx, ebx
5544                              <1>       ;xor   ecx, ecx
5545 000096E6 30C9                <1>       xor    cl, cl
5546                              <1>
5547                              <1> loc_move_mcfg_first_cmd_char:
5548 000096E8 8A041E              <1>       mov    al, [esi+ebx]
5549 000096EB FEC3                <1>       inc    bl
5550 000096ED 3C20                <1>       cmp    al, 20h
5551 000096EF 7712                <1>       ja     short loc_move_mcfg_cmd_capitalizing
5552 000096F1 7237                <1>       jb     short loc_move_mcfg_cmd_arguments_ok
5553 000096F3 80FB4F              <1>       cmp    bl, 79
5554 000096F6 72F0                <1>       jb     short loc_move_mcfg_first_cmd_char
5555 000096F8 EB30                <1>       jmp    short loc_move_mcfg_cmd_arguments_ok
5556                              <1>
5557                              <1> loc_move_mcfg_next_cmd_char:
5558 000096FA 8A041E              <1>       mov    al, [esi+ebx]
5559 000096FD FEC3                <1>       inc    bl
5560 000096FF 3C20                <1>       cmp    al, 20h
5561 00009701 7614                <1>       jna    short loc_move_mcfg_cmd_ok
5562                              <1>
5563                              <1> loc_move_mcfg_cmd_capitalizing:
5564 00009703 3C61                <1>       cmp    al, 61h ; 'a'
5565 00009705 7206                <1>       jb     short loc_move_mcfg_cmd_caps_ok
5566 00009707 3C7A                <1>       cmp    al, 7Ah ; 'z'
5567 00009709 7702                <1>       ja     short loc_move_mcfg_cmd_caps_ok
5568 0000970B 24DF                <1>       and    al, 0DFh ; sub      al, 'a'-'A'
5569                              <1>
5570                              <1> loc_move_mcfg_cmd_caps_ok:
5571 0000970D AA                  <1>       stosb
5572 0000970E FEC1                <1>       inc    cl
5573 00009710 80FB4F              <1>       cmp    bl, 79
5574 00009713 72E5                <1>       jb     short loc_move_mcfg_next_cmd_char
5575 00009715 EB13                <1>       jmp    short loc_move_mcfg_cmd_arguments_ok
5576                              <1>
5577                              <1> loc_move_mcfg_cmd_ok:
5578 00009717 30C0                <1>       xor    al, al ; 0
5579                              <1>
5580                              <1> loc_move_mcfg_cmd_arguments:
5581 00009719 8807                <1>       mov    [edi], al
5582 0000971B 47                  <1>       inc    edi
5583 0000971C 80FB4F              <1>       cmp    bl, 79
5584 0000971F 7309                <1>       jnb    short loc_move_mcfg_cmd_arguments_ok
5585 00009721 8A041E              <1>       mov    al, [esi+ebx]
5586 00009724 FEC3                <1>       inc    bl
5587 00009726 3C20                <1>       cmp    al, 20h
5588 00009728 73EF                <1>       jnb    short loc_move_mcfg_cmd_arguments
5589                              <1>
5590                              <1> loc_move_mcfg_cmd_arguments_ok:
5591 0000972A C60700              <1>       mov    byte [edi], 0
5592                              <1>
5593                              <1> loc_mcfg_process_cmd_interpreter:
5594 0000972D E8F4DFFFFF          <1>       call    command_interpreter
5595                              <1>
5596                              <1> loc_mcfg_ci_return_addr:
5597 00009732 A1[B4520100]        <1>       mov    eax, [MainProgCfg_FileSize]
5598 00009737 89C2                <1>       mov    edx, eax
5599 00009739 8B35[B8520100]      <1>       mov    esi, [MainProgCfg_LineOffset]
5600 0000973F 01F2                <1>       add    edx, esi
5601 00009741 0305[E05E0100]      <1>       add    eax, [csftdf_sf_mem_addr]
5602 00009747 29F0                <1>       sub    eax, esi
5603 00009749 0F874EFFFFFF        <1>         ja      loc_mcfg_process_next_line_check
5604                              <1>
5605 0000974F E81D000000          <1>       call   mcfg_deallocate_mem
```

```
5606                                    <1>
5607 00009754 B94F000000                <1>     mov     ecx, 79 ; 80 ?
5608 00009759 BF[76530100]              <1>     mov     edi, CommandBuffer
5609 0000975E 30C0                      <1>     xor     al, al
5610 00009760 F3AA                      <1>     rep     stosb
5611                                    <1>
5612                                    <1>     ; 06/05/2016
5613 00009762 BE[4B130100]              <1>     mov     esi, nextline
5614 00009767 E8F1CBFFFF                <1>     call    print_msg
5615 0000976C E92DD6FFFF                <1>     jmp     dos_prompt
5616                                    <1>
5617                                    <1> mcfg_deallocate_mem:
5618 00009771 A1[E05E0100]              <1>     mov     eax, [csftdf_sf_mem_addr] ; start address
5619 00009776 8B0D[E45E0100]            <1>     mov     ecx, [csftdf_sf_mem_bsize] ; block size
5620                                    <1>     ;call  deallocate_memory_block
5621                                    <1>     ;retn
5622 0000977C E9AFBEFFFF                <1>     jmp     deallocate_memory_block
5623                                    <1>
5624                                    <1> mcfg_read_file_sectors:
5625                                    <1>     ; 14/04/2016
5626 00009781 807E0300                  <1>     cmp     byte [esi+LD_FATType], 0
5627 00009785 7669                      <1>       jna  short mcfg_read_fs_file_sectors
5628                                    <1>
5629                                    <1> mcfg_read_fat_file_sectors:
5630                                    <1>     ; return:
5631                                    <1>     ;   CF = 0 & EDX > 0 -> END OF FILE
5632                                    <1>     ;   CF = 0 & EDX = 0 -> not EOF
5633                                    <1>     ;   CF = 1 -> read error (error code in AL)
5634                                    <1>
5635                                    <1> mcfg_read_fat_file_secs_0:
5636 00009787 8B15[B4520100]            <1>     mov     edx, [MainProgCfg_FileSize]
5637 0000978D 2B15[F85E0100]            <1>     sub     edx, [csftdf_sf_rbytes]
5638 00009793 3B15[F05E0100]            <1>     cmp     edx, [csftdf_r_size]
5639 00009799 7306                      <1>     jnb     short mcfg_read_fat_file_secs_1
5640 0000979B 8915[F05E0100]            <1>     mov     [csftdf_r_size], edx
5641                                    <1>
5642                                    <1> mcfg_read_fat_file_secs_1:
5643 000097A1 A1[F05E0100]              <1>     mov     eax, [csftdf_r_size]
5644 000097A6 29D2                      <1>     sub     edx, edx
5645 000097A8 0FB74E11                  <1>     movzx   ecx, word [esi+LD_BPB+BytesPerSec]
5646 000097AC 01C8                      <1>     add     eax, ecx
5647 000097AE 48                        <1>     dec     eax
5648 000097AF F7F1                      <1>     div     ecx
5649 000097B1 89C1                      <1>     mov     ecx, eax ; sector count
5650 000097B3 A1[E85E0100]              <1>     mov     eax, [csftdf_sf_cluster]
5651                                    <1>
5652                                    <1>     ; EBX = memory block address (current)
5653                                    <1>
5654 000097B8 E88E230000                <1>     call    read_fat_file_sectors
5655 000097BD 7230                      <1>     jc      short mcfg_read_fat_file_secs_3
5656                                    <1>
5657                                    <1>     ; EBX = next memory address
5658                                    <1>
5659 000097BF A1[F85E0100]              <1>     mov     eax, [csftdf_sf_rbytes]
5660 000097C4 0305[F05E0100]            <1>     add     eax, [csftdf_r_size]
5661 000097CA 8B15[B4520100]            <1>     mov     edx, [MainProgCfg_FileSize]
5662 000097D0 39D0                      <1>     cmp     eax, edx
5663 000097D2 731B                      <1>     jnb     short mcfg_read_fat_file_secs_3 ; edx > 0
5664 000097D4 A3[F85E0100]              <1>     mov     [csftdf_sf_rbytes], eax
5665                                    <1>
5666 000097D9 53                        <1>     push    ebx ; *
5667                                    <1>     ; get next cluster (csftdf_r_size! bytes)
5668 000097DA A1[E85E0100]              <1>     mov     eax, [csftdf_sf_cluster]
5669 000097DF E839210000                <1>     call    get_next_cluster
5670 000097E4 5B                        <1>     pop     ebx ; *
5671 000097E5 7301                      <1>     jnc     short mcfg_read_fat_file_secs_2
5672                                    <1>
5673                                    <1>     ;mov   eax, 17; Read error !
5674 000097E7 C3                        <1>     retn
5675                                    <1>
5676                                    <1> mcfg_read_fat_file_secs_2:
5677 000097E8 29D2                      <1>     sub     edx, edx ; 0
5678 000097EA A3[E85E0100]              <1>     mov     [csftdf_sf_cluster], eax ; next cluster
5679                                    <1>
5680                                    <1> mcfg_read_fat_file_secs_3:
5681 000097EF C3                        <1>     retn
5682                                    <1>
5683                                    <1> mcfg_read_fs_file_sectors:
5684 000097F0 C3                        <1>     retn
5685                                    <1>
5686                                    <1> loc_mcfg_load_fs_file:
5687 000097F1 C3                        <1>     retn
5688                                    <1>
5689                                    <1> load_and_execute_file:
5690                                    <1>     ; 04/01/2017
5691                                    <1>     ; 06/05/2016, 07/05/2016, 11/05/2016
5692                                    <1>     ; 23/04/2016, 24/04/2016
5693                                    <1>     ; 22/04/2016 (TRDOS 386 = TRDOS v2.0)
5694                                    <1>     ; 05/11/2011
5695                                    <1>     ; (TRDOS v1, CMDINTR.ASM, 'cmp_cmd_run', 'cmp_cmd_external')
5696                                    <1>     ; ('loc_run_check_filename')
5697                                    <1>     ; 29/08/2011
5698                                    <1>     ; 10/09/2011
5699                                    <1>     ; INPUT->
5700                                    <1>     ;     ESI = Path Name address (CommandBuffer address)
5701                                    <1>     ; OUTPUT ->
5702                                    <1>     ;     none (error message will be shown if an error will occur)
5703                                    <1>     ;
5704                                    <1>     ; (EAX, EBX, ECX, EDX, ESI, EDI, EBP will be changed)
5705                                    <1>     ;
5706                                    <1> loc_run_check_filename:
5707 000097F2 803E20                    <1>     cmp     byte [esi], 20h
```

```
5708 000097F5 0F82F2E2FFFF        <1>       jb    loc_cmd_failed
5709 000097FB 7703                <1>       ja    short loc_run_check_filename_ok
5710 000097FD 46                  <1>       inc   esi
5711 000097FE EBF2                <1>       jmp   short loc_run_check_filename
5712                              <1>
5713                              <1> loc_run_check_filename_ok:
5714 00009800 C605[27530100]00    <1>       mov   byte [CmdArgStart], 0 ; reset
5715 00009807 56                  <1>       push  esi ; *
5716                              <1> loc_run_get_first_arg_pos:
5717 00009808 46                  <1>       inc   esi
5718 00009809 8A06                <1>       mov   al, [esi]
5719 0000980B 3C20                <1>       cmp   al, 20h
5720 0000980D 77F9                <1>       ja    short loc_run_get_first_arg_pos
5721 0000980F C60600              <1>       mov   byte [esi], 0
5722                              <1> loc_run_get_external_arg_pos:
5723                              <1>       ; 11/05/2016
5724 00009812 46                  <1>       inc   esi
5725 00009813 8A06                <1>       mov   al, [esi]
5726 00009815 3C20                <1>       cmp   al, 20h
5727 00009817 760C                <1>       jna   short loc_run_parse_path_name
5728 00009819 89F0                <1>       mov   eax, esi
5729 0000981B 2D[76530100]        <1>       sub   eax, CommandBuffer
5730 00009820 A2[27530100]        <1>       mov   byte [CmdArgStart], al
5731                              <1> loc_run_parse_path_name:
5732 00009825 5E                  <1>       pop   esi ; *
5733 00009826 BF[6A5C0100]        <1>       mov   edi, FindFile_Drv
5734 0000982B E8D6090000          <1>       call  parse_path_name
5735 00009830 0F82B7E2FFFF        <1>       jc    loc_cmd_failed
5736                              <1>
5737                              <1> loc_run_check_filename_exists:
5738 00009836 BE[AC5C0100]        <1>       mov   esi, FindFile_Name
5739 0000983B 803E20              <1>       cmp   byte [esi], 20h
5740 0000983E 0F86A9E2FFFF        <1>       jna   loc_cmd_failed
5741                              <1>
5742                              <1> loc_run_check_exe_filename_ext:
5743 00009844 E891020000          <1>       call  check_prg_filename_ext
5744 00009849 0F829EE2FFFF        <1>       jc    loc_cmd_failed
5745                              <1>
5746                              <1> loc_run_check_exe_filename_ext_ok:
5747 0000984F 66A3[825F0100]      <1>       mov   word [EXE_ID], ax
5748                              <1>
5749                              <1> loc_run_drv:
5750 00009855 C605[815F0100]00    <1>       mov   byte [Run_Manual_Path], 0
5751 0000985C A1[C0520100]        <1>       mov   eax, [Current_Dir_FCluster]
5752 00009861 A3[7C5F0100]        <1>         mov     [Run_CDirFC], eax
5753                              <1>       ;
5754 00009866 8A35[C6520100]      <1>       mov   dh, [Current_Drv]
5755 0000986C 8835[265B0100]      <1>       mov   [RUN_CDRV], dh
5756                              <1>
5757 00009872 8A15[6A5C0100]      <1>       mov   dl, [FindFile_Drv]
5758 00009878 38F2                <1>       cmp   dl, dh
5759 0000987A 7412                <1>       je    short loc_run_change_directory
5760                              <1>
5761 0000987C 8005[815F0100]02    <1>       add   byte [Run_Manual_Path], 2
5762                              <1>
5763 00009883 E8D5D3FFFF          <1>       call  change_current_drive
5764 00009888 0F828AE2FFFF        <1>       jc    loc_run_cmd_failed
5765                              <1>
5766                              <1> loc_run_change_directory:
5767 0000988E 803D[6B5C0100]20    <1>       cmp   byte [FindFile_Directory], 20h
5768 00009895 7623                <1>       jna   short loc_run_find_executable_file
5769                              <1>
5770 00009897 FE05[815F0100]      <1>       inc   byte [Run_Manual_Path]
5771                              <1>
5772 0000989D FE05[AD060100]      <1>       inc   byte [Restore_CDIR]
5773                              <1>
5774 000098A3 BE[6B5C0100]        <1>       mov   esi, FindFile_Directory
5775 000098A8 30E4                <1>       xor   ah, ah ; CD_COMMAND sign -> 0
5776 000098AA E843030000          <1>       call  change_current_directory
5777 000098AF 0F8263E2FFFF        <1>       jc    loc_run_cmd_failed
5778                              <1>
5779                              <1> loc_run_change_prompt_dir_string:
5780 000098B5 E858020000          <1>       call  change_prompt_dir_string
5781                              <1>
5782                              <1> loc_run_find_executable_file:
5783 000098BA 66C705[805F0100]00- <1>       mov   word [Run_Auto_Path], 0
5783 000098C2 00                  <1>
5784                              <1>
5785                              <1> loc_run_find_executable_file_next:
5786 000098C3 BE[AC5C0100]        <1>       mov   esi, FindFile_Name
5787                              <1> loc_run_find_program_file_next:
5788 000098C8 66B80018            <1>       mov   ax, 1800h ; Except volume label and dirs
5789 000098CC E81EE7FFFF          <1>       call  find_first_file
5790                              <1>       ; ESI = Directory Entry (FindFile_DirEntry) Location
5791                              <1>       ; EDI = Directory Buffer Directory Entry Location
5792                              <1>       ; EAX = File size
5793 000098D1 0F835C010000        <1>       jnc   loc_load_and_run_file
5794                              <1>
5795 000098D7 3C02                <1>       cmp   al, 2 ; file not found
5796 000098D9 0F8539E2FFFF        <1>       jne   loc_run_cmd_failed
5797                              <1>
5798 000098DF 66A1[825F0100]      <1>       mov   ax, word [EXE_ID]
5799 000098E5 80FC2E              <1>       cmp   ah, '.' ; File name has extension sign
5800 000098E8 7424                <1>       je    short loc_run_check_auto_path
5801                              <1>
5802 000098EA 08C0                <1>       or    al, al
5803 000098EC 7520                <1>       jnz   short loc_run_check_auto_path
5804                              <1>
5805 000098EE 80FC08              <1>       cmp   ah, 8 ; count of file name chars
5806 000098F1 771B                <1>       ja    short loc_run_check_auto_path
5807                              <1>
5808                              <1> loc_run_change_file_ext_to_prg:
```

```
5809 000098F3 0FB6DC              <1>       movzx  ebx, ah ; count of file name chars
5810 000098F6 BE[AC5C0100]        <1>       mov    esi, FindFile_Name
5811 000098FB 01F3                <1>       add    ebx, esi
5812                              <1>       ; 07/05/2016
5813 000098FD C7032E505247        <1>       mov    dword [ebx],  '.PRG'
5814 00009903 66C705[825F0100]50- <1>       mov    word [EXE_ID], 'P.'
5814 0000990B 2E                  <1>
5815 0000990C EBBA                <1>       jmp    short loc_run_find_program_file_next
5816                              <1>
5817                              <1> loc_run_check_auto_path:
5818                              <1>       ; NOTE: /// 07/05/2016 ///
5819                              <1>       ; If the path is given, value of byte [Run_Manual_Path]
5820                              <1>       ; will not be ZERO. If so, file searching by using
5821                              <1>       ; Automatic Path (via 'PATH' environment variable)
5822                              <1>       ; will not be applicable, because the program file
5823                              <1>       ; is already/absolutely not found.
5824                              <1>
5825 0000990E A0[815F0100]        <1>       mov    al, [Run_Manual_Path]
5826 00009913 08C0                <1>       or     al, al
5827 00009915 0F85D2E1FFFF        <1>       jnz    loc_cmd_failed
5828                              <1>
5829                              <1> loc_run_check_auto_path_again:
5830 0000991B 66833D[805F0100]FF  <1>       cmp    word [Run_Auto_Path], 0FFFFh
5831                              <1>              ; 0FFFFh = Not a valid run path (in ENV block)
5832 00009923 0F83C4E1FFFF        <1>       jnb    loc_cmd_failed
5833                              <1>       ; xor  al, al
5834 00009929 BE[79070100]        <1>       mov    esi, Cmd_Path ; 'PATH'
5835 0000992E BF[C6530100]        <1>       mov    edi, TextBuffer
5836 00009933 E857F9FFFF          <1>       call   get_environment_string
5837 00009938 730E                <1>       jnc    short loc_run_chk_filename_ext_again
5838 0000993A 66C705[805F0100]FF- <1>       mov    word [Run_Auto_Path], 0FFFFh ; invalid
5838 00009942 FF                  <1>
5839 00009943 E9A5E1FFFF          <1>       jmp    loc_cmd_failed
5840                              <1>
5841                              <1> loc_run_chk_filename_ext_again:
5842 00009948 89C1                <1>       mov    ecx, eax ; string length (with zero tail)
5843 0000994A 49                  <1>       dec    ecx ; without zero tail
5844 0000994B 66A1[825F0100]      <1>       mov    ax, [EXE_ID]
5845 00009951 80FC2E              <1>       cmp    ah, '.'
5846 00009954 740E                <1>       je     short loc_run_chk_auto_path_pos
5847                              <1>
5848                              <1> loc_run_change_file_ext_to_noext_again:
5849 00009956 0FB6DC              <1>       movzx  ebx, ah
5850 00009959 BE[AC5C0100]        <1>       mov    esi, FindFile_Name
5851 0000995E 01F3                <1>       add    ebx, esi
5852 00009960 29C0                <1>       sub    eax, eax
5853 00009962 8903                <1>       mov    [ebx], eax ; 0 ; erase extension (.PRG)
5854                              <1>
5855                              <1> loc_run_chk_auto_path_pos:
5856                              <1>       ;movzx eax,  word [Run_Auto_Path]
5857 00009964 66A1[805F0100]      <1>       mov    ax, [Run_Auto_Path]
5858 0000996A 39C8                <1>       cmp    eax, ecx ; ecx = string length (except zero tail)
5859 0000996C 0F837BE1FFFF        <1>       jnb    loc_cmd_failed
5860                              <1>       ;or    eax, eax
5861 00009972 6609C0              <1>       or     ax, ax
5862 00009975 7502                <1>       jnz    short loc_run_auto_path_pos_move
5863 00009977 B005                <1>       mov    al, 5
5864                              <1>
5865                              <1> loc_run_auto_path_pos_move:
5866 00009979 89FE                <1>       mov    esi, edi ; offset TextBuffer
5867 0000997B 01C6                <1>       add    esi, eax
5868                              <1>
5869                              <1> loc_run_auto_path_pos_space_loop:
5870 0000997D AC                  <1>       lodsb
5871 0000997E 3C20                <1>       cmp    al, 20h
5872 00009980 74FB                <1>       je     short loc_run_auto_path_pos_space_loop
5873 00009982 0F8265E1FFFF        <1>       jb     loc_cmd_failed
5874 00009988 AA                  <1>       stosb
5875                              <1> loc_run_auto_path_pos_move_next:
5876 00009989 AC                  <1>       lodsb
5877 0000998A 3C3B                <1>       cmp    al, ';'
5878 0000998C 7414                <1>       je     short loc_run_auto_path_pos_move_last_byte
5879 0000998E 3C20                <1>       cmp    al, 20h
5880 00009990 74F7                <1>       je     short loc_run_auto_path_pos_move_next
5881 00009992 7203                <1>       jb     short loc_byte_ptr_end_of_path
5882 00009994 AA                  <1>       stosb
5883 00009995 EBF2                <1>       jmp    short loc_run_auto_path_pos_move_next
5884                              <1>
5885                              <1> loc_byte_ptr_end_of_path:
5886 00009997 66C705[805F0100]FF- <1>       mov    word [Run_Auto_Path], 0FFFFh ; end of path
5886 0000999F FF                  <1>
5887 000099A0 EB0D                <1>       jmp    short loc_run_auto_path_move_ok
5888                              <1>
5889                              <1> loc_run_auto_path_pos_move_last_byte:
5890 000099A2 89F0                <1>       mov    eax, esi
5891 000099A4 2D[C6530100]        <1>       sub    eax, TextBuffer
5892 000099A9 66A3[805F0100]      <1>       mov    [Run_Auto_Path], ax ; next path position
5893                              <1>
5894                              <1> loc_run_auto_path_move_ok:
5895 000099AF 4F                  <1>       dec    edi
5896 000099B0 B02F                <1>       mov    al, '/'
5897 000099B2 3807                <1>       cmp    [edi], al
5898 000099B4 7403                <1>       je     short loc_run_auto_path_move_file_name
5899 000099B6 47                  <1>       inc    edi
5900 000099B7 8807                <1>       mov    [edi], al
5901                              <1>
5902                              <1> loc_run_auto_path_move_file_name:
5903 000099B9 47                  <1>       inc    edi
5904 000099BA BE[AC5C0100]        <1>       mov    esi, FindFile_Name
5905                              <1>
5906                              <1> loc_run_auto_path_move_fn_loop:
5907 000099BF AC                  <1>       lodsb
```

```
5908 000099C0 AA                <1>       stosb
5909 000099C1 08C0              <1>       or    al, al
5910 000099C3 75FA              <1>       jnz   short loc_run_auto_path_move_fn_loop
5911                            <1>
5912 000099C5 BE[C6530100]      <1>       mov   esi, TextBuffer
5913 000099CA BF[6A5C0100]      <1>       mov   edi, FindFile_Drv
5914 000099CF E832080000        <1>       call  parse_path_name
5915 000099D4 0F8213E1FFFF      <1>       jc    loc_cmd_failed
5916                            <1>
5917 000099DA 8A35[C6520100]    <1>       mov   dh, [Current_Drv]
5918 000099E0 8A15[6A5C0100]    <1>       mov   dl, [FindFile_Drv]
5919 000099E6 38F2              <1>       cmp   dl, dh
5920 000099E8 740B              <1>       je    short loc_run_change_directory_again
5921                            <1>
5922 000099EA E86ED2FFFF        <1>       call  change_current_drive
5923 000099EF 0F8223E1FFFF      <1>       jc    loc_run_cmd_failed
5924                            <1>
5925                            <1> loc_run_change_directory_again:
5926 000099F5 803D[6B5C0100]20  <1>       cmp   byte [FindFile_Directory], 20h
5927 000099FC 761D              <1>       jna   short loc_load_executable_cdir_chk_again
5928                            <1>
5929 000099FE FE05[AD060100]    <1>       inc   byte [Restore_CDIR]
5930 00009A04 BE[6B5C0100]      <1>       mov   esi, FindFile_Directory
5931 00009A09 30E4              <1>       xor   ah, ah ; CD_COMMAND sign -> 0
5932 00009A0B E8E2010000        <1>       call  change_current_directory
5933 00009A10 0F8202E1FFFF      <1>       jc    loc_run_cmd_failed
5934                            <1>
5935                            <1> loc_run_chg_prompt_dir_str_again:
5936 00009A16 E8F7000000        <1>       call  change_prompt_dir_string
5937                            <1>
5938                            <1> loc_load_executable_cdir_chk_again:
5939 00009A1B A1[C0520100]      <1>       mov   eax, [Current_Dir_FCluster]
5940 00009A20 3B05[7C5F0100]    <1>       cmp   eax, [Run_CDirFC]
5941 00009A26 0F8597FEFFFF      <1>       jne   loc_run_find_executable_file_next
5942 00009A2C 30C0              <1>       xor   al, al ; 0
5943 00009A2E E9E8FEFFFF        <1>       jmp   loc_run_check_auto_path_again
5944                            <1>
5945                            <1> loc_load_and_run_file:
5946                            <1>       ; 04/01/2017
5947                            <1>       ; 23/04/2016
5948 00009A33 BE[AC5C0100]      <1>       mov   esi, FindFile_Name
5949 00009A38 BF[C6530100]      <1>       mov   edi, TextBuffer
5950                            <1>
5951                            <1>       ; 24/04/2016
5952 00009A3D 31D2              <1>       xor   edx, edx
5953 00009A3F 668915[4A040300]  <1>       mov   word [argc], dx ; 0
5954 00009A46 8915[8C030300]    <1>       mov   dword [u.nread], edx ; 0
5955                            <1>
5956                            <1> loc_load_and_run_file_1:
5957 00009A4C AC                <1>       lodsb
5958 00009A4D AA                <1>       stosb
5959 00009A4E FF05[8C030300]    <1>       inc   dword [u.nread]
5960 00009A54 20C0              <1>       and   al, al
5961 00009A56 75F4              <1>       jnz   short loc_load_and_run_file_1
5962                            <1>
5963 00009A58 A0[27530100]      <1>       mov   al, [CmdArgStart]
5964 00009A5D 20C0              <1>       and   al, al
5965 00009A5F 7445              <1>       jz    short loc_load_and_run_file_7
5966                            <1>
5967 00009A61 0FB6F0            <1>       movzx esi, al ; 11/05/2016
5968 00009A64 B950000000        <1>       mov   ecx, 80
5969 00009A69 29F1              <1>       sub   ecx, esi
5970 00009A6B 81C6[76530100]    <1>       add   esi, CommandBuffer
5971                            <1>
5972 00009A71 66FF05[4A040300]  <1>       inc   word [argc] ; 11/05/2016
5973                            <1>
5974                            <1> loc_load_and_run_file_2:
5975 00009A78 AC                <1>       lodsb
5976 00009A79 3C20              <1>       cmp   al, 20h
5977 00009A7B 7717              <1>       ja    short loc_load_and_run_file_5
5978 00009A7D 721E              <1>       jb    short loc_load_and_run_file_6
5979                            <1>
5980                            <1> loc_load_and_run_file_3:
5981 00009A7F 803E20            <1>       cmp   byte [esi], 20h
5982 00009A82 7707              <1>       ja    short loc_load_and_run_file_4
5983 00009A84 7217              <1>       jb    short loc_load_and_run_file_6
5984 00009A86 46                <1>       inc   esi
5985 00009A87 E2F6              <1>       loop  loc_load_and_run_file_3
5986 00009A89 EB12              <1>       jmp   short loc_load_and_run_file_6
5987                            <1>
5988                            <1> loc_load_and_run_file_4:
5989 00009A8B 28C0              <1>       sub   al, al ; 0
5990 00009A8D 66FF05[4A040300]  <1>       inc   word [argc]
5991                            <1> loc_load_and_run_file_5:
5992 00009A94 AA                <1>       stosb
5993 00009A95 FF05[8C030300]    <1>       inc   dword [u.nread]
5994 00009A9B E2DB              <1>       loop  loc_load_and_run_file_2
5995                            <1>
5996                            <1> loc_load_and_run_file_6:
5997 00009A9D 30C0              <1>       xor   al, al ; 0
5998 00009A9F AA                <1>       stosb
5999 00009AA0 FF05[8C030300]    <1>       inc   dword [u.nread]
6000                            <1> loc_load_and_run_file_7:
6001 00009AA6 8807              <1>       mov   [edi], al ; 0
6002 00009AA8 66FF05[4A040300]  <1>       inc   word [argc] ; 24/04/2016
6003 00009AAF FF05[8C030300]    <1>       inc   dword [u.nread] ; 24/04/2016
6004 00009AB5 BE[C6530100]      <1>       mov   esi, TextBuffer
6005 00009ABA 8B15[D85C0100]    <1>       mov   edx, [FindFile_DirEntry+DirEntry_FileSize]
6006 00009AC0 66A1[D05C0100]    <1>       mov   ax, [FindFile_DirEntry+DirEntry_FstClusHI]
6007 00009AC6 66C1E010          <1>       shl   ax, 16
6008 00009ACA 66A1[D65C0100]    <1>       mov   ax, [FindFile_DirEntry+DirEntry_FstClusLO]
6009                            <1>       ; EAX = First Cluster number
```

```
6010                                    <1>       ; EDX = File Size
6011                                    <1>       ; ESI = Argument list address
6012                                    <1>       ; [argc] = argument count
6013                                    <1>       ; [u.nread] = argument list length
6014 00009AD0 E8AE450000                <1>       call   load_and_run_file ; trdosk6.s
6015                                    <1>         ;jc  loc_run_cmd_failed ; 04/01/2017
6016                                    <1> loc_load_and_run_file_8: ; 06/05/2016
6017 00009AD5 E945E9FFFF                <1>       jmp    loc_file_rw_restore_retn
6018                                    <1>
6019                                    <1> check_prg_filename_ext:
6020                                    <1>       ; 23/04/2016 (TRDOS 386 = TRDOS v2.0)
6021                                    <1>       ; 10/09/2011
6022                                    <1>       ; (TRDOS v1, CMDINTR.ASM, 'proc_check_exe_filename_ext')
6023                                    <1>       ; 14/11/2009
6024                                    <1>       ; INPUT ->
6025                                    <1>       ;     ESI = Dot File Name
6026                                    <1>       ; OUTPUT ->
6027                                    <1>       ;     cf = 0 -> EXE_ID in AL
6028                                    <1>       ;     ESI = Last char + 1 position
6029                                    <1>       ;     cf = 1 -> Invalid executable file name
6030                                    <1>       ;     or no file name extension if AH<=8
6031                                    <1>       ;     AL = Last file name char
6032                                    <1>       ;     cf = 0 -> AL='P' (PRG), AL=0 (no extension)
6033                                    <1>       ;
6034                                    <1>       ; (Modified registers: EAX, ESI)
6035                                    <1>
6036 00009ADA 30E4                     <1>       xor    ah, ah
6037                                    <1> loc_run_check_filename_ext:
6038 00009ADC AC                        <1>       lodsb
6039 00009ADD 3C21                      <1>       cmp    al, 21h
6040 00009ADF 7229                      <1>       jb     short loc_check_exe_fn_retn
6041 00009AE1 FEC4                      <1>       inc    ah
6042 00009AE3 3C2E                      <1>       cmp    al, '.'
6043 00009AE5 75F5                      <1>       jne    short loc_run_check_filename_ext
6044                                    <1>
6045                                    <1> loc_run_check_filename_ext_dot:
6046 00009AE7 80FC02                    <1>       cmp    ah, 2 ; .??? is not valid
6047 00009AEA 88C4                      <1>       mov    ah, al ; '.'
6048 00009AEC 7219                      <1>       jb     short loc_check_prg_fn_retn
6049                                    <1>
6050                                    <1> loc_run_check_filename_ext_dot_ok:
6051 00009AEE AC                        <1>       lodsb
6052 00009AEF 24DF                      <1>       and    al, 0DFh
6053                                    <1>
6054                                    <1> loc_run_check_filename_ext_prg:
6055 00009AF1 3C50                      <1>       cmp    al, 'P'
6056 00009AF3 7212                      <1>       jb     short loc_check_prg_fn_retn
6057 00009AF5 7711                      <1>       ja     short loc_check_prg_fn_stc
6058 00009AF7 AC                        <1>       lodsb
6059 00009AF8 24DF                      <1>       and    al, 0DFh
6060 00009AFA 3C52                      <1>       cmp    al, 'R'
6061 00009AFC 750A                      <1>       jne    short loc_check_prg_fn_stc
6062 00009AFE AC                        <1>       lodsb
6063 00009AFF 24DF                      <1>       and    al, 0DFh
6064 00009B01 3C47                      <1>       cmp    al, 'G'
6065 00009B03 7503                      <1>       jne    short loc_check_prg_fn_stc
6066                                    <1>
6067 00009B05 B050                      <1>       mov    al, 'P'
6068                                    <1> loc_check_prg_fn_retn:
6069 00009B07 C3                        <1>       retn
6070                                    <1>
6071                                    <1> loc_check_prg_fn_stc:
6072 00009B08 F9                        <1>       stc
6073 00009B09 C3                        <1>       retn
6074                                    <1>
6075                                    <1> loc_check_exe_fn_retn:
6076 00009B0A 28C0                      <1>       sub    al, al ; 0
6077 00009B0C C3                        <1>       retn
6078                                    <1>
6079                                    <1> find_and_list_files:
6080 00009B0D C3                        <1>       retn
6081                                    <1> set_exec_arguments:
6082 00009B0E C3                        <1>       retn
6083                                    <1> delete_fs_directory:
6084 00009B0F 31C0                      <1>       xor eax, eax
6085 00009B11 C3                        <1>       retn
2307                                        %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: 17/10/2016
   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> ; 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
```

```
                                <1>
00009B12 BE[275B0100]           <1>       mov    esi, PATH_Array
                                <1> change_prompt_dir_str: ; 05/10/2016 (call from 'set_working_path')
00009B17 BF[CA520100]           <1>       mov    edi, Current_Directory
00009B1C 8A25[C4520100]         <1>       mov    ah, [Current_Dir_Level]
00009B22 E807000000             <1>       call   set_current_directory_string
00009B27 880D[25530100]         <1>       mov    [Current_Dir_StrLen], cl
                                <1>
00009B2D 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>
00009B2E 57                     <1>       push    edi
00009B2F 80FC00                 <1>       cmp     ah, 0
00009B32 7652                   <1>       jna    short pass_write_path
00009B34 83C610                 <1>       add    esi, 16
00009B37 89F3                   <1>       mov    ebx, esi
                                <1> loc_write_path:
00009B39 B908000000             <1>       mov    ecx, 8
                                <1> path_write_dirname1:
00009B3E AC                     <1>       lodsb
00009B3F 3C20                   <1>       cmp    al, 20h
00009B41 7612                   <1>       jna    short pass_write_dirname1
00009B43 AA                     <1>       stosb
00009B44 81FF[24530100]         <1>       cmp    edi, End_Of_Current_Dir_Str
00009B4A 733A                   <1>       jnb    short pass_write_path
00009B4C E2F0                   <1>       loop   path_write_dirname1
00009B4E 803E20                 <1>       cmp    byte [esi], 20h
00009B51 7624                   <1>       jna    short pass_write_dirname2
00009B53 EB0A                   <1>       jmp     short loc_put_dot_cont_ext
                                <1> pass_write_dirname1:
00009B55 89DE                   <1>       mov    esi, ebx
00009B57 83C608                 <1>       add    esi, 8
00009B5A 803E20                 <1>       cmp    byte [esi], 20h
00009B5D 7618                   <1>       jna    short pass_write_dirname2
                                <1> loc_put_dot_cont_ext:
00009B5F C6072E                 <1>       mov    byte [edi], "."
                                <1>       ;mov   ecx, 3
00009B62 B103                   <1>       mov    cl, 3
                                <1> loc_check_dir_name_ext:
00009B64 AC                     <1>       lodsb
00009B65 47                     <1>       inc    edi
00009B66 3C20                   <1>       cmp    al, 20h
00009B68 760D                   <1>       jna    short pass_write_dirname2
00009B6A 8807                   <1>       mov    [edi], al
00009B6C 81FF[24530100]         <1>       cmp    edi, End_Of_Current_Dir_Str
00009B72 7312                   <1>       jnb    short pass_write_path
00009B74 E2EE                   <1>       loop    loc_check_dir_name_ext
00009B76 47                     <1>       inc    edi
                                <1> pass_write_dirname2:
00009B77 FECC                   <1>       dec    ah
00009B79 740B                   <1>       jz      short pass_write_path
00009B7B 83C310                 <1>       add    ebx, 16
00009B7E 89DE                   <1>       mov    esi, ebx
00009B80 C6072F                 <1>       mov    byte [edi],"/"
00009B83 47                     <1>       inc    edi
00009B84 EBB3                   <1>       jmp    short loc_write_path
                                <1> pass_write_path:
00009B86 C60700                 <1>       mov    byte [edi], 0
00009B89 47                     <1>       inc    edi
00009B8A 89F9                   <1>       mov    ecx, edi
00009B8C 5F                     <1>       pop    edi
00009B8D 29F9                   <1>       sub    ecx, edi
                                <1>       ; ECX = Current Directory String Length
00009B8F 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:
00009B90 80FA00                 <1>       cmp    dl, 0
00009B93 7708                   <1>       ja     short loc_get_current_drive_1
00009B95 8A15[C6520100]         <1>       mov    dl, [Current_Drv]
00009B9B EB17                   <1>       jmp    short loc_get_current_drive_2
```

```
                                <1> loc_get_current_drive_1:
00009B9D FECA                   <1>       dec    dl
00009B9F 3A15[AC060100]         <1>       cmp    dl, [Last_DOS_DiskNo]
00009BA5 760D                   <1>       jna    short loc_get_current_drive_2
00009BA7 B80F000000             <1>       mov    eax, 0Fh ; Invalid drive (Drive not ready!)
00009BAC F5                     <1>       cmc    ; stc
00009BAD C3                     <1>       retn
                                <1>
                                <1> loc_get_current_drive_not_ready_retn:
00009BAE 5E                     <1>       pop    esi
                                <1>       ;mov   eax, 15
00009BAF 66B80F00               <1>       mov    ax, 15 ; Drive not ready
00009BB3 C3                     <1>       retn
                                <1>
                                <1> loc_get_current_drive_2:
00009BB4 31C0                   <1>       xor    eax, eax
00009BB6 88D4                   <1>       mov    ah, dl
00009BB8 56                     <1>       push   esi
00009BB9 BE00010900             <1>       mov    esi, Logical_DOSDisks
00009BBE 01C6                   <1>       add    esi, eax
00009BC0 8A06                   <1>       mov    al, [esi+LD_Name]
00009BC2 3C41                   <1>       cmp    al, 'A'
00009BC4 72E8                   <1>       jb     short loc_get_current_drive_not_ready_retn
                                <1>
00009BC6 8A667F                 <1>       mov    ah, [esi+LD_CDirLevel]
00009BC9 08E4                   <1>       or     ah, ah
00009BCB 7506                   <1>       jnz    short loc_get_current_drive_3
                                <1>
                                <1>       ;xor   ah, ah ; mov ah, 0
00009BCD 8826                   <1>       mov    [esi], ah
00009BCF 31C9                   <1>       xor    ecx, ecx
00009BD1 EB1C                   <1>       jmp    short loc_get_current_drive_4
                                <1>
                                <1> loc_get_current_drive_3:
00009BD3 BF[275B0100]           <1>         mov     edi, PATH_Array
00009BD8 57                     <1>       push   edi
00009BD9 81C680000000           <1>       add    esi, LD_CurrentDirectory
00009BDF B920000000             <1>       mov    ecx, 32
00009BE4 F3A5                   <1>       rep    movsd
00009BE6 5E                     <1>       pop    esi ; Path Array Address
00009BE7 5F                     <1>       pop    edi ; pushed esi (current dir buffer offset)
                                <1>       ;
00009BE8 E841FFFFFF             <1>       call   set_current_directory_string
00009BED 89FE                   <1>       mov    esi, edi
                                <1>
                                <1> loc_get_current_drive_4:
00009BEF 30C0                   <1>       xor    al, al
00009BF1 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 -> succesful
                                <1>       ;        ESI = PATH_Array
                                <1>       ;        EAX = Current Directory First Cluster
                                <1>       ;
                                <1>       ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)
                                <1>
00009BF2 8825[B55B0100]         <1>       mov    [CD_COMMAND], ah
00009BF8 803E2F                 <1>       cmp    byte [esi], '/'
00009BFB 7505                   <1>       jne    short loc_ccd_cdir_level
00009BFD 46                     <1>       inc    esi
00009BFE 30C0                   <1>       xor    al, al
00009C00 EB05                   <1>       jmp    short loc_ccd_parse_path_name
                                <1> loc_ccd_cdir_level:
00009C02 A0[C4520100]           <1>       mov    al, [Current_Dir_Level]
                                <1> loc_ccd_parse_path_name:
00009C07 88C4                   <1>       mov    ah, al
00009C09 BF[275B0100]           <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>
00009C0E 0FB6C8                 <1>       movzx  ecx, al
00009C11 FEC1                   <1>       inc    cl
00009C13 C0E104                 <1>       shl    cl, 4
00009C16 01CF                   <1>       add    edi, ecx
00009C18 B107                   <1>       mov    cl, 7
00009C1A 28C1                   <1>       sub    cl, al
00009C1C C0E102                 <1>       shl    cl, 2
00009C1F 89C3                   <1>       mov    ebx, eax
00009C21 31C0                   <1>       xor    eax, eax ; 0
00009C23 F3AB                   <1>       rep    stosd
```

```
00009C25 89D8                <1>       mov    eax, ebx
                             <1>
00009C27 BF[275B0100]        <1>       mov    edi, PATH_Array
                             <1>
00009C2C 803E20              <1>       cmp    byte [esi], 20h
00009C2F F5                  <1>       cmc
00009C30 7305                <1>       jnc    short pass_ccd_parse_dir_name
                             <1>
                             <1>              ; ESI = Path name
                             <1>              ; AL = CCD_Level
00009C32 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:
00009C37 9C                  <1>       pushf
                             <1>
                             <1>       ;mov   [CCD_Level], al
                             <1>         ;mov [Last_Dir_Level], ah
00009C38 66A3[AB5B0100]      <1>       mov    [CCD_Level], ax
                             <1>
00009C3E 31DB                <1>       xor    ebx, ebx
00009C40 8A3D[C6520100]      <1>       mov    bh, [Current_Drv]
00009C46 BE00010900          <1>       mov    esi, Logical_DOSDisks
00009C4B 01DE                <1>       add    esi, ebx
                             <1>
00009C4D 9D                  <1>       popf
00009C4E 720A                <1>       jc     short loc_ccd_bad_path_name_retn
                             <1>
00009C50 8935[A75B0100]      <1>       mov    [CCD_DriveDT], esi
                             <1>
00009C56 3C07                <1>       cmp    al, 7
00009C58 7209                <1>       jb     short loc_ccd_load_child_dir
                             <1>
                             <1> loc_ccd_bad_path_name_retn:
00009C5A 87F7                <1>       xchg   esi, edi
00009C5C B813000000          <1>       mov    eax, 19 ; Bad directory/path name
00009C61 F9                  <1>       stc
                             <1> loc_ccd_retn_p:
00009C62 C3                  <1>       retn
                             <1>
                             <1> loc_ccd_load_child_dir:
                             <1>       ; AL = CCD_Level
00009C63 08C0                <1>       or     al, al
00009C65 7468                <1>       jz     short loc_ccd_load_root_dir
                             <1>
00009C67 6689C1              <1>       mov    cx, ax
00009C6A C0E004              <1>       shl    al, 4
00009C6D 0FB6F0              <1>       movzx  esi, al
00009C70 01FE                <1>       add    esi, edi  ; offset PATH_Array
                             <1>
00009C72 8B460C              <1>       mov    eax, [esi+12]
00009C75 38E9                <1>       cmp    cl, ch
00009C77 0F84FA000000        <1>         je       loc_ccd_load_sub_directory
00009C7D A3[C0520100]        <1>       mov    [Current_Dir_FCluster], eax
                             <1>
                             <1> loc_ccd_load_child_dir_next:
00009C82 83C610              <1>       add    esi, 16 ; DOS DirEntry Format FileName Address
                             <1>
                             <1>       ; Directory attribute : 10h
00009C85 B010                <1>       mov    al, 00010000b ; 10h (Attrib AND mask)
                             <1>       ;mov   ah, 11001000b ; C8h
                             <1>       ; Volume name attribute: 8h
00009C87 B408                <1>       mov    ah, 00001000b ; 08h (Attrib NAND, AND --> zero mask)
                             <1>
00009C89 6631C9              <1>       xor    cx, cx
00009C8C E8B5010000          <1>       call   locate_current_dir_file
00009C91 7353                <1>       jnc    short loc_ccd_set_dir_cluster_ptr
                             <1>
                             <1>        ; 19/02/2016
                             <1>       ;mov   edi, [CCD_DriveDT]
00009C93 8A25[AB5B0100]      <1>       mov    ah, [CCD_Level]
00009C99 803D[B55B0100]CD    <1>       cmp    byte [CD_COMMAND], 0CDh ;'CD' command or another
00009CA0 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.)
00009CA2 88E1                <1>       mov    cl, ah
00009CA4 50                  <1>       push   eax
00009CA5 E8E3000000          <1>       call   loc_ccd_save_current_dir
00009CAA 58                  <1>       pop    eax
                             <1> loc_ccd_load_child_dir_err:
00009CAB 3C03                <1>       cmp    al, 3  ; AL = 2 => File not found error
00009CAD 7202                <1>       jb     short loc_ccd_path_not_found_retn
00009CAF F9                  <1>       stc
00009CB0 C3                  <1>       retn
                             <1>
                             <1> loc_ccd_path_not_found_retn:
00009CB1 B003                <1>       mov    al, 3  ; Path not found
00009CB3 C3                  <1>       retn
                             <1>
                             <1> loc_ccd_load_FAT_root_dir:
00009CB4 803D[C5520100]02    <1>       cmp    byte [Current_FATType], 2
00009CBB 776B                <1>       ja     short loc_ccd_load_FAT32_root_dir
                             <1>
                             <1>       ;mov   esi, [CCD_DriveDT]
                             <1>       ;push  esi
00009CBD E8B61D0000          <1>       call   load_FAT_root_directory
                             <1>       ;pop   edi ; Dos Drv Description Table
                             <1>
```

```
00009CC2 89F7                <1>       mov    edi, esi
00009CC4 BE[275B0100]        <1>       mov    esi, PATH_Array
00009CC9 7297                <1>       jc     short loc_ccd_retn_p
                             <1>
00009CCB 31C0                <1>       xor    eax, eax
00009CCD EB78                <1>         jmp  short loc_ccd_set_cdfc
                             <1>
                             <1> loc_ccd_load_root_dir:
00009CCF 803D[C5520100]01    <1>       cmp    byte [Current_FATType], 1
00009CD6 73DC                <1>       jnb    short loc_ccd_load_FAT_root_dir
                             <1>
                             <1> loc_ccd_load_FS_root_dir:
00009CD8 E8621E0000          <1>       call   load_FS_root_directory
00009CDD EB5C                <1>       jmp    short pass_ccd_load_FAT_sub_directory
                             <1>
                             <1> loc_ccd_load_FS_sub_directory_next:
00009CDF E85C1E0000          <1>       call   load_FS_sub_directory
00009CE4 EB1F                <1>       jmp    short pass_ccd_set_dir_cluster_ptr
                             <1>
                             <1> loc_ccd_set_dir_cluster_ptr:
                             <1>       ; EDI = Directory Entry
00009CE6 668B4714            <1>       mov    ax, [edi+20] ; First Cluster High Word
00009CEA C1E010              <1>       shl    eax, 16
00009CED 668B471A            <1>       mov    ax, [edi+26] ; First Cluster Low Word
                             <1>
00009CF1 8B35[A75B0100]      <1>       mov    esi, [CCD_DriveDT]
00009CF7 803D[C5520100]01    <1>       cmp    byte [Current_FATType], 1
00009CFE 72DF                <1>       jb     short loc_ccd_load_FS_sub_directory_next
                             <1>       ;push  esi
00009D00 E8FE1D0000          <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
00009D05 BE[275B0100]        <1>       mov    esi, PATH_Array
00009D0A 7264                <1>       jc     short loc_ccd_retn_c
                             <1>
00009D0C A1[F55A0100]        <1>       mov    eax, [DirBuff_Cluster]
                             <1>
00009D11 FE05[AB5B0100]      <1>       inc    byte [CCD_Level]
00009D17 0FB61D[AB5B0100]    <1>       movzx  ebx, byte [CCD_Level]
00009D1E C0E304              <1>       shl    bl, 4 ; * 16 (<= 128)
00009D21 01DE                <1>       add    esi, ebx ; 19/02/2016
00009D23 89460C              <1>       mov    [esi+12], eax
00009D26 EB1F                <1>       jmp    short loc_ccd_set_cdfc
                             <1>
                             <1> loc_ccd_load_FAT32_root_dir:
00009D28 BE[275B0100]        <1>       mov    esi, PATH_Array
00009D2D 8B460C              <1>       mov    eax, [esi+12]
00009D30 8B35[A75B0100]      <1>       mov    esi, [CCD_DriveDT]
                             <1>
                             <1> loc_ccd_load_FAT_sub_directory:
                             <1>       ;push  esi
00009D36 E8C81D0000          <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
00009D3B BE[275B0100]        <1>       mov    esi, PATH_Array
00009D40 722E                <1>       jc     short loc_ccd_retn_c
                             <1>
00009D42 A1[F55A0100]        <1>       mov    eax, [DirBuff_Cluster]
                             <1>
                             <1> loc_ccd_set_cdfc:
00009D47 8A0D[AB5B0100]      <1>       mov    cl, [CCD_Level]
00009D4D 880D[C4520100]      <1>       mov    [Current_Dir_Level], cl
00009D53 A3[C0520100]        <1>       mov    [Current_Dir_FCluster], eax
                             <1>
00009D58 8A2D[AC5B0100]      <1>       mov    ch, [Last_Dir_Level]
00009D5E 38E9                <1>       cmp    cl, ch
00009D60 0F821CFFFFFF        <1>       jb     loc_ccd_load_child_dir_next
                             <1>
00009D66 803D[B55B0100]CD    <1>       cmp    byte [CD_COMMAND], 0CDh ;'CD' command or another
00009D6D 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>
00009D6F F8                  <1>       clc
                             <1>
                             <1> loc_ccd_retn_c:
00009D70 8B3D[A75B0100]      <1>       mov    edi, [CCD_DriveDT]
00009D76 C3                  <1>       retn
                             <1>
                             <1> loc_ccd_load_sub_directory:
00009D77 8B35[A75B0100]      <1>       mov    esi, [CCD_DriveDT]
00009D7D 803D[C5520100]01    <1>       cmp    byte [Current_FATType], 1
00009D84 73B0                <1>       jnb    short loc_ccd_load_FAT_sub_directory
00009D86 E8B51D0000          <1>       call   load_FS_sub_directory
00009D8B EBAE                <1>       jmp    short pass_ccd_load_FAT_sub_directory
                             <1>
                             <1> loc_ccd_save_current_dir:
00009D8D BE[275B0100]        <1>       mov    esi, PATH_Array ; 19/02/2016
00009D92 8B3D[A75B0100]      <1>       mov    edi, [CCD_DriveDT]
00009D98 57                  <1>       push   edi
00009D99 83C77F              <1>         add     edi, LD_CDirLevel
00009D9C 880F                <1>       mov    [edi], cl
00009D9E 47                  <1>       inc    edi ; LD_CurrentDirectory
00009D9F 56                  <1>       push   esi
                             <1>       ;mov   ecx, 32  ; always < 65536 (in this procedure)
00009DA0 66B92000            <1>       mov    cx, 32
00009DA4 F3A5                <1>       rep    movsd
```

```
                        <1>       ; Current directory has been saved to
                        <1>       ; the DOS drive description table, cdir area !
00009DA6 5E             <1>       pop    esi  ; PATH_Array
00009DA7 5F             <1>       pop    edi  ; Dos Drv Description Table
                        <1>
00009DA8 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
00009DA9 88C4           <1>       mov    ah, al
00009DAB 66A3[4C5C0100] <1>       mov    [PATH_CDLevel], ax
                        <1> repeat_ppdn_check_slash:
00009DB1 AC             <1>       lodsb
00009DB2 3C2F           <1>       cmp    al, '/'
00009DB4 74FB           <1>       je     short repeat_ppdn_check_slash
00009DB6 3C21           <1>       cmp    al, 21h
00009DB8 7219           <1>       jb     short loc_ppdn_retn
00009DBA 57             <1>       push   edi
                        <1> loc_ppdn_get_dir_name:
00009DBB B90C000000     <1>       mov    ecx, 12
00009DC0 BF[4E5C0100]   <1>       mov    edi, Dir_File_Name
                        <1> repeat_ppdn_get_dir_name:
00009DC5 AA             <1>       stosb
00009DC6 AC             <1>       lodsb
00009DC7 3C2F           <1>       cmp    al, '/'
00009DC9 740A           <1>       je     short loc_check_level_dot_conv_dir_name
00009DCB 3C20           <1>       cmp    al, 20h
00009DCD 7605           <1>       jna    short loc_ppdn_end_of_path_scan
00009DCF E2F4           <1>       loop   repeat_ppdn_get_dir_name
00009DD1 5F             <1>       pop    edi
00009DD2 F9             <1>       stc
                        <1> loc_ppdn_retn:
00009DD3 C3             <1>       retn
                        <1>
                        <1> loc_ppdn_end_of_path_scan:
00009DD4 4E             <1>       dec    esi
                        <1> loc_check_level_dot_conv_dir_name:
00009DD5 31C0           <1>       xor    eax, eax
00009DD7 AA             <1>       stosb
00009DD8 89F3           <1>       mov    ebx, esi
00009DDA BE[4E5C0100]   <1>       mov    esi, Dir_File_Name
00009DDF AC             <1>       lodsb
                        <1> repeat_ppdn_name_check_dot:
00009DE0 3C2E           <1>       cmp    al, '.'
00009DE2 7509           <1>       jne    short loc_ppdn_convert_sub_dir_name
                        <1> repeat_ppdn_name_dot_dot:
00009DE4 AC             <1>       lodsb
00009DE5 3C2E           <1>       cmp    al, '.'
00009DE7 743E           <1>       je     short loc_ppdn_dot_dot
00009DE9 3C21           <1>       cmp    al, 21h
00009DEB 7226           <1>       jb     short pass_ppdn_convert_sub_dir_name
                        <1> loc_ppdn_convert_sub_dir_name:
00009DED 8A25[4D5C0100] <1>       mov    ah, [PATH_Level]
00009DF3 80FC07         <1>       cmp    ah, 7
00009DF6 731B           <1>       jnb    short pass_ppdn_convert_sub_dir_name
00009DF8 FEC4           <1>       inc    ah
00009DFA 8825[4D5C0100] <1>       mov    [PATH_Level], ah
00009E00 BE[4E5C0100]   <1>       mov    esi, Dir_File_Name
                        <1>       ;mov   edi, [PATH_Array_Ptr]
00009E05 B010           <1>       mov    al, 16
00009E07 F6E4           <1>       mul    ah
00009E09 8B3C24         <1>       mov    edi, [esp]
                        <1>       ;push  edi
00009E0C 01C7           <1>       add    edi, eax
00009E0E E828030000     <1>       call   convert_file_name
                        <1>       ;pop   edi
                        <1> pass_ppdn_convert_sub_dir_name:
00009E13 89DE           <1>       mov    esi, ebx
                        <1> repeat_ppdn_check_last_slash:
00009E15 AC             <1>       lodsb
00009E16 3C2F           <1>       cmp    al, '/'
00009E18 74FB           <1>       je     short repeat_ppdn_check_last_slash
00009E1A 3C21           <1>       cmp    al, 21h
00009E1C 739D           <1>       jnb    short loc_ppdn_get_dir_name
                        <1> end_of_parse_dir_name:
00009E1E 5F             <1>       pop    edi
00009E1F F5             <1>       cmc
                        <1>       ;mov   al, [PATH_CDLevel]
                        <1>       ;mov   ah, [PATH_Level]
00009E20 66A1[4C5C0100] <1>       mov    ax, [PATH_CDLevel]
00009E26 C3             <1>       retn
                        <1>
                        <1> loc_ppdn_dot_dot:
00009E27 AC             <1>       lodsb
```

```
00009E28 3C21                <1>       cmp    al, 21h
00009E2A 73F2                <1>       jnb    short end_of_parse_dir_name
                             <1> loc_ppdn_dot_dot_prev_level:
00009E2C 66A1[4C5C0100]      <1>       mov    ax, [PATH_CDLevel]
00009E32 80EC01              <1>       sub    ah, 1
00009E35 80D400              <1>       adc    ah, 0
00009E38 38E0                <1>       cmp    al, ah
00009E3A 7602                <1>       jna    short pass_ppdn_set_al_to_ah
00009E3C 88E0                <1>       mov    al, ah
                             <1> pass_ppdn_set_al_to_ah:
00009E3E 66A3[4C5C0100]      <1>       mov    [PATH_CDLevel], ax
00009E44 EBCD                <1>       jmp    short pass_ppdn_convert_sub_dir_name
                             <1>
                             <1> locate_current_dir_file:
                             <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>
00009E46 8935[AF5B0100]      <1>       mov    [CDLF_FNAddress], esi
00009E4C 66A3[AD5B0100]      <1>       mov    [CDLF_AttributesMask], ax
00009E52 66890D[B35B0100]    <1>       mov    [CDLF_DEType], cx
                             <1>
00009E59 31DB                <1>       xor    ebx, ebx
00009E5B 881D[C45B0100]      <1>       mov    [PreviousAttr], bl ; 0  ; 13/02/2016
                             <1>
00009E61 8A3D[C6520100]      <1>       mov    bh, [Current_Drv]
00009E67 381D[F05A0100]      <1>       cmp    byte [DirBuff_ValidData], bl ; 0
00009E6D 761D                <1>       jna    short loc_lcdf_reload_current_dir2
00009E6F 8A1D[EE5A0100]      <1>         mov     bl, [DirBuff_DRV]
00009E75 80EB41              <1>       sub    bl, 'A'
00009E78 38DF                <1>       cmp    bh, bl
00009E7A 750E                <1>       jne    short loc_lcdf_reload_current_dir1
00009E7C 8B15[F55A0100]      <1>       mov    edx, [DirBuff_Cluster]
00009E82 3B15[C0520100]      <1>       cmp    edx, [Current_Dir_FCluster]
00009E88 7412                <1>       je     short loc_cdir_locatefile_search
                             <1>
                             <1> loc_lcdf_reload_current_dir1:
00009E8A 30DB                <1>       xor    bl, bl
                             <1> loc_lcdf_reload_current_dir2:
00009E8C 89DE                <1>       mov    esi, ebx
00009E8E 81C600010900        <1>         add     esi, Logical_DOSDisks
00009E94 E872000000          <1>       call   reload_current_directory
00009E99 735B                <1>       jnc    short loc_locatefile_search_again
00009E9B C3                  <1>       retn
                             <1>
                             <1> loc_cdir_locatefile_search:
00009E9C 31DB                <1>       xor    ebx, ebx
00009E9E E8A5000000          <1>       call   find_directory_entry
00009EA3 7349                <1>       jnc    short loc_cdir_locate_file_retn
                             <1>
                             <1> loc_locatefile_check_stc_reason:
00009EA5 08ED                <1>       or     ch, ch
00009EA7 7444                <1>       jz     short loc_cdir_locate_file_stc_retn
                             <1>
                             <1> loc_locatefile_check_next_entryblock:
00009EA9 8A3D[C6520100]      <1>       mov    bh, [Current_Drv]
00009EAF 28DB                <1>       sub    bl, bl
00009EB1 0FB7F3              <1>       movzx  esi, bx
00009EB4 81C600010900        <1>         add     esi, Logical_DOSDisks
                             <1>
00009EBA 803D[C4520100]00    <1>       cmp    byte [Current_Dir_Level], 0
00009EC1 760A                <1>       jna    short loc_locatefile_check_FAT_type
                             <1>
00009EC3 803D[C5520100]01    <1>       cmp    byte [Current_FATType], 1
00009ECA 730A                <1>       jnb    short loc_locatefile_load_subdir_cluster
```

```
00009ECC C3                  <1>       retn
                             <1>
                             <1> loc_locatefile_check_FAT_type:
00009ECD 803D[C5520100]03    <1>       cmp    byte [Current_FATType], 3
00009ED4 7218                <1>       jb     short loc_cdir_locate_file_retn
                             <1>
                             <1> loc_locatefile_load_subdir_cluster:
00009ED6 A1[F55A0100]        <1>       mov    eax, [DirBuff_Cluster]
00009EDB E83D1A0000          <1>       call   get_next_cluster
00009EE0 730D                <1>       jnc    short loc_locatefile_next_cluster
00009EE2 09C0                <1>       or     eax, eax
00009EE4 7507                <1>       jnz    short loc_locatefile_drive_not_ready_read_err
00009EE6 F9                  <1>       stc
                             <1> loc_locatefile_file_notfound:
00009EE7 B802000000          <1>       mov    eax, 2 ; File/Directory/VolName not found
00009EEC 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:
00009EED F5                  <1>       cmc ;stc
                             <1> loc_cdir_locate_file_retn:
00009EEE C3                  <1>       retn
                             <1>
                             <1> loc_locatefile_next_cluster:
00009EEF E80F1C0000          <1>       call   load_FAT_sub_directory
                             <1>       ;jc    short loc_locatefile_drive_not_ready_read_err
00009EF4 72F8                <1>       jc     short loc_cdir_locate_file_retn
                             <1>
                             <1> loc_locatefile_search_again:
00009EF6 8B35[AF5B0100]      <1>       mov    esi, [CDLF_FNAddress]
00009EFC 66A1[AD5B0100]      <1>       mov    ax, [CDLF_AttributesMask]
00009F02 668B0D[B35B0100]    <1>       mov    cx, [CDLF_DEType]
00009F09 EB91                <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]
00009F0B A0[C5520100]        <1>       mov    al, [Current_FATType]
00009F10 3C02                <1>       cmp    al, 2
00009F12 7729                <1>       ja     short loc_reload_FAT_sub_directory
00009F14 8A25[C4520100]      <1>       mov    ah, [Current_Dir_Level]
00009F1A 08C0                <1>       or     al, al
00009F1C 740A                <1>       jz     short loc_reload_FS_directory
00009F1E 08E4                <1>       or     ah, ah
00009F20 751B                <1>       jnz    short loc_reload_FAT_sub_directory
                             <1> loc_reload_FAT_12_16_root_directory:
00009F22 E8511B0000          <1>       call   load_FAT_root_directory
00009F27 C3                  <1>       retn
                             <1> loc_reload_FS_directory:
00009F28 20E4                <1>       and    ah, ah
00009F2A 7506                <1>       jnz    short loc_reload_FS_sub_directory
                             <1> loc_reload_FS_root_directory:
00009F2C E80E1C0000          <1>       call   load_FS_root_directory
00009F31 C3                  <1>       retn
                             <1> loc_reload_FS_sub_directory:
00009F32 A1[C0520100]        <1>       mov    eax, [Current_Dir_FCluster]
00009F37 E8041C0000          <1>       call   load_FS_sub_directory
00009F3C C3                  <1>       retn
                             <1> loc_reload_FAT_sub_directory:
00009F3D A1[C0520100]        <1>       mov    eax, [Current_Dir_FCluster]
00009F42 E8BC1B0000          <1>       call   load_FAT_sub_directory
00009F47 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>
00009F48 663B1D[F35A0100]       <1>       cmp   bx, [DirBuff_LastEntry]
00009F4F 0F8739010000           <1>         ja      loc_ffde_stc_retn_255
                                <1>
                                <1>       ;mov    [DirBuff_CurrentEntry], bx
                                <1>
00009F55 BF00000800             <1>       mov   edi, Directory_Buffer
00009F5A 66A3[C05B0100]         <1>       mov   [FDE_AttrMask], ax
                                <1>
00009F60 29C0                   <1>       sub   eax, eax
                                <1>
                                <1>       ;;mov  [PreviousAttr], al ; 0 ;; 13/02/2016
00009F62 66A3[C25B0100]         <1>       mov   [AmbiguousFileName], ax ; 0
                                <1>
00009F68 6689D8                 <1>       mov   ax, bx
00009F6B 66C1E005               <1>       shl   ax, 5 ; ; * 32 ; Directory entry size
00009F6F 01C7                   <1>       add     edi, eax
                                <1>
00009F71 08ED                   <1>       or    ch, ch
00009F73 0F852C010000           <1>         jnz     loc_find_free_deleted_entry_0
                                <1>
00009F79 08C9                   <1>       or      cl, cl
00009F7B 0F850D010000           <1>         jnz     loc_ffde_stc_retn_255
                                <1>
                                <1> check_find_dir_entry:
00009F81 66A1[C05B0100]         <1>       mov   ax, [FDE_AttrMask]
00009F87 8A2F                   <1>       mov   ch, [edi]
00009F89 80FD00                 <1>       cmp     ch, 0 ; Is it never used entry?
00009F8C 0F86FF000000           <1>       jna   loc_find_direntry_stc_retn
00009F92 56                     <1>       push  esi
00009F93 8A570B                 <1>       mov   dl, [edi+0Bh] ; File attributes
00009F96 80FDE5                 <1>       cmp   ch, 0E5h ; Is it a deleted file?
00009F99 746D                   <1>       je    short loc_find_dir_next_entry_prevdeleted
                                <1>
00009F9B 80FA0F                 <1>       cmp     dl, 0Fh ; longname sub component check
00009F9E 7505                   <1>       jne     short loc_check_attributes_mask
00009FA0 E8ED010000             <1>       call  save_longname_sub_component
                                <1>
                                <1> loc_check_attributes_mask:
00009FA5 88C6                   <1>       mov   dh, al
00009FA7 20D6                   <1>       and   dh, dl
00009FA9 38F0                   <1>       cmp   al, dh
00009FAB 0F85BA000000           <1>         jne     loc_find_dir_next_entry
00009FB1 20D4                   <1>       and   ah, dl
00009FB3 0F85B2000000           <1>         jnz     loc_find_dir_next_entry
00009FB9 80FA0F                 <1>       cmp   dl, 0Fh
00009FBC 751A                   <1>       jne   short pass_direntry_attr_check
                                <1>
00009FBE 3C0F                   <1>       cmp   al, 0Fh ; AL = 0Fh -> find long name
00009FC0 0F85A5000000           <1>         jne     loc_find_dir_next_entry
                                <1>
00009FC6 5E                     <1>       pop   esi
00009FC7 6631C0                 <1>       xor   ax, ax
00009FCA 8A35[C45B0100]         <1>       mov   dh, [PreviousAttr]
00009FD0 66891D[F15A0100]       <1>       mov   [DirBuff_CurrentEntry], bx
00009FD7 C3                     <1>       retn
                                <1>
                                <1> pass_direntry_attr_check:
00009FD8 89FD                   <1>       mov   ebp, edi ; 14/02/2016
00009FDA B908000000             <1>       mov   ecx, 8
                                <1> loc_lodsb_find_dir:
00009FDF AC                     <1>       lodsb
00009FE0 3C2A                   <1>       cmp   al, '*'
00009FE2 7508                   <1>       jne   short pass_fde_ambiguous1_check
00009FE4 FE05[C35B0100]         <1>         inc     byte [AmbiguousFileName+1]
00009FEA EB28                   <1>       jmp   short loc_check_direntry_extension
                                <1>
                                <1> pass_fde_ambiguous1_check:
00009FEC 3C3F                   <1>       cmp   al, '?'
00009FEE 750D                   <1>       jne   short pass_fde_ambiguous2_check
00009FF0 FE05[C25B0100]         <1>       inc   byte [AmbiguousFileName]
00009FF6 803F20                 <1>       cmp   byte [edi], 20h
00009FF9 764E                   <1>       jna   short loc_find_dir_next_entry_ebp
00009FFB EB14                   <1>       jmp   short loc_scasb_find_dir_inc_di
                                <1>
                                <1> pass_fde_ambiguous2_check:
00009FFD 3C20                   <1>       cmp   al, 20h
00009FFF 750C                   <1>       jne   short loc_scasb_find_dir
0000A001 803F20                 <1>       cmp   byte [edi], 20h
0000A004 7543                   <1>       jne   short loc_find_dir_next_entry_ebp
0000A006 EB0C                   <1>       jmp   short loc_check_direntry_extension
                                <1>
                                <1> loc_find_dir_next_entry_prevdeleted:
0000A008 80CA80                 <1>       or    dl, 80h  ; Bit 7 -> deleted entry sign
0000A00B EB5E                   <1>       jmp   short loc_find_dir_next_entry
                                <1>
                                <1> loc_scasb_find_dir:
0000A00D 3A07                   <1>       cmp   al, [edi]
0000A00F 7538                   <1>       jne   short loc_find_dir_next_entry_ebp
                                <1> loc_scasb_find_dir_inc_di:
0000A011 47                     <1>       inc   edi
0000A012 E2CB                   <1>       loop  loc_lodsb_find_dir
```

```
                            <1>
                            <1> loc_check_direntry_extension:
0000A014 BE08000000          <1>       mov    esi, 8
0000A019 89F7                <1>       mov    edi, esi ; 8
0000A01B 033424              <1>       add    esi, [esp] ; Sub Dir or File Name Address
0000A01E 01EF                <1>       add    edi, ebp
0000A020 B103                <1>       mov    cl, 3
                            <1> loc_lodsb_find_dir_ext:
0000A022 AC                  <1>       lodsb
0000A023 3C2A                <1>       cmp    al, '*'
0000A025 7508                <1>       jne    short pass_fde_ambiguous3_check
0000A027 FE05[C35B0100]      <1>       inc    byte [AmbiguousFileName+1]
0000A02D EB1E                <1>       jmp    short loc_find_dir_proper_direntry
                            <1>
                            <1> pass_fde_ambiguous3_check:
0000A02F 3C3F                <1>       cmp    al, '?'
0000A031 750D                <1>       jne    short pass_fde_ambiguous4_check
0000A033 FE05[C25B0100]      <1>       inc    byte [AmbiguousFileName]
0000A039 803F20              <1>       cmp    byte [edi], 20h
0000A03C 760B                <1>       jna    short loc_find_dir_next_entry_ebp
0000A03E EB49                <1>       jmp    short loc_scasb_find_dir_ext_inc_di
                            <1>
                            <1> pass_fde_ambiguous4_check:
0000A040 3C20                <1>       cmp    al, 20h
0000A042 7541                <1>       jne    short loc_scasb_find_dir_ext
0000A044 803F20              <1>       cmp    byte [edi], 20h
0000A047 7404                <1>       je     short loc_find_dir_proper_direntry
                            <1>
                            <1> loc_find_dir_next_entry_ebp:
0000A049 89EF                <1>       mov    edi, ebp ; 14/02/2016
0000A04B EB1E                <1>       jmp    short loc_find_dir_next_entry
                            <1>
                            <1> loc_find_dir_proper_direntry:
0000A04D 30C9                <1>       xor    cl, cl
                            <1> loc_find_dir_proper_direntry_1:
0000A04F 5E                  <1>       pop    esi
0000A050 89EF                <1>         mov     edi, ebp
0000A052 8A2F                <1>       mov    ch, [edi]
0000A054 8A570B              <1>       mov     dl, [edi+0Bh] ; Dir entry attributes
0000A057 66A1[C25B0100]      <1>       mov    ax, [AmbiguousFileName]
                            <1> loc_find_dir_proper_direntry_2:
0000A05D 8A35[C45B0100]      <1>       mov     dh, [PreviousAttr]
0000A063 66891D[F15A0100]    <1>       mov    [DirBuff_CurrentEntry], bx
0000A06A C3                  <1>       retn
                            <1>
                            <1> loc_find_dir_next_entry:
0000A06B 8815[C45B0100]      <1>       mov    byte [PreviousAttr], dl ; LongName check
                            <1> loc_find_dir_next_entry_1:
0000A071 5E                  <1>       pop    esi
0000A072 83C720              <1>       add    edi, 32
                            <1>       ;inc   word [DirBuff_EntryCounter]
0000A075 6643                <1>       inc    bx
0000A077 663B1D[F35A0100]    <1>       cmp    bx, [DirBuff_LastEntry]
0000A07E 770E                <1>       ja     short loc_ffde_stc_retn_255
0000A080 E9FCFEFFFF          <1>         jmp     check_find_dir_entry
                            <1>
                            <1> loc_scasb_find_dir_ext:
0000A085 3A07                <1>       cmp    al, [edi]
0000A087 75C0                <1>       jne    short loc_find_dir_next_entry_ebp
                            <1> loc_scasb_find_dir_ext_inc_di:
0000A089 47                  <1>       inc    edi
0000A08A E296                <1>       loop    loc_lodsb_find_dir_ext
0000A08C EBC1                <1>       jmp    short loc_find_dir_proper_direntry_1
                            <1>
                            <1> loc_ffde_stc_retn_255:
                            <1>       ;mov   cx, 0FFFFh
0000A08E 31C9                <1>       xor    ecx, ecx
0000A090 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
0000A091 B802000000          <1>       mov    eax, 2 ; File Not Found
0000A096 8A35[C45B0100]      <1>       mov    dh, [PreviousAttr]
0000A09C 66891D[F15A0100]    <1>       mov    [DirBuff_CurrentEntry], bx
0000A0A3 F9                  <1>       stc
0000A0A4 C3                  <1>       retn
                            <1>
                            <1> loc_find_free_deleted_entry_0:
0000A0A5 66A1[C05B0100]      <1>       mov    ax, [FDE_AttrMask]
0000A0AB 8A2F                <1>       mov    ch, [edi]
0000A0AD 8A570B              <1>       mov    dl, [edi+0Bh] ; File attributes
0000A0B0 08C9                <1>       or     cl, cl
0000A0B2 7407                <1>       jz     short loc_check_ffde_0_repeat
                            <1>       ;cmp   cl, 0E5h
                            <1>       ;je    short pass_loc_check_ffde_0_err
0000A0B4 80F9FF              <1>       cmp    cl, 0FFh
0000A0B7 7432                <1>       je     short loc_find_free_deleted_entry_1
0000A0B9 EB4D                <1>       jmp    short pass_loc_check_ffde_0_err
                            <1>
                            <1> loc_check_ffde_0_repeat:
0000A0BB 08ED                <1>       or     ch, ch
0000A0BD 7511                <1>       jnz    short loc_check_ffde_0_next
                            <1>
                            <1> loc_check_ffde_retn_2:
0000A0BF 6629C0              <1>       sub    ax, ax
0000A0C2 8A35[C45B0100]      <1>       mov    dh, [PreviousAttr]
0000A0C8 66891D[F15A0100]    <1>       mov    [DirBuff_CurrentEntry], bx
0000A0CF C3                  <1>       retn
                            <1>
                            <1> loc_check_ffde_0_next:
0000A0D0 6643                <1>       inc    bx
```

```
0000A0D2 83C720              <1>       add    edi, 32
                             <1>       ;inc   word [DirBuff_EntryCounter]
                             <1>
0000A0D5 663B1D[F35A0100]    <1>         cmp  bx, [DirBuff_LastEntry]
0000A0DC 77B0                <1>       ja     short loc_ffde_stc_retn_255
0000A0DE 8815[C45B0100]      <1>       mov    [PreviousAttr], dl
0000A0E4 8A2F                <1>       mov    ch, [edi]
0000A0E6 8A570B              <1>       mov    dl, [edi+0Bh] ; file attributes
0000A0E9 EBD0                <1>       jmp    short loc_check_ffde_0_repeat
                             <1>
                             <1> loc_find_free_deleted_entry_1:
0000A0EB 28D2                <1>       sub    dl, dl
                             <1> loc_find_free_deleted_entry_2:
0000A0ED 20ED                <1>       and    ch, ch
0000A0EF 74CE                <1>       jz     short loc_check_ffde_retn_2
0000A0F1 80FDE5              <1>       cmp    ch, 0E5h
0000A0F4 74C9                <1>       je     short loc_check_ffde_retn_2
0000A0F6 6643                <1>       inc    bx
0000A0F8 83C720              <1>       add    edi, 32
0000A0FB 663B1D[F35A0100]    <1>       cmp    bx, [DirBuff_LastEntry]
0000A102 778A                <1>       ja     short loc_ffde_stc_retn_255
0000A104 8A2F                <1>       mov    ch, [edi]
0000A106 EBE5                <1>       jmp    short loc_find_free_deleted_entry_2
                             <1>
                             <1> pass_loc_check_ffde_0_err:
0000A108 38CD                <1>       cmp    ch, cl
0000A10A 741F                <1>       je     short loc_check_ffde_attrib
                             <1>
0000A10C 6643                <1>       inc    bx
0000A10E 83C720              <1>       add    edi, 32
0000A111 663B1D[F35A0100]    <1>       cmp    bx, [DirBuff_LastEntry]
0000A118 0F8770FFFFFF        <1>         ja     loc_ffde_stc_retn_255
0000A11E 8815[C45B0100]      <1>       mov    [PreviousAttr], dl
0000A124 8A2F                <1>       mov    ch, [edi]
0000A126 8A570B              <1>       mov    dl, [edi+0Bh]
0000A129 EBDD                <1>       jmp    short pass_loc_check_ffde_0_err
                             <1>
                             <1> loc_check_ffde_attrib:
0000A12B 88C6                <1>       mov    dh, al
0000A12D 20D6                <1>       and    dh, dl
0000A12F 38F0                <1>       cmp    al, dh
0000A131 759D                <1>       jne    short loc_check_ffde_0_next
0000A133 20D4                <1>       and    ah, dl
0000A135 7599                <1>       jnz    short loc_check_ffde_0_next
0000A137 30C9                <1>       xor    cl, cl
0000A139 EB84                <1>         jmp     loc_check_ffde_retn_2
                             <1>
                             <1> convert_file_name:
                             <1>       ; 06/03/2016
                             <1>       ; 11/02/2016
                             <1>       ; 07/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 06/10/2009
                             <1>       ; 2005
                             <1>       ;
                             <1>       ; INPUT  ->
                             <1>       ;      ESI = Dot File Name Location
                             <1>       ;      EDI = Dir Entry Format File Name Location
                             <1>       ; OUTPUT ->
                             <1>       ;      EDI = Dir Entry Format File Name Location
                             <1>       ;      ESI = Dot File Name Location (capitalized)
                             <1>       ;
                             <1>       ; (ECX, AL will be changed)
                             <1>
0000A13B 56                  <1>       push   esi
0000A13C 57                  <1>       push   edi
                             <1>
0000A13D B90B000000          <1>       mov    ecx, 11
0000A142 B020                <1>       mov    al, 20h
0000A144 F3AA                <1>       rep    stosb
                             <1>
0000A146 8B3C24              <1>       mov    edi, [esp]
                             <1>
0000A149 B10C                <1>       mov    cl, 12 ; file name length (max.)
                             <1>       ; 06/03/2016
                             <1>       ; Directory entry name limit (11 bytes) check for
                             <1>       ; 'rename_directory_entry' procedure.
                             <1>       ; (EDI points to Directory Entry)
                             <1>       ; (If the file name would not contain a dot
                             <1>       ; and file name length would be 12, this would cause to
                             <1>       ; overwrite the attributes byte of the directory entry.)
                             <1>       ;
0000A14B B50B                <1>       mov    ch, 11 ; directory entry's name length
                             <1> loc_check_first_dot:
0000A14D 8A06                <1>       mov    al, [esi]
0000A14F 3C2E                <1>       cmp    al, 2Eh
0000A151 750C                <1>       jne    short pass_check_first_dot
0000A153 8807                <1>       mov    [edi], al
0000A155 47                  <1>       inc    edi
0000A156 46                  <1>       inc    esi
0000A157 FEC9                <1>       dec    cl
0000A159 75F2                <1>       jnz    short loc_check_first_dot
                             <1>       ;;(ecx <= 12)
                             <1>       ;;loop loc_check_first_dot
0000A15B EB30                <1>       jmp    short stop_convert_file
                             <1>
                             <1> loc_get_fchar:
0000A15D 8A06                <1>       mov    al, [esi]
                             <1> pass_check_first_dot:
0000A15F 3C61                <1>       cmp    al, 61h ; 'a'
0000A161 7208                <1>       jb     short pass_name_capitalize
0000A163 3C7A                <1>       cmp    al, 7Ah ; 'z'
0000A165 7704                <1>       ja     short pass_name_capitalize
```

```
1046 0000A167 24DF              <1>       and    al, 0DFh
1047 0000A169 8806              <1>       mov    [esi], al
1048                            <1> pass_name_capitalize:
1049 0000A16B 3C21              <1>       cmp    al, 21h
1050 0000A16D 721E              <1>       jb     short stop_convert_file
1051 0000A16F 3C2E              <1>       cmp    al, 2Eh ; '.'
1052 0000A171 750C              <1>       jne    short pass_dot_space
1053                            <1> add_dot_space:
1054 0000A173 80F904            <1>       cmp    cl, 4
1055 0000A176 760E              <1>       jna    short inc_and_loop
1056 0000A178 47                <1>       inc    edi
1057 0000A179 FECD              <1>       dec    ch ; 06/03/2016
1058 0000A17B FEC9              <1>       dec    cl
1059 0000A17D EBF4              <1>       jmp    short add_dot_space
1060                            <1>
1061                            <1>       ;mov   al, 4
1062                            <1>       ;cmp   cl, al
1063                            <1>       ;jna   short inc_and_loop
1064                            <1>       ;sub   cl, al
1065                            <1>       ;add   edi, ecx
1066                            <1>       ;mov   cl, al
1067                            <1>       ;jmp   short inc_and_loop
1068                            <1>
1069                            <1> pass_dot_space:
1070 0000A17F 8807              <1>       mov    [edi], al
1071                            <1> loc_after_double_dot:
1072                            <1>       ; 06/03/2016
1073 0000A181 FECD              <1>       dec    ch ; count down for 11 bytes dir entry limit
1074 0000A183 740A              <1>       jz     short stop_convert_file_x
1075 0000A185 47                <1>       inc    edi
1076                            <1> inc_and_loop:
1077 0000A186 FEC9              <1>       dec    cl ; count down for 12 bytes filename limit
1078 0000A188 7403              <1>       jz     short stop_convert_file
1079 0000A18A 46                <1>       inc    esi
1080                            <1>       ;;(ecx <= 12)
1081                            <1>       ;;loop loc_get_fchar
1082 0000A18B EBD0              <1>       jmp    short loc_get_fchar
1083                            <1>
1084                            <1> stop_convert_file:
1085                            <1>       ; 06/03/2016
1086 0000A18D 30ED              <1>       xor    ch, ch
1087                            <1>       ; ECX < 256 ; 'find_first_file' -> xor cl, cl
1088                            <1> stop_convert_file_x:
1089 0000A18F 5F                <1>       pop    edi
1090 0000A190 5E                <1>       pop    esi
1091 0000A191 C3                <1>       retn
1092                            <1>
1093                            <1> save_longname_sub_component:
1094                            <1>       ; 13/02/2016
1095                            <1>       ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)
1096                            <1>       ; 28/02/2010
1097                            <1>       ; 17/10/2009
1098                            <1>       ; INPUT ->
1099                            <1>       ;      EDI = Directory Entry
1100                            <1>       ;      // This procedure is called
1101                            <1>       ;      // from 'find_directory_entry' procedure.
1102                            <1>       ;      // If the last entry returns with
1103                            <1>       ;      // a non-zero LongnameFound value and
1104                            <1>       ;      // if LFN_CheckSum value is equal to
1105                            <1>       ;      // the next shortname checksum,
1106                            <1>       ;      // long name is valid.
1107                            <1>       ;      // If a longname is longer than 65 bytes,
1108                            <1>       ;      // it is invalid for trdos. (>45h)
1109                            <1>
1110 0000A192 57                <1>       push   edi
1111 0000A193 56                <1>       push   esi
1112                            <1>       ;push  ebx
1113                            <1>       ;push  ecx
1114                            <1>       ;push  edx
1115 0000A194 50                <1>       push   eax
1116                            <1>
1117 0000A195 29C9              <1>       sub    ecx, ecx
1118                            <1>       ;sub   eax, eax
1119 0000A197 B11A              <1>       mov    cl, 26
1120                            <1>
1121 0000A199 0FB607            <1>       movzx  eax, byte [edi] ; LDIR_Order
1122 0000A19C 3C41              <1>       cmp    al, 41h  ; 40h (last long entry sign) + 1
1123 0000A19E 722B              <1>       jb     short pass_pslnsc_last_long_entry
1124                            <1>
1125 0000A1A0 88C4              <1>       mov    ah, al
1126 0000A1A2 80EC40            <1>       sub    ah, 40h
1127 0000A1A5 8825[C65B0100]    <1>       mov    [LFN_EntryLength], ah
1128                            <1>
1129 0000A1AB 3C45              <1>       cmp    al, 45h  ; 40h (last long entry sign) + 5
1130                            <1>              ; Max 130 byte length is usable in TRDOS
1131                            <1> ; 26*5 = 130
1132 0000A1AD 7753              <1>       ja     short loc_pslnsc_retn
1133                            <1>
1134 0000A1AF 2407              <1>       and    al, 07h ; 0Fh
1135 0000A1B1 A2[C55B0100]      <1>       mov    [LongNameFound], al
1136                            <1>
1137 0000A1B6 FEC8              <1>       dec    al
1138                            <1>       ;mov   cl, 26
1139 0000A1B8 F6E1              <1>       mul    cl
1140                            <1>
1141 0000A1BA 89C6              <1>       mov    esi, eax
1142 0000A1BC 01CE              <1>       add    esi, ecx
1143                            <1>              ; to make is an ASCIZZ string
1144                            <1>              ; with ax+26 bytes length
1145 0000A1BE 81C6[C85B0100]    <1>       add    esi, LongFileName
1146 0000A1C4 66C7060000        <1>       mov    word [esi], 0
1147 0000A1C9 EB16              <1>       jmp    short loc_pslsc_move_ldir_name2
```

```
1148                                    <1>
1149                                    <1> pass_pslnsc_last_long_entry:
1150 0000A1CB 3C04                     <1>         cmp    al, 04h
1151 0000A1CD 7733                     <1>         ja     short loc_pslnsc_retn
1152 0000A1CF FE0D[C55B0100]           <1>         dec    byte [LongNameFound]
1153 0000A1D5 3A05[C55B0100]           <1>         cmp    al, [LongNameFound]
1154 0000A1DB 7525                     <1>         jne    short loc_pslnsc_retn
1155                                    <1>
1156                                    <1> loc_pslsc_move_ldir_name1:
1157 0000A1DD FEC8                     <1>         dec    al
1158                                    <1>         ;mov   cl, 26
1159 0000A1DF F6E1                     <1>         mul    cl
1160                                    <1>
1161                                    <1> loc_pslsc_move_ldir_name2:
1162 0000A1E1 8A4F0D                   <1>         mov    cl, [edi+0Dh] ; long name checksum
1163 0000A1E4 880D[C75B0100]           <1>         mov    [LFN_CheckSum], cl
1164 0000A1EA 89FE                     <1>         mov    esi, edi ; LDIR_Order
1165 0000A1EC BF[C85B0100]             <1>         mov    edi, LongFileName
1166 0000A1F1 01C7                     <1>         add    edi, eax
1167 0000A1F3 46                       <1>         inc    esi
1168 0000A1F4 B105                     <1>         mov    cl, 5 ; chars 1 to 5
1169 0000A1F6 F366A5                   <1>         rep    movsw
1170 0000A1F9 83C603                   <1>         add    esi, 3
1171 0000A1FC A5                       <1>         movsd  ; char 6 & 7
1172 0000A1FD A5                       <1>         movsd  ; char 8 & 9
1173 0000A1FE A5                       <1>         movsd  ; char 10 & 11
1174 0000A1FF 46                       <1>         inc    esi
1175 0000A200 46                       <1>         inc    esi
1176 0000A201 A5                       <1>         movsd   ; char 12 & 13
1177                                    <1>
1178                                    <1> loc_pslnsc_retn:
1179 0000A202 58                       <1>         pop    eax
1180                                    <1>         ;pop   edx
1181                                    <1>         ;pop   ecx
1182                                    <1>         ;pop   ebx
1183 0000A203 5E                       <1>         pop    esi
1184 0000A204 5F                       <1>         pop    edi
1185                                    <1>
1186 0000A205 C3                       <1>         retn
1187                                    <1>
1188                                    <1> parse_path_name:
1189                                    <1>         ; 10/02/2016
1190                                    <1>         ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)
1191                                    <1>         ; 10/009/2011 ('proc_parse_pathname')
1192                                    <1>         ; 27/11/2009
1193                                    <1>         ; 05/12/2004
1194                                    <1>         ;
1195                                    <1>         ; INPUT ->
1196                                    <1>         ;      ESI = Beginning of ASCIIZ pathname string
1197                                    <1>         ;       EDI = Destination Address
1198                                    <1>         ;            (which is TR-DOS FindFile data buffer)
1199                                    <1>         ; OUTPUT ->
1200                                    <1>         ;      CF = 1 -> Error
1201                                    <1>         ;           EAX = Error Code (AL)
1202                                    <1>         ;
1203                                    <1>         ; (Modified registers: eax, ecx, esi, edi)
1204                                    <1>
1205                                    <1>         ; Clear the pathname bytes in TR-DOS Findfile data buffer
1206 0000A206 57                       <1>         push   edi
1207 0000A207 B914000000               <1>         mov    ecx, 20  ; 80 bytes
1208 0000A20C 31C0                     <1>         xor    eax, eax
1209 0000A20E F3AB                     <1>         rep    stosd
1210 0000A210 5F                       <1>         pop    edi
1211                                    <1>
1212 0000A211 668B06                   <1>         mov    ax, [esi]
1213 0000A214 80FC3A                   <1>         cmp    ah, ':'
1214 0000A217 741C                     <1>         je     short loc_ppn_change_drive
1215 0000A219 A0[C6520100]             <1>         mov    al, [Current_Drv]
1216 0000A21E EB33                     <1>         jmp    short pass_ppn_change_drive
1217                                    <1>
1218                                    <1> pass_ppn_cdir:
1219 0000A220 8B35[EA5C0100]           <1>         mov    esi, [First_Path_Pos]
1220 0000A226 AC                       <1>         lodsb
1221                                    <1> loc_ppn_get_filename:
1222 0000A227 83C741                   <1>         add    edi, 65 ; FindFile_Name location
1223                                    <1>         ; TRDOS Filename length must not be more than 12 bytes
1224                                    <1>         ;mov   ecx, 12
1225 0000A22A B10C                     <1>         mov    cl, 12
1226                                    <1> loc_ppn_get_fnchar_next:
1227 0000A22C AA                       <1>         stosb
1228 0000A22D AC                       <1>         lodsb
1229 0000A22E 3C21                     <1>         cmp    al, 21h
1230 0000A230 7274                     <1>         jb     short loc_ppn_clc_return
1231 0000A232 E2F8                     <1>           loop    loc_ppn_get_fnchar_next
1232                                    <1> loc_ppn_return:
1233 0000A234 C3                       <1>         retn
1234                                    <1>
1235                                    <1> loc_ppn_change_drive:
1236 0000A235 24DF                     <1>         and    al, 0DFh
1237 0000A237 2C41                     <1>         sub    al, 'A'; A:
1238 0000A239 726F                     <1>         jc     short loc_ppn_invalid_drive
1239 0000A23B 3805[AC060100]           <1>         cmp    [Last_DOS_DiskNo], al
1240 0000A241 7267                     <1>         jb     short loc_ppn_invalid_drive
1241                                    <1>
1242 0000A243 46                       <1>         inc    esi
1243 0000A244 46                       <1>         inc    esi
1244 0000A245 8A26                     <1>         mov    ah, [esi]
1245 0000A247 80FC21                   <1>         cmp    ah, 21h
1246 0000A24A 7307                     <1>         jnb    short pass_ppn_change_drive
1247                                    <1>
1248                                    <1> loc_ppn_cmd_failed:
1249                                    <1>         ; File or directory name is not existing
```

```
1250 0000A24C 8807               <1>       mov   [edi], al ; Drv
1251 0000A24E 66B80100           <1>       mov   ax, 1 ; eax = 1
1252                             <1>       ; TR-DOS Error Code 01h = Bad Command Argument
1253                             <1>       ; MS-DOS Error Code 01h : Invalid Function Number
1254                             <1>       ;stc
1255                             <1>       ; (MainProg ErrMsg: "Bad command or file name!")
1256 0000A252 C3                 <1>       retn
1257                             <1>
1258                             <1> pass_ppn_change_drive:
1259 0000A253 8935[EA5C0100]     <1>       mov   [First_Path_Pos], esi
1260 0000A259 C705[EE5C0100]0000- <1>      mov   dword [Last_Slash_Pos], 0
1260 0000A261 0000               <1>
1261 0000A263 AA                 <1>       stosb
1262 0000A264 8A06               <1>       mov   al, [esi]
1263                             <1> loc_scan_ppn_dslash:
1264 0000A266 3C2F               <1>       cmp   al, '/'
1265 0000A268 7506               <1>       jne   short loc_scan_next_slash_pos
1266 0000A26A 8935[EE5C0100]     <1>       mov   [Last_Slash_Pos], esi
1267                             <1> loc_scan_next_slash_pos:
1268 0000A270 46                 <1>       inc   esi
1269 0000A271 8A06               <1>       mov   al, [esi]
1270 0000A273 3C20               <1>       cmp   al, 20h
1271 0000A275 77EF               <1>       ja    short loc_scan_ppn_dslash
1272 0000A277 833D[EE5C0100]00   <1>       cmp   dword [Last_Slash_Pos], 0
1273 0000A27E 76A0               <1>       jna   short pass_ppn_cdir
1274                             <1>
1275 0000A280 8B0D[EE5C0100]     <1>       mov   ecx, [Last_Slash_Pos]
1276 0000A286 8B35[EA5C0100]     <1>       mov   esi, [First_Path_Pos]
1277 0000A28C 29F1               <1>       sub   ecx, esi
1278 0000A28E 41                 <1>       inc   ecx
1279                             <1>       ;cmp  ecx, 64
1280 0000A28F 80F940             <1>       cmp   cl, 64
1281 0000A292 7715               <1>       ja    short loc_ppn_invalid_drive_stc
1282                             <1>
1283 0000A294 89F8               <1>       mov   eax, edi ; Dest Dir String Location (65 byte)
1284 0000A296 F3A4               <1>       rep   movsb
1285                             <1>       ;mov  [edi], cl ; 0, End of Dir String
1286 0000A298 8B35[EE5C0100]     <1>       mov   esi, [Last_Slash_Pos]
1287 0000A29E 46                 <1>       inc   esi
1288 0000A29F 89C7               <1>       mov   edi, eax
1289 0000A2A1 AC                 <1>       lodsb
1290 0000A2A2 3C21               <1>       cmp   al, 21h
1291 0000A2A4 7381               <1>       jnb   short loc_ppn_get_filename
1292                             <1> loc_ppn_clc_return:
1293                             <1>       ;clc
1294 0000A2A6 31C0               <1>       xor   eax, eax
1295 0000A2A8 C3                 <1>       retn
1296                             <1>
1297                             <1> loc_ppn_invalid_drive_stc:
1298 0000A2A9 F5                 <1>       cmc    ; stc
1299                             <1> loc_ppn_invalid_drive:
1300                             <1>       ; cf = 1
1301                             <1>       ; The Drive Letter/Char < "A" or > "Z"
1302 0000A2AA 66B80F00           <1>       mov   ax, 0Fh
1303                             <1>       ; MS-DOS Error Code 0Fh = Disk Drive Invalid
1304                             <1>       ; (MainProg ErrMsg: "Drive not ready or read error!")
1305 0000A2AE C3                 <1>       retn
1306                             <1>
1307                             <1> find_longname:
1308                             <1>       ; 13/02/2016 (TRDOS 386 = TRDOS v2.0)
1309                             <1>       ; 24/01/2010 (DIR.ASM, 'proc_find_longname')
1310                             <1>       ; 17/10/2009
1311                             <1>
1312                             <1>       ; INPUT ->
1313                             <1>       ;     ESI = DOS short file name address
1314                             <1>       ;     for example: "filename.ext"
1315                             <1>       ;
1316                             <1>       ; OUTPUT ->
1317                             <1>       ;     ESI = ASCIIZ longname address (cf = 0)
1318                             <1>       ;     cf = 1 -> error number returns in EAX (AL)
1319                             <1>       ;     AL = 0 & CF=1 -> longname not found
1320                             <1>       ;          the file/directory has no longname
1321                             <1>       ;     cf = 0 -> AL = FAT Type
1322                             <1>
1323                             <1>       ; 17/10/2009
1324                             <1>       ; ASCIIZ string will be returned
1325                             <1>       ; as LongFileName
1326                             <1>       ; clearing/reset is not needed
1327                             <1>       ;mov  ecx, 33
1328                             <1>       ;mov  edi, LongFileName
1329                             <1>       ;sub  ax, ax ; 0
1330                             <1>       ;rep  stosw
1331                             <1>
1332                             <1>       ;mov  byte [LongNameFound], 0
1333                             <1>
1334                             <1>       ; ESI = ASCIIZ file/directory name address
1335                             <1>       ;   AL = Attributes AND mask
1336                             <1>       ;      (Result of AND must be equal to AL)
1337                             <1>       ;   AH = Negative attributes mask
1338                             <1>       ;      (Result of AND must be ZERO)
1339 0000A2AF 66B80008           <1>       mov   ax, 0800h
1340                             <1>             ; it must not be volume name or longname
1341 0000A2B3 E837DDFFFF         <1>       call  find_first_file
1342 0000A2B8 7216               <1>       jc    short loc_fln_retn
1343                             <1>
1344                             <1> loc_fln_check_FAT_Type:
1345 0000A2BA 803D[C5520100]01   <1>       cmp   byte [Current_FATType], 1
1346 0000A2C1 7306               <1>       jnb   short loc_fln_check_longname_yes_sign
1347                             <1>
1348 0000A2C3 E839000000         <1>       call  get_fs_longname
1349 0000A2C8 C3                 <1>       retn
1350                             <1>
```

```
1351                                <1> loc_fln_check_longname_yes_sign:
1352 0000A2C9 08FF                 <1>       or     bh, bh
1353 0000A2CB 7504                 <1>       jnz    short loc_fln_check_longnamefound_number
1354                                <1> loc_fln_longname_not_found_retn:
1355 0000A2CD 31C0                 <1>       xor    eax, eax
1356                                <1>       ; cf = 1 & al = 0 -> longname not found
1357 0000A2CF F9                   <1>       stc
1358                                <1> loc_fln_retn:
1359 0000A2D0 C3                   <1>       retn
1360                                <1>
1361                                <1> loc_fln_check_longnamefound_number:
1362                                <1>       ; 'LongNameFound' is set by
1363                                <1>         ; by 'save_longname_sub_component'
1364                                <1>       ; which is called from
1365                                <1>       ; 'find_directory_entry'
1366                                <1>       ; which is called from
1367                                <1>       ; 'find_first_file'
1368                                <1>       ; It must 1 if the longname is valid
1369 0000A2D1 803D[C55B0100]01     <1>         cmp     byte [LongNameFound], 1
1370 0000A2D8 75F3                 <1>       jne    short loc_fln_longname_not_found_retn
1371                                <1>
1372                                <1> loc_fln_calculate_checksum:
1373 0000A2DA E813000000           <1>       call   calculate_checksum
1374                                <1>       ; AL = shortname checksum
1375                                <1>
1376                                <1> loc_fln_longname_validation:
1377                                <1>       ; 'LFN_CheckSum' has been set already
1378                                <1>       ; by 'save_longname_sub_component'
1379                                <1>       ; which is called from
1380                                <1>       ; 'find_directory_entry'
1381                                <1>       ; which is called from
1382                                <1>       ; 'find_first_file'
1383 0000A2DF 3805[C75B0100]       <1>       cmp    [LFN_CheckSum], al
1384 0000A2E5 75E6                 <1>       jne    short loc_fln_longname_not_found_retn
1385                                <1>
1386 0000A2E7 BE[C85B0100]         <1>       mov    esi, LongFileName
1387 0000A2EC A0[C5520100]         <1>       mov    al, [Current_FATType]
1388 0000A2F1 C3                   <1>       retn
1389                                <1>
1390                                <1> calculate_checksum:
1391                                <1>       ; 13/02/2016 (TRDOS 386 = TRDOS v2.0)
1392                                <1>       ; 17/10/2009 (DIR.ASM, 'proc_calculate_checksum')
1393                                <1>         ;
1394                                <1>       ; INPUT ->
1395                                <1>       ;     ESI = 11 byte DOS File Name location
1396                                <1>       ;     (in DOS Directory Entry Format)
1397                                <1>       ; OUTPUT ->
1398                                <1>       ;      AL = 8 bit checksum (CRC) value
1399                                <1>       ;
1400                                <1>       ; (Modified registers: EAX, ECX, ESI)
1401                                <1>
1402                                <1>       ; Erdogan Tan [ 17-10-2009 ]
1403                                <1>       ;  'ror al, 1' instruction
1404                                <1>
1405                                <1>       ; Erdogan Tan [ 20-06-2004 ]
1406                                <1>       ; This 8086 assembly code is an original code
1407                                <1>       ; which is adapted from C code in
1408                                <1>       ; Microsoft FAT32 File System Specification
1409                                <1>       ; Version 1.03, December 6, 2000
1410                                <1>       ; Page 28
1411                                <1>
1412 0000A2F2 30C0                 <1>       xor    al, al
1413 0000A2F4 B90B000000           <1>       mov    ecx, 11
1414                                <1> loc_next_sum:
1415                                <1>       ;xor   ah, ah
1416                                <1>       ;test  al, 1
1417                                <1>       ;jz    short pass_ah_80h
1418                                <1>       ;mov   ah, 80h
1419                                <1> ;pass_ah_80h:
1420                                <1>       ;shr   al, 1
1421 0000A2F9 D0C8                 <1>       ror    al, 1 ; 17/10/2009
1422 0000A2FB 0206                 <1>       add    al, [esi]
1423 0000A2FD 46                   <1>       inc    esi
1424                                <1>       ;add   al, ah
1425 0000A2FE E2F9                 <1>       loop   loc_next_sum
1426 0000A300 C3                   <1>       retn
1427                                <1>
1428                                <1> get_fs_longname:
1429                                <1>       ; temporary (13/02/2016)
1430 0000A301 31C0                 <1>       xor eax, eax
1431 0000A303 F9                   <1>       stc
1432 0000A304 C3                   <1>       retn
1433                                <1>
1434                                <1> make_sub_directory:
1435                                <1>       ; 16/10/2016
1436                                <1>       ; 02/03/2016, 03/03/2016
1437                                <1>       ; 26/02/2016, 27/02/2016
1438                                <1>       ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)
1439                                <1>       ; 01/08/2011 (DIR.ASM, 'proc_make_directory')
1440                                <1>       ; 10/07/2010
1441                                <1>       ; INPUT ->
1442                                <1>       ;     ESI = ASCIIZ Directory Name
1443                                <1>       ;     CL = Directory Attributes
1444                                <1>       ; OUTPUT ->
1445                                <1>       ;     EAX = New sub dir's first cluster
1446                                <1>       ;     ESI = Logical Dos Drv Descr. Table Addr.
1447                                <1>       ;     CF = 1 -> error code in AL (EAX)
1448                                <1>
1449                                <1>       ;test  cl, 10h  ; directory
1450                                <1>       ;jz    short loc_make_directory_access_denied
1451                                <1>       ;test  cl, 08h ; volume name
1452                                <1>       ;jnz   short loc_make_directory_access_denied
```

```
1453                                    <1>
1454 0000A305 80E107                   <1>       and    cl, 07h
1455 0000A308 880D[445D0100]            <1>       mov    byte [mkdir_attrib], cl
1456                                    <1>
1457 0000A30E 56                        <1>       push   esi
1458 0000A30F 31DB                      <1>       xor    ebx, ebx
1459 0000A311 8A3D[C6520100]            <1>       mov    bh, [Current_Drv]
1460 0000A317 BE00010900                <1>       mov    esi, Logical_DOSDisks
1461 0000A31C 01DE                      <1>       add    esi, ebx
1462 0000A31E 5B                        <1>       pop    ebx
1463                                    <1>
1464                                    <1>       ; 10/07/2010 -> 1st writable disk check for trdos
1465                                    <1>       ; LD_DiskType = 0 for write protection (read only)
1466 0000A31F 807E0101                  <1>       cmp    byte [esi+LD_DiskType], 1 ; 0 = Invalid
1467 0000A323 730B                      <1>       jnb    short loc_mkdir_check_file_sytem
1468                                    <1>       ; 16/10/2016 (13h -> 30)
1469 0000A325 B81E000000                <1>       mov    eax, 30 ; 'Disk write-protected' error
1470 0000A32A BA00000000                <1>       mov    edx, 0
1471                                    <1>       ; err retn: EDX = 0, EBX = Dir name offset
1472                                    <1>       ;ESI = Logical DOS drive description table address
1473 0000A32F C3                        <1>       retn
1474                                    <1>
1475                                    <1> ;loc_make_directory_access_denied:
1476                                    <1>       ;mov   ax, 05h ; access denied (invalid attributes input)
1477                                    <1>       ;stc
1478                                    <1>       ;retn
1479                                    <1>
1480                                    <1> loc_mkdir_check_file_sytem:
1481 0000A330 807E0301                  <1>       cmp    byte [esi+LD_FATType], 1
1482 0000A334 730B                      <1>       jnb    short loc_mkdir_check_free_sectors
1483                                    <1>
1484                                    <1> loc_make_fs_directory:
1485 0000A336 A1[C0520100]              <1>       mov    eax, [Current_Dir_FCluster]
1486                                    <1>       ; EAX = Parent directory DDT Address
1487                                    <1>       ; ESI = Logical DOS Drive DT Address
1488                                    <1>       ; EBX = Directory name offset (as ASCIIZ name)
1489 0000A33B E8D8150000                <1>       call   make_fs_directory
1490 0000A340 C3                        <1>       retn
1491                                    <1>
1492                                    <1> loc_mkdir_check_free_sectors:
1493 0000A341 0FB64613                  <1>         movzx   eax, byte [esi+LD_BPB+SecPerClust]
1494 0000A345 8B4E74                    <1>       mov    ecx, [esi+LD_FreeSectors]
1495 0000A348 39C1                      <1>       cmp    ecx, eax
1496 0000A34A 7255                      <1>       jb     short loc_mkdir_insufficient_disk_space
1497                                    <1>
1498                                    <1> loc_make_fat_directory:
1499 0000A34C 891D[345D0100]            <1>       mov    [mkdir_DirName_Offset], ebx
1500 0000A352 890D[405D0100]            <1>       mov    [mkdir_FreeSectors], ecx
1501                                    <1>
1502                                    <1>       ;mov   al, [esi+LD_BPB+SecPerClust]
1503 0000A358 A2[465D0100]              <1>       mov    byte [mkdir_SecPerClust], al
1504                                    <1>
1505                                    <1> loc_mkdir_gffc_1:
1506 0000A35D E812180000                <1>       call   get_first_free_cluster
1507 0000A362 722A                      <1>       jc     short loc_mkdir_gffc_retn
1508                                    <1>
1509                                    <1> ;loc_mkdir_gffc_1_cont:
1510                                    <1>       ;cmp   eax, 2
1511                                    <1>       ;jb    short loc_mkdir_gffc_insufficient_disk_space
1512                                    <1>
1513                                    <1> ;loc_mkdir_gffc_1_save_fcluster:
1514 0000A364 A3[385D0100]              <1>       mov    [mkdir_FFCluster], eax
1515                                    <1>
1516                                    <1> loc_mkdir_locate_ffe:
1517                                    <1>       ; Current directory fcluster <> Directory buffer cluster
1518                                    <1>       ; Current directory will be reloaded by
1519                                    <1>       ; 'locate_current_dir_file' procedure
1520                                    <1>       ;
1521                                    <1>       ; ESI = Logical DOS Drive Description Table Address
1522                                    <1>       ;push  esi ; 27/02/2016
1523 0000A369 31C0                      <1>       xor    eax, eax
1524 0000A36B 89C1                      <1>         mov  ecx, eax
1525 0000A36D 6649                      <1>       dec    cx ; FFFFh
1526                                    <1>       ; CX = FFFFh -> find first deleted or free entry
1527                                    <1>       ; ESI would be ASCIIZ filename address if the call
1528                                    <1>       ; would not be for first free or deleted dir entry
1529 0000A36F E8D2FAFFFF                <1>       call   locate_current_dir_file
1530 0000A374 734C                      <1>       jnc    short loc_mkdir_set_ff_dir_entry_1
1531                                    <1>       ;pop   esi
1532                                    <1>       ; ESI = Logical DOS Drive Description Table Address
1533 0000A376 83F802                    <1>       cmp    eax, 2  ; cmp al, 2 ; File/Dir not found !
1534 0000A379 752B                      <1>       jne    short loc_mkdir_stc_return
1535                                    <1>
1536                                    <1> loc_mkdir_add_new_cluster:
1537 0000A37B 3805[C5520100]            <1>       cmp    byte [Current_FATType], al ; 2
1538                                    <1>       ;cmp   byte ptr [esi+LD_FATType], 2
1539 0000A381 770C                      <1>       ja     short loc_mkdir_add_new_cluster_check_fsc
1540 0000A383 803D[C4520100]01          <1>       cmp    byte [Current_Dir_Level], 1
1541                                    <1>       ;cmp   byte [esi+LD_CDirLevel], 1
1542 0000A38A 7303                      <1>       jnb    short loc_mkdir_add_new_cluster_check_fsc
1543                                    <1>
1544 0000A38C B00C                      <1>       mov    al, 12 ; No more files
1545                                    <1> loc_mkdir_gffc_retn:
1546 0000A38E C3                        <1>       retn
1547                                    <1>
1548                                    <1> loc_mkdir_add_new_cluster_check_fsc:
1549 0000A38F 8B0D[405D0100]            <1>       mov    ecx, [mkdir_FreeSectors]
1550                                    <1>       ;movzx eax, byte [mkdir_SecPerClust]
1551 0000A395 A0[465D0100]              <1>       mov    al, [mkdir_SecPerClust]
1552 0000A39A 66D1E0                    <1>       shl    ax, 1 ; AX = 2 * AX
1553 0000A39D 39C1                      <1>       cmp    ecx, eax
1554 0000A39F 7350                      <1>       jnb    short loc_mkdir_add_new_subdir_cluster
```

```
1555                                    <1>
1556                                    <1> loc_mkdir_insufficient_disk_space:
1557                                    <1>       ;mov   edx, ecx
1558                                    <1> ;loc_mkdir_gffc_insufficient_disk_space:
1559 0000A3A1 66B82700                 <1>       mov    ax, 27h ; MSDOS err => insufficient disk space
1560                                    <1>       ; err retn: EDX = Free sectors, EBX = Dir name offset
1561                                    <1>         ; ESI -> Dos drive description table address
1562                                    <1>       ;; ecx = edx
1563                                    <1>       ;
1564 0000A3A5 C3                       <1>       retn
1565                                    <1>
1566                                    <1> loc_mkdir_stc_return:
1567 0000A3A6 F9                       <1>       stc
1568 0000A3A7 C3                       <1>       retn
1569                                    <1>
1570                                    <1> loc_mkdir_gffc_2:
1571 0000A3A8 E8C7170000               <1>       call   get_first_free_cluster
1572 0000A3AD 72DF                     <1>       jc     short loc_mkdir_gffc_retn
1573                                    <1>
1574                                    <1> ;loc_mkdir_gffc_1_cont:
1575                                    <1>       ;cmp   eax, 2
1576                                    <1>       ;jb    short loc_mkdir_gffc_insufficient_disk_space
1577                                    <1>
1578                                    <1> ;loc_mkdir_gffc_2_save_fcluster:
1579 0000A3AF A3[385D0100]               <1>       mov    [mkdir_FFCluster], eax
1580                                    <1>
1581 0000A3B4 A1[3C5D0100]             <1>       mov    eax, [mkdir_LastDirCluster]
1582                                    <1>
1583 0000A3B9 E845170000               <1>       call   load_FAT_sub_directory
1584 0000A3BE 72CE                     <1>       jc     short loc_mkdir_gffc_retn
1585                                    <1>
1586 0000A3C0 31FF                     <1>       xor    edi, edi
1587                                    <1> loc_mkdir_set_ff_dir_entry_1:
1588                                    <1>       ; 27/02/2016
1589 0000A3C2 56                       <1>       push   esi ; Logical DOS Drv Desc. Tbl. address
1590                                    <1>       ; EDI = Directory Entry Address
1591 0000A3C3 8B35[345D0100]           <1>       mov    esi, [mkdir_DirName_Offset]
1592 0000A3C9 A1[385D0100]             <1>       mov    eax, [mkdir_FFCluster]
1593                                    <1>
1594 0000A3CE 66B91000                 <1>       mov    cx, 10h      ; CL = Directory attribute
1595                                    <1>                     ; CH = 0 -> File size is 0
1596 0000A3D2 0A0D[445D0100]           <1>       or     cl, [mkdir_attrib] ; S, H, R
1597 0000A3D8 E8B0010000               <1>       call   make_directory_entry
1598                                    <1>
1599 0000A3DD 5E                       <1>       pop    esi
1600                                    <1>
1601 0000A3DE C605[F05A0100]02         <1>       mov    byte [DirBuff_ValidData], 2
1602 0000A3E5 E880020000               <1>       call   save_directory_buffer
1603 0000A3EA 0F83DA000000             <1>         jnc     loc_mkdir_set_ff_dir_entry_2
1604                                    <1>
1605                                    <1> loc_mkdir_return:
1606 0000A3F0 C3                       <1>       retn
1607                                    <1>
1608                                    <1> loc_mkdir_add_new_subdir_cluster:
1609 0000A3F1 8B15[F55A0100]           <1>       mov    edx, [DirBuff_Cluster]
1610 0000A3F7 8915[3C5D0100]           <1>       mov    [mkdir_LastDirCluster], edx
1611                                    <1>
1612 0000A3FD A1[385D0100]             <1>       mov    eax, [mkdir_FFCluster]
1613 0000A402 E8FC160000               <1>       call   load_FAT_sub_directory
1614 0000A407 72E7                     <1>       jc     short loc_mkdir_return
1615                                    <1>       ; eax = 0
1616                                    <1>       ; ecx =  directory buffer sector count (<= 128)
1617                                    <1>
1618                                    <1> pass_mkdir_add_new_subdir_cluster:
1619 0000A409 29FF                     <1>       sub    edi, edi ; 0
1620                                    <1>       ;mov   al, 128 ; double word
1621                                    <1>       ;mul   ecx ; ecx =  directory buffer sector count
1622                                    <1>       ;mov   ecx, eax
1623                                    <1>       ;shl   cx, 7 ; 128 * sector count
1624 0000A40B 668B4611                 <1>       mov    ax, [esi+LD_BPB+BytesPerSec] ; 512
1625 0000A40F 66C1E802                 <1>       shr    ax, 2 ; 'byte count / 4' for 'stosd'
1626 0000A413 66F7E1                   <1>       mul    cx ; max = 128*(512/4) -> 16384 (stosd)
1627 0000A416 6689C1                   <1>       mov    cx, ax
1628 0000A419 6629C0                   <1>       sub    ax, ax ; 0
1629 0000A41C F3AB                     <1>       rep    stosd ; clear directory buffer
1630                                    <1>
1631 0000A41E C605[F05A0100]02         <1>       mov    byte [DirBuff_ValidData], 2
1632 0000A425 E840020000               <1>       call   save_directory_buffer
1633 0000A42A 72C4                     <1>       jc     short loc_mkdir_return
1634                                    <1>
1635                                    <1> loc_mkdir_save_added_cluster:
1636 0000A42C A1[3C5D0100]             <1>       mov    eax, [mkdir_LastDirCluster]
1637 0000A431 8B0D[385D0100]           <1>       mov    ecx, [mkdir_FFCluster]
1638                                    <1>       ; 01/03/2016
1639 0000A437 31D2                     <1>       xor    edx, edx
1640 0000A439 8915[E65A0100]           <1>       mov    [FAT_ClusterCounter], edx ; 0 ; reset
1641 0000A43F E803180000               <1>       call   update_cluster
1642 0000A444 7304                     <1>       jnc    short loc_mkdir_save_fat_buffer_0
1643 0000A446 09C0                     <1>       or     eax, eax ; EAX = 0 -> cluster value is 0 or eocc
1644 0000A448 7518                     <1>       jnz    short loc_mkdir_save_fat_buffer_stc_retn
1645                                    <1>
1646                                    <1> loc_mkdir_save_fat_buffer_0:
1647 0000A44A A1[385D0100]             <1>       mov    eax, [mkdir_FFCluster]
1648 0000A44F A3[3C5D0100]             <1>       mov    [mkdir_LastDirCluster], eax
1649                                    <1>
1650 0000A454 31C9                     <1>       xor    ecx, ecx
1651 0000A456 49                       <1>       dec    ecx ; FFFFFFFFh
1652                                    <1>       ; ESI = Logical DOS Drive Description Table address
1653 0000A457 E8EB170000               <1>       call   update_cluster
1654 0000A45C 731A                     <1>       jnc    short loc_mkdir_save_fat_buffer_1
1655 0000A45E 09C0                     <1>       or     eax, eax
1656 0000A460 7416                     <1>       jz     short loc_mkdir_save_fat_buffer_1
```

```
1657                                <1>
1658                                <1> loc_mkdir_save_fat_buffer_stc_retn:
1659                                <1>       ; 01/03/2016
1660 0000A462 803D[E65A0100]01     <1>       cmp    byte [FAT_ClusterCounter], 1
1661 0000A469 720C                 <1>       jb     short loc_mkdir_save_fat_buffer_retn
1662                                <1>
1663 0000A46B 66BB00FF             <1>       mov    bx, 0FF00h ; recalculate free space (BL = 0)
1664                                <1>                         ; (BH = FFh -> Use ESI as Drv Param. Tbl.)
1665 0000A46F 50                   <1>       push   eax
1666 0000A470 E8241B0000           <1>       call   calculate_fat_freespace
1667 0000A475 58                   <1>       pop    eax
1668 0000A476 F9                   <1>       stc
1669                                <1> loc_mkdir_save_fat_buffer_retn:
1670 0000A477 C3                   <1>       retn
1671                                <1>
1672                                <1> loc_mkdir_save_fat_buffer_1:
1673                                <1>       ; byte [FAT_BuffValidData] = 2
1674 0000A478 E8871A0000           <1>       call   save_fat_buffer
1675 0000A47D 72E3                 <1>       jc     short loc_mkdir_save_fat_buffer_stc_retn
1676                                <1>
1677                                <1>       ; 01/03/2016
1678 0000A47F 803D[E65A0100]01     <1>       cmp    byte [FAT_ClusterCounter], 1
1679 0000A486 721B                 <1>       jb     short loc_mkdir_save_fat_buffer_2
1680                                <1>
1681                                <1>       ; ESI = Logical DOS Drive Description Table address
1682 0000A488 A1[E65A0100]         <1>       mov    eax, [FAT_ClusterCounter]
1683 0000A48D 66BB01FF             <1>       mov    bx, 0FF01h ; add free clusters
1684 0000A491 E8031B0000           <1>       call   calculate_fat_freespace
1685                                <1>
1686                                <1>       ;inc   eax ; 0FFFFFFFFh -> 0 ; recalculation is needed!
1687                                <1>       ;jnz   short loc_mkdir_save_fat_buffer_2
1688                                <1>
1689                                <1>       ; ecx > 0 -> Recalculation is needed
1690 0000A496 09C9                 <1>       or     ecx, ecx
1691 0000A498 7409                 <1>       jz     short loc_mkdir_save_fat_buffer_2
1692                                <1>
1693 0000A49A 66BB00FF             <1>       mov    bx, 0FF00h ; ; recalculate free space
1694 0000A49E E8F61A0000           <1>       call   calculate_fat_freespace
1695                                <1>
1696                                <1> loc_mkdir_save_fat_buffer_2:
1697 0000A4A3 C605[475D0100]01     <1>       mov    byte [mkdir_add_new_cluster], 1
1698 0000A4AA E9C4000000           <1>       jmp    loc_mkdir_upd_parent_dir_lmdt
1699                                <1>
1700                                <1> loc_mkdir_update_sub_dir_cluster:
1701 0000A4AF A1[385D0100]         <1>       mov    eax, [mkdir_FFCluster]
1702 0000A4B4 29C9                 <1>       sub    ecx, ecx ; 0
1703                                <1>       ; 01/03/2016
1704 0000A4B6 890D[E65A0100]       <1>       mov    [FAT_ClusterCounter], ecx ; 0 ; Reset
1705 0000A4BC 49                   <1>       dec    ecx ; 0FFFFFFFFh
1706                                <1>
1707                                <1>       ; ESI = Logical DOS Drive Descisption Table address
1708 0000A4BD E885170000           <1>       call   update_cluster
1709 0000A4C2 7379                 <1>       jnc    short loc_mkdir_save_fat_buffer_3
1710 0000A4C4 09C0                 <1>       or     eax, eax ; EAX = 0 -> cluster value is 0 or eocc
1711 0000A4C6 7475                 <1>       jz     short loc_mkdir_save_fat_buffer_3
1712                                <1>       ; 01/03/2016
1713 0000A4C8 EB98                 <1>       jmp    short loc_mkdir_save_fat_buffer_stc_retn
1714                                <1>
1715                                <1> loc_mkdir_set_ff_dir_entry_2:
1716                                <1>       ; ESI = Logical DOS Drive Description Table address
1717 0000A4CA A1[385D0100]         <1>       mov    eax, [mkdir_FFCluster]
1718                                <1>       ; Load disk sectors as a directory cluster
1719 0000A4CF E82F160000           <1>       call   load_FAT_sub_directory
1720 0000A4D4 7266                 <1>       jc     short retn_make_fat_directory
1721                                <1>
1722                                <1>       ; eax = 0
1723                                <1>       ; ecx =  directory buffer sector count (<= 128)
1724                                <1>
1725 0000A4D6 BF40000800           <1>       mov    edi, Directory_Buffer + 64 ; 26/02/2016
1726                                <1>
1727                                <1>       ; 02/03/2016
1728 0000A4DB 668B4611             <1>       mov    ax, [esi+LD_BPB+BytesPerSec] ; 512
1729 0000A4DF 66C1E802             <1>       shr    ax, 2 ; 'byte count / 4' for 'stosd'
1730 0000A4E3 F7E1                 <1>       mul    ecx
1731 0000A4E5 89C1                 <1>       mov    ecx, eax
1732 0000A4E7 6629C0               <1>       sub    ax, ax
1733 0000A4EA F3AB                 <1>       rep    stosd
1734                                <1>
1735                                <1>       ;;mov  al, 128 ; double word
1736                                <1>       ;;mul  ecx ; ecx =  directory buffer sector count
1737                                <1>       ;;mov  ecx, eax
1738                                <1>       ;shl   cx, 7 ; 128 * sector count
1739                                <1>       ;;sub  eax, eax
1740                                <1>       ;;sub  al, al ; 0
1741                                <1>       ;rep   stosd ; clear directory buffer
1742                                <1>
1743 0000A4EC BF00000800           <1>       mov    edi, Directory_Buffer ; 26/02/2016
1744                                <1>
1745 0000A4F1 56                   <1>       push   esi
1746                                <1>
1747 0000A4F2 BE[485D0100]         <1>       mov    esi, mkdir_Name
1748 0000A4F7 66C7062E00           <1>       mov    word [esi], 2Eh ; db '.', '0'
1749                                <1>
1750 0000A4FC A1[385D0100]         <1>       mov    eax, [mkdir_FFCluster]
1751 0000A501 66B91000             <1>       mov    cx, 10h ; CL = Directory attribute
1752                                <1>                      ; CH = 0 -> File size is 0
1753 0000A505 E883000000           <1>       call   make_directory_entry
1754                                <1>
1755 0000A50A BF20000800           <1>       mov    edi, Directory_Buffer + 32 ; 26/02/2016
1756                                <1>
1757                                <1>       ; 03/03/2016
1758                                <1>       ; Following modification has been done according to
```

```
1759                                <1>       ; 'Microsoft Extensible Firmware Initiative
1760                                <1>       ; FAT32 File System Specification' document,
1761                                <1>       ; 'FAT: General Overview of On-Disk Format–Page 25'.
1762                                <1>       ; "Finally, you set DIR_FstClusLO and DIR_FstClusHI
1763                                <1>       ; for the dotdot entry (the second entry) to the
1764                                <1>       ; first cluster number of the directory in which you
1765                                <1>       ; just created the directory (value is 0 if this directory
1766                                <1>       ; is the root directory even for FAT32 volumes)."
1767                                <1>       ; (Correctness of this modification has been verified
1768                                <1>       ;  by using Windows 98 'scandisk.exe'.)
1769                                <1>
1770 0000A50F 29C0                 <1>       sub    eax, eax
1771 0000A511 3805[C4520100]       <1>       cmp    byte [Current_Dir_Level], al ; 0
1772 0000A517 7605                 <1>       jna    short loc_mkdir_set_ff_dir_entry_3
1773 0000A519 A1[C0520100]         <1>       mov    eax, [Current_Dir_FCluster] ; parent dir
1774                                <1> loc_mkdir_set_ff_dir_entry_3:
1775 0000A51E 66C746012E00         <1>       mov    word [esi+1], 2Eh ; db '.', '0'
1776                                <1>
1777                                <1>       ;mov   cx, 10h
1778 0000A524 E864000000           <1>       call   make_directory_entry
1779                                <1>
1780 0000A529 5E                   <1>       pop    esi
1781                                <1>
1782 0000A52A C605[F05A0100]02     <1>       mov    byte [DirBuff_ValidData], 2
1783 0000A531 E834010000           <1>       call   save_directory_buffer
1784 0000A536 0F8373FFFFFF         <1>         jnc     loc_mkdir_update_sub_dir_cluster
1785                                <1>
1786                                <1> retn_make_fat_directory:
1787 0000A53C C3                   <1>       retn
1788                                <1>
1789                                <1> loc_mkdir_save_fat_buffer_3:
1790                                <1>       ; 01/03/2016
1791                                <1>       ; byte [FAT_BuffValidData] = 2
1792 0000A53D E8C2190000           <1>       call   save_fat_buffer
1793 0000A542 0F821AFFFFFF         <1>         jc      loc_mkdir_save_fat_buffer_stc_retn
1794                                <1>
1795 0000A548 803D[E65A0100]01     <1>       cmp    byte [FAT_ClusterCounter], 1
1796 0000A54F 721B                 <1>       jb     short loc_mkdir_save_fat_buffer_4
1797                                <1>
1798                                <1>       ; ESI = Logical DOS Drive Description Table address
1799 0000A551 A1[E65A0100]         <1>       mov    eax, [FAT_ClusterCounter]
1800 0000A556 66BB01FF             <1>       mov    bx, 0FF01h ; add free clusters
1801 0000A55A E83A1A0000           <1>       call   calculate_fat_freespace
1802                                <1>
1803                                <1>       ;inc   eax ; 0FFFFFFFFh -> 0 ; recalculation is needed!
1804                                <1>         ;jnz    short loc_mkdir_save_fat_buffer_4
1805                                <1>
1806                                <1>       ; ecx > 0 -> Recalculation is needed
1807 0000A55F 09C9                 <1>       or     ecx, ecx
1808 0000A561 7409                 <1>         jz      short loc_mkdir_save_fat_buffer_4
1809                                <1>
1810 0000A563 66BB00FF             <1>       mov    bx, 0FF00h ; recalculate free space
1811 0000A567 E82D1A0000           <1>       call   calculate_fat_freespace
1812                                <1>
1813                                <1> loc_mkdir_save_fat_buffer_4:
1814 0000A56C C605[475D0100]00     <1>       mov    byte [mkdir_add_new_cluster], 0
1815                                <1>
1816                                <1> loc_mkdir_upd_parent_dir_lmdt:
1817 0000A573 E88D010000           <1>       call   update_parent_dir_lmdt
1818                                <1>
1819                                <1>       ; 01/03/2016
1820 0000A578 803D[475D0100]00     <1>       cmp    byte [mkdir_add_new_cluster], 0
1821 0000A57F 0F8723FEFFFF         <1>         ja      loc_mkdir_gffc_2
1822                                <1>
1823                                <1> loc_mkdir_retn_new_dir_cluster:
1824 0000A585 A1[385D0100]         <1>       mov    eax, [mkdir_FFCluster]
1825 0000A58A 31D2                 <1>       xor    edx, edx
1826                                <1> loc_mkdir_retn:
1827 0000A58C C3                   <1>       retn
1828                                <1>
1829                                <1> make_directory_entry:
1830                                <1>       ; 02/03/2016
1831                                <1>       ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)
1832                                <1>       ; 09/08/2010 (DIR.ASM, 'proc_make_directory_entry')
1833                                <1>       ; 17/07/2010
1834                                <1>       ; INPUT ->
1835                                <1>       ;     EDI = Directory Entry Address
1836                                <1>       ;     ESI = Dot File Name Location
1837                                <1>       ;     EAX = First Cluster
1838                                <1>       ;     File Size = 0 (Must be set later)
1839                                <1>       ;     CL = Attributes
1840                                <1>       ;     CH = 0 (File size = 0)
1841                                <1>       ;     (If CH>0, File size is in dword [EBX]) (*)
1842                                <1>       ; OUTPUT ->
1843                                <1>       ;     EDI = Directory Entry Address
1844                                <1>       ;     ESI = Dot File Name Location (Capitalized)
1845                                <1>       ;     If CH input = 0, File Size = 0
1846                                <1>       ;     Otherwise file size is as dword [EBX] (*)
1847                                <1>       ;     DX = Date, AX = Time in DOS Dir Entry format
1848                                <1>       ;     EBX = same
1849                                <1>       ;     ECX = same
1850                                <1>
1851 0000A58D 51                   <1>       push   ecx
1852                                <1>
1853 0000A58E 884F0B               <1>       mov    [edi+11], cl ; Attributes
1854 0000A591 6689471A             <1>       mov    [edi+26], ax ; FClusterLw, 26
1855 0000A595 C1E810               <1>       shr    eax, 16
1856 0000A598 66894714             <1>       mov    [edi+20], ax ; FClusterHw, 20
1857 0000A59C 6631C0               <1>       xor    ax, ax
1858 0000A59F 6689470C             <1>       mov    [edi+12], ax ; NTReserved, 12
1859                                <1>                           ; CrtTimeTenth, 13
1860 0000A5A3 08ED                 <1>       or     ch, ch
```

```
1861 0000A5A5 7402              <1>       jz    short loc_make_direntry_set_filesize
1862                            <1>
1863 0000A5A7 8B03              <1>       mov   eax, [ebx]
1864                            <1>
1865                            <1> loc_make_direntry_set_filesize:
1866 0000A5A9 89471C            <1>       mov   [edi+28], eax ; FileSize, 28
1867                            <1>
1868 0000A5AC E88AFBFFFF        <1>       call  convert_file_name
1869                            <1>       ;EDI = Dir Entry Format File Name Location
1870                            <1>       ;ESI = Dot File Name Location (capitalized)
1871                            <1>
1872 0000A5B1 E816000000        <1>       call  convert_current_date_time
1873                            <1>       ; OUTPUT -> DX = Date in dos dir entry format
1874                            <1>         ;       AX = Time in dos dir entry format
1875 0000A5B6 6689470E          <1>       mov   [edi+14], ax ; CrtTime, 14
1876 0000A5BA 66895710          <1>       mov   [edi+16], dx ; CrtDate, 16
1877 0000A5BE 66895712          <1>       mov   [edi+18], dx ; LastAccDate, 18
1878 0000A5C2 66894716          <1>       mov   [edi+22], ax ; WrtTime, 14
1879 0000A5C6 66895718          <1>       mov   [edi+24], dx ; WrtDate, 16
1880 0000A5CA 59                <1>       pop   ecx
1881                            <1>
1882 0000A5CB C3                <1>       retn
1883                            <1>
1884                            <1> convert_current_date_time:
1885                            <1>       ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)
1886                            <1>       ; 13/06/2010 (DIR.ASM, 'proc_convert_current_date_time')
1887                            <1>       ; converts date&time to dos dir entry format
1888                            <1>       ; INPUT -> none
1889                            <1>       ; OUTPUT -> DX = Date in dos dir entry format
1890                            <1>       ;           AX = Time in dos dir entry format
1891                            <1>
1892 0000A5CC B404                <1>       mov   ah, 04h ; Return Current Date
1893 0000A5CE E8A6B3FFFF        <1>       call  int1Ah
1894                            <1>
1895 0000A5D3 88E8              <1>       mov   al, ch ; <- century BCD
1896 0000A5D5 240F              <1>       and   al, 0Fh
1897 0000A5D7 88EC              <1>       mov   ah, ch
1898 0000A5D9 C0EC04            <1>       shr   ah, 4
1899 0000A5DC D50A              <1>       aad
1900 0000A5DE 88C5              <1>       mov   ch, al ; -> century
1901                            <1>
1902 0000A5E0 88C8              <1>       mov   al, cl ; <- year BCD
1903 0000A5E2 240F              <1>       and   al, 0Fh
1904 0000A5E4 88CC              <1>       mov   ah, cl
1905 0000A5E6 C0EC04            <1>       shr   ah, 4
1906 0000A5E9 D50A              <1>       aad
1907 0000A5EB 88C1              <1>       mov   cl, al ; -> year
1908                            <1>
1909 0000A5ED 88E8              <1>       mov   al, ch
1910 0000A5EF B464              <1>       mov   ah, 100
1911 0000A5F1 F6E4              <1>       mul   ah
1912 0000A5F3 30ED              <1>       xor   ch, ch
1913 0000A5F5 6601C8            <1>       add   ax, cx
1914 0000A5F8 662DBC07          <1>       sub   ax, 1980 ; ms-dos epoch
1915 0000A5FC 6689C1            <1>       mov   cx, ax
1916                            <1>
1917 0000A5FF 88F0              <1>       mov   al, dh ; <- month in bcd
1918 0000A601 240F              <1>       and   al, 0Fh
1919 0000A603 88F4              <1>       mov   ah, dh
1920 0000A605 C0EC04            <1>       shr   ah, 4
1921 0000A608 D50A              <1>       aad
1922 0000A60A 88C6              <1>       mov   dh, al ; -> month
1923                            <1>
1924 0000A60C 88D0              <1>       mov   al, dl ; <- day BCD
1925 0000A60E 240F              <1>       and   al, 0Fh
1926 0000A610 88D4              <1>       mov   ah, dl
1927 0000A612 C0EC04            <1>       shr   ah, 4
1928 0000A615 D50A              <1>       aad
1929 0000A617 88C2              <1>       mov   dl, al ; -> day
1930                            <1>
1931 0000A619 88C8              <1>       mov   al, cl ; count of years from 1980
1932 0000A61B 66C1E004          <1>       shl   ax, 4
1933 0000A61F 08F0              <1>       or    al, dh ; month of year, 1 to 12
1934 0000A621 66C1E005          <1>       shl   ax, 5
1935 0000A625 08D0              <1>       or    al, dl ; day of year, 1 to 31
1936                            <1>
1937 0000A627 6650              <1>       push  ax ; push date
1938                            <1>
1939 0000A629 B402              <1>       mov   ah, 02h ; Return Current Time
1940 0000A62B E849B3FFFF        <1>       call  int1Ah
1941                            <1>
1942 0000A630 88E8              <1>       mov   al, ch ; <- hours BCD
1943 0000A632 240F              <1>       and   al, 0Fh
1944 0000A634 88EC              <1>       mov   ah, ch
1945 0000A636 C0EC04            <1>       shr   ah, 4
1946 0000A639 D50A              <1>       aad
1947 0000A63B 88C5              <1>       mov   ch, al ; -> hours
1948                            <1>
1949 0000A63D 88C8              <1>       mov   al, cl ; <- minutes BCD
1950 0000A63F 240F              <1>       and   al, 0Fh
1951 0000A641 88CC              <1>       mov   ah, cl
1952 0000A643 C0EC04            <1>       shr   ah, 4
1953 0000A646 D50A              <1>       aad
1954 0000A648 88C1              <1>       mov   cl, al ; -> minutes
1955                            <1>
1956 0000A64A 88F0              <1>       mov   al, dh ; <- seconds BCD
1957 0000A64C 240F              <1>       and   al, 0Fh
1958 0000A64E 88F4              <1>       mov   ah, dh
1959 0000A650 C0EC04            <1>       shr   ah, 4
1960 0000A653 D50A              <1>       aad
1961 0000A655 88C6              <1>       mov   dh, al ; -> seconds
1962                            <1>
```

```
1963 0000A657 88E8              <1>       mov    al, ch ; hours
1964 0000A659 66C1E006          <1>       shl    ax, 6
1965 0000A65D 08C8              <1>       or     al, cl ; minutes
1966 0000A65F 66C1E005          <1>       shl    ax, 5
1967 0000A663 D0EE              <1>       shr    dh, 1 ; 2 seconds
1968                            <1>       ; There is a bug in TRDOS v1 here !
1969                            <1>       ; it was 'or al, dl' !
1970 0000A665 08F0              <1>       or     al, dh ; seconds
1971                            <1>
1972 0000A667 665A              <1>       pop    dx ; pop date
1973                            <1>
1974 0000A669 C3                <1>       retn
1975                            <1>
1976                            <1> save_directory_buffer:
1977                            <1>       ; 15/10/2016
1978                            <1>       ; 23/03/2016
1979                            <1>       ; 26/02/2016
1980                            <1>       ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)
1981                            <1>       ; 01/08/2011
1982                            <1>       ; 14/03/2010
1983                            <1>       ; INPUT ->
1984                            <1>       ;      none
1985                            <1>       ; OUTPUT ->
1986                            <1>       ;  cf = 0 -> write OK...
1987                            <1>       ;  cf = 1 -> error code in AL (EAX)
1988                            <1>       ;  cf = 1 & AL = 0Dh => CH & CL = FS & FAT type
1989                            <1>       ;  EBX = Directory Buffer Address
1990                            <1>       ;
1991                            <1>       ;  (EAX, ECX, EDX will be modified)
1992                            <1>
1993 0000A66A BB00000800        <1>       mov    ebx, Directory_Buffer
1994 0000A66F 803D[F05A0100]02  <1>       cmp    byte [DirBuff_ValidData], 2
1995 0000A676 7403              <1>       je     short loc_save_dir_buffer
1996 0000A678 31C0              <1>       xor    eax, eax
1997 0000A67A C3                <1>       retn
1998                            <1>
1999                            <1> loc_save_dir_buffer:
2000 0000A67B 56                <1>       push   esi
2001 0000A67C 31DB              <1>       xor    ebx, ebx
2002 0000A67E 8A3D[EE5A0100]    <1>         mov     bh, [DirBuff_DRV]
2003 0000A684 80EF41            <1>       sub    bh, 'A'
2004 0000A687 BE00010900        <1>         mov     esi, Logical_DOSDisks
2005 0000A68C 01DE              <1>       add    esi, ebx
2006 0000A68E 668B4E03          <1>         mov     cx, [esi+LD_FATType]
2007                            <1>       ; CH = FS Type (A1h for FS)
2008                            <1>       ; CL = FAT Type (0 for FS)
2009 0000A692 08C9              <1>       or     cl, cl
2010 0000A694 7433              <1>       jz     short loc_save_dir_buff_stc_retn
2011                            <1>
2012                            <1> loc_save_dir_buffer_check_cluster_no:
2013 0000A696 A1[F55A0100]      <1>       mov    eax, [DirBuff_Cluster]
2014 0000A69B 28FF              <1>       sub    bh, bh ; ebx = 0
2015 0000A69D 09C0              <1>       or     eax, eax
2016 0000A69F 7540              <1>       jnz    short loc_save_sub_dir_buffer
2017 0000A6A1 8A25[EF5A0100]    <1>       mov    ah, [DirBuff_FATType]
2018 0000A6A7 FEC3              <1>       inc    bl ;  bl = 1
2019 0000A6A9 38DC              <1>       cmp    ah, bl
2020 0000A6AB 721D              <1>       jb     short loc_save_dir_buff_inv_data_retn
2021 0000A6AD FEC3              <1>       inc    bl ; bl = 2
2022 0000A6AF 38E3              <1>       cmp    bl, ah
2023 0000A6B1 7217              <1>       jb     short loc_save_dir_buff_inv_data_retn
2024                            <1>
2025                            <1> loc_save_root_dir_buffer:
2026 0000A6B3 668B5E17          <1>       mov    bx, [esi+LD_BPB+RootDirEnts]
2027 0000A6B7 6683C30F          <1>       add    bx, 15
2028 0000A6BB 66C1EB04          <1>       shr    bx, 4 ; 16 dir entries per sector
2029 0000A6BF 6609DB            <1>       or     bx, bx
2030 0000A6C2 7405              <1>       jz     short loc_save_dir_buff_stc_retn
2031                            <1>       ;mov   ecx, ebx
2032 0000A6C4 8B4664            <1>       mov    eax, [esi+LD_ROOTBegin] ; 26/02/2016
2033 0000A6C7 EB23              <1>       jmp    short loc_write_directory_to_disk
2034                            <1>
2035                            <1> loc_save_dir_buff_stc_retn:
2036 0000A6C9 F9                <1>       stc
2037                            <1> loc_save_dir_buff_inv_data_retn:
2038                            <1>       ; 15/10/2016 (0Dh -> 29)
2039 0000A6CA B01D              <1>       mov    al, 29 ; Invalid data !
2040 0000A6CC C605[F05A0100]00  <1>       mov    byte [DirBuff_ValidData], 0
2041 0000A6D3 EB05              <1>       jmp    short loc_save_dir_buff_retn
2042                            <1>
2043                            <1> loc_write_directory_to_disk_err:
2044                            <1>       ; 15/10/2016 (disk write error code, 1Dh -> 18)
2045 0000A6D5 B812000000        <1>       mov    eax, 18 ; Drive not ready or write error
2046                            <1>
2047                            <1> loc_save_dir_buff_retn:
2048 0000A6DA BB00000800        <1>       mov    ebx, Directory_Buffer
2049 0000A6DF 5E                <1>       pop    esi
2050 0000A6E0 C3                <1>       retn
2051                            <1>
2052                            <1> loc_save_sub_dir_buffer:
2053                            <1>       ; ebx  = 0
2054 0000A6E1 83E802            <1>       sub    eax, 2
2055 0000A6E4 8A5E13            <1>       mov    bl, [esi+LD_BPB+SecPerClust]
2056 0000A6E7 F7E3              <1>       mul    ebx
2057 0000A6E9 034668            <1>         add     eax, [esi+LD_DATABegin]
2058                            <1>       ;mov   ecx, ebx
2059                            <1>
2060                            <1> loc_write_directory_to_disk:
2061 0000A6EC 89D9              <1>       mov    ecx, ebx
2062 0000A6EE BB00000800        <1>       mov    ebx, Directory_Buffer
2063 0000A6F3 E8AB4A0000        <1>       call   disk_write
2064 0000A6F8 72DB              <1>       jc     short loc_write_directory_to_disk_err
```

```
2065                                <1>
2066                                <1> loc_save_dir_buff_validate_retn:
2067 0000A6FA C605[F05A0100]01     <1>       mov    byte [DirBuff_ValidData], 1
2068 0000A701 31C0                 <1>       xor    eax, eax
2069                                <1>       ; 26/02/2016
2070 0000A703 EBD5                 <1>       jmp    short loc_save_dir_buff_retn
2071                                <1>
2072                                <1> update_parent_dir_lmdt:
2073                                <1>       ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)
2074                                <1>       ; 01/08/2011
2075                                <1>       ; 16/10/2010
2076                                <1>       ;
2077                                <1>       ; INPUT ->
2078                                <1>       ;      none
2079                                <1>       ; OUTPUT ->
2080                                <1>       ;      (last modification date & time of the parent dir
2081                                <1>       ;      will be changed/updated)
2082                                <1>       ;
2083                                <1>       ; (EAX, EBX, ECX, EDX, EDI will be changed)
2084                                <1>
2085 0000A705 29C0                 <1>       sub    eax, eax
2086 0000A707 8A25[C4520100]       <1>       mov    ah, [Current_Dir_Level]
2087 0000A70D A0[C5520100]         <1>       mov    al, [Current_FATType]
2088 0000A712 3C01                 <1>       cmp    al, 1
2089 0000A714 723A                 <1>       jb     short loc_UPDLMDT_proc_retn
2090                                <1>
2091                                <1> loc_update_parent_dir_lm_date_time:
2092 0000A716 08E4                 <1>       or     ah, ah
2093 0000A718 7436                 <1>       jz     short loc_UPDLMDT_proc_retn
2094                                <1>
2095 0000A71A 56                   <1>       push   esi ; *
2096 0000A71B 8825[685D0100]       <1>       mov    [UPDLMDT_CDirLevel], ah
2097 0000A721 8B15[C0520100]       <1>       mov    edx, [Current_Dir_FCluster]
2098 0000A727 8915[695D0100]       <1>       mov    [UPDLMDT_CDirFCluster], edx
2099                                <1>
2100 0000A72D FECC                 <1>       dec    ah
2101 0000A72F B90C000000           <1>       mov    ecx, 12
2102 0000A734 BE[275B0100]         <1>         mov     esi, PATH_Array
2103                                <1>
2104 0000A739 8825[C4520100]       <1>       mov    [Current_Dir_Level], ah
2105 0000A73F 08E4                 <1>       or     ah, ah
2106 0000A741 750E                 <1>       jnz    short loc_update_parent_dir_lmdt_load_sub_dir_1
2107 0000A743 803D[C5520100]02     <1>       cmp    byte [Current_FATType], 2
2108 0000A74A 770B                 <1>       ja     short loc_update_parent_dir_lmdt_load_sub_dir_2
2109 0000A74C 28C0                 <1>       sub    al, al ; eax = 0
2110 0000A74E EB0A                 <1>       jmp    short loc_update_parent_dir_lmdt_load_sub_dir_3
2111                                <1>
2112                                <1> loc_UPDLMDT_proc_retn:
2113 0000A750 C3                   <1>       retn
2114                                <1>
2115                                <1> loc_update_parent_dir_lmdt_load_sub_dir_1:
2116 0000A751 B010                 <1>       mov    al, 16
2117 0000A753 F6E4                 <1>       mul    ah
2118 0000A755 01C6                 <1>       add    esi, eax
2119                                <1>
2120                                <1> loc_update_parent_dir_lmdt_load_sub_dir_2:
2121 0000A757 8B460C               <1>       mov    eax, [esi+12] ; Parent Dir First Cluster
2122                                <1>
2123                                <1> loc_update_parent_dir_lmdt_load_sub_dir_3:
2124 0000A75A A3[C0520100]         <1>       mov    [Current_Dir_FCluster], eax
2125                                <1>
2126 0000A75F 83C610               <1>       add    esi, 16
2127 0000A762 66BF[4E5C]           <1>       mov    di, Dir_File_Name
2128 0000A766 F3A4                 <1>       rep    movsb
2129                                <1>
2130 0000A768 BE00010900           <1>       mov    esi, Logical_DOSDisks
2131 0000A76D 29DB                 <1>       sub    ebx, ebx
2132 0000A76F 8A3D[C6520100]       <1>       mov    bh, [Current_Drv]
2133 0000A775 01DE                 <1>       add    esi, ebx
2134 0000A777 E88FF7FFFF           <1>       call   reload_current_directory
2135 0000A77C 7232                 <1>       jc     short loc_update_parent_dir_lmdt_restore_cdirlevel
2136                                <1>
2137                                <1> loc_update_parent_dir_lmdt_locate_dir:
2138 0000A77E BE[4E5C0100]         <1>       mov    esi, Dir_File_Name
2139 0000A783 6631C9               <1>       xor    cx, cx
2140 0000A786 66B81008             <1>       mov    ax, 0810h ; Only directories
2141 0000A78A E8B7F6FFFF           <1>         call    locate_current_dir_file
2142                                <1>       ; EDI = DirBuff Directory Entry Address
2143 0000A78F 721F                 <1>       jc short loc_update_parent_dir_lmdt_restore_cdirlevel
2144                                <1>
2145 0000A791 E836FEFFFF           <1>       call   convert_current_date_time
2146 0000A796 66895712             <1>       mov    [edi+18], dx ; Last Access Date
2147 0000A79A 66895718             <1>       mov    [edi+24], dx ; Last Write Date
2148 0000A79E 66894716             <1>       mov    [edi+22], ax ; Last Write Time
2149                                <1>
2150 0000A7A2 C605[F05A0100]02     <1>       mov    byte [DirBuff_ValidData], 2
2151 0000A7A9 E8BCFEFFFF           <1>       call   save_directory_buffer
2152 0000A7AE 7200                 <1>       jc     short loc_update_parent_dir_lmdt_restore_cdirlevel
2153                                <1>       ;xor   al, al
2154                                <1> loc_update_parent_dir_lmdt_restore_cdirlevel:
2155                                <1>       ;current directory level restoration
2156 0000A7B0 8A25[685D0100]       <1>       mov    ah, [UPDLMDT_CDirLevel]
2157 0000A7B6 8825[C4520100]       <1>       mov    [Current_Dir_Level], ah
2158 0000A7BC 8B15[695D0100]       <1>         mov     edx, [UPDLMDT_CDirFCluster]
2159 0000A7C2 8915[C0520100]       <1>       mov    [Current_Dir_FCluster], edx
2160                                <1>
2161 0000A7C8 5E                   <1>       pop    esi ; *
2162 0000A7C9 C3                   <1>       retn
2163                                <1>
2164                                <1> delete_longname:
2165                                <1>       ; 27/02/2016 (TRDOS 386 = TRDOS v2.0)
2166                                <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>
0000A7CA 66A3[985D0100]         <1>       mov   [DLN_EntryNumber], ax
0000A7D0 C605[9A5D0100]40       <1>         mov     byte [DLN_40h], 40h
                                <1>
0000A7D7 E858000000             <1>       call  locate_current_dir_entry
0000A7DC 7308                   <1>       jnc   short loc_dln_check_attributes
0000A7DE C3                     <1>       retn
                                <1>
                                <1> loc_dln_longname_not_found:
0000A7DF B802000000             <1>       mov   eax, 2
0000A7E4 F9                     <1>       stc
0000A7E5 C3                     <1>       retn
                                <1>
                                <1> loc_dln_check_attributes:
0000A7E6 B00F                   <1>       mov   al, 0Fh  ; long name
0000A7E8 8A670B                 <1>       mov   ah, [edi+0Bh] ; dir entry attributes
0000A7EB 38C4                   <1>       cmp   ah, al
0000A7ED 75F0                   <1>       jne   short loc_dln_longname_not_found
0000A7EF 8A27                   <1>       mov   ah, [edi]
0000A7F1 2A25[9A5D0100]         <1>       sub   ah, [DLN_40h]
0000A7F7 76E6                   <1>       jna   short loc_dln_longname_not_found
0000A7F9 80FC14                 <1>       cmp   ah, 14h ; 84-64=20 -> 20*13=260 bytes
0000A7FC 77E1                   <1>       ja    short loc_dln_longname_not_found
                                <1>
0000A7FE C607E5                 <1>       mov   byte [edi], 0E5h  ; deleted sign
0000A801 C605[F05A0100]02       <1>       mov   byte [DirBuff_ValidData], 2 ; changed/write sign
0000A808 C605[9A5D0100]00       <1>       mov   byte [DLN_40h], 0 ; 40h -> 0
                                <1>
                                <1> loc_dln_delete_next_ln_entry:
0000A80F 80FC01                 <1>       cmp   ah, 1
0000A812 7616                   <1>       jna   short loc_dln_longname_retn
                                <1> loc_dln_delete_next_ln_entry_0:
0000A814 66FF05[985D0100]       <1>       inc   word [DLN_EntryNumber]
0000A81B 0FB705[985D0100]       <1>       movzx eax, word [DLN_EntryNumber]
0000A822 E80D000000             <1>       call  locate_current_dir_entry
0000A827 73BD                   <1>       jnc   short loc_dln_check_attributes
                                <1>
                                <1> loc_dln_longname_stc_retn:
0000A829 C3                     <1>       retn
                                <1>
                                <1> loc_dln_longname_retn:
                                <1>       ;cmp  byte [DirBuff_ValidData], 2
                                <1>       ;jne  short loc_dln_longname_retn_xor_eax
0000A82A E83BFEFFFF             <1>       call  save_directory_buffer
0000A82F 72F8                   <1>       jc    short loc_dln_longname_stc_retn
                                <1>
                                <1> loc_dln_longname_retn_xor_eax:
0000A831 31C0                   <1>       xor   eax, eax
0000A833 C3                     <1>       retn
                                <1>
                                <1> locate_current_dir_entry:
                                <1>       ; 16/10/2016
                                <1>       ; 15/10/2016
                                <1>       ; 23/03/2016
                                <1>       ; 27/02/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>       ; 01/08/2011 (DIR.ASM, 'proc_locate_current_dir_entry')
                                <1>       ; 07/03/2010
                                <1>       ; INPUT ->
                                <1>       ;     EAX = Directory Entry (Index) Number (< 65536)
                                <1>       ; OUTPUT ->
                                <1>       ;     EDI = Directory Entry Address
                                <1>       ;     EAX = Cluster Number of Directory Buffer
                                <1>       ;     EBX = Directory Buffer Entry Offset
                                <1>       ;     ECX = DirBuff Valid Data identifier (CL)
                                <1>       ;     If CF = 0 and CL = 2 then
                                <1>       ;        directory buffer modified and
                                <1>       ;        must be written to disk.
                                <1>       ;     If CF = 0  and CL = 1 then
                                <1>       ;        dir buffer has been written to disk, already.
                                <1>       ;     CF = 1 -> Error code in EAX (AL)
                                <1>       ;
                                <1>       ; (Modified registers: EAX, EDX, ECX, EBX, EDI)
                                <1>
                                <1> loc_locate_current_dir_entry:
0000A834 56                     <1>       push  esi
0000A835 89C1                   <1>       mov   ecx, eax
0000A837 BA20000000             <1>       mov   edx, 32
0000A83C F7E2                   <1>       mul   edx
0000A83E A3[A45D0100]           <1>       mov   [LCDE_ByteOffset], eax
0000A843 31DB                   <1>       xor   ebx, ebx
0000A845 8A3D[C6520100]         <1>       mov   bh, [Current_Drv]
0000A84B A0[EE5A0100]           <1>         mov     al, [DirBuff_DRV]
0000A850 2C41                   <1>       sub   al, 'A'
0000A852 BE00010900             <1>         mov     esi, Logical_DOSDisks
0000A857 01DE                   <1>       add   esi, ebx
0000A859 38C7                   <1>       cmp   bh, al
0000A85B 0F8592000000           <1>         jne     loc_lcde_reload_current_directory
                                <1> loc_lcde_cdl_check:
0000A861 803D[C4520100]00       <1>       cmp   byte [Current_Dir_Level], 0
0000A868 772A                   <1>       ja    short loc_lcde_calc_dirbuff_cluster_offset
                                <1>       ; 27/02/2016
                                <1>       ; TRDOS v1 has bug here for FAT32 fs !
                                <1>       ; (Root Directory Entries for FAT32 = 0)
0000A86A 807E0303               <1>       cmp   byte [esi+LD_FATType], 3  ; FAT32
```

```
2269 0000A86E 7324                <1>       jnb    short loc_lcde_calc_dirbuff_cluster_offset
2270                              <1>
2271                              <1> loc_lcde_cdl_check_FAT12_16:
2272 0000A870 668B4617            <1>       mov    ax, [esi+LD_BPB+RootDirEnts]
2273 0000A874 6648                <1>       dec    ax
2274                              <1>       ;xor   dx, dx
2275 0000A876 6639C8              <1>       cmp    ax, cx ; cx = Directory Entry (Index) Number
2276 0000A879 720E                <1>       jb     short loc_lcde_stc_12h_retn
2277 0000A87B 66890D[9C5D0100]    <1>       mov    [LCDE_EntryIndex], cx
2278 0000A882 31C0                <1>       xor    eax, eax
2279 0000A884 E993000000          <1>         jmp     loc_lcde_check_dir_buffer_cluster
2280                              <1>
2281                              <1> loc_lcde_stc_12h_retn:
2282 0000A889 5E                  <1>       pop    esi
2283 0000A88A 89CB                <1>       mov    ebx, ecx
2284 0000A88C 89D1                <1>       mov    ecx, edx
2285                              <1>       ; 16/10/2016 (12h -> 12)
2286 0000A88E B80C000000          <1>       mov    eax, 12 ; No more files
2287 0000A893 C3                  <1>       retn
2288                              <1>
2289                              <1> loc_lcde_calc_dirbuff_cluster_offset:
2290 0000A894 8A5E13              <1>       mov    bl, [esi+LD_BPB+SecPerClust]
2291 0000A897 30FF                <1>       xor    bh, bh
2292 0000A899 668B4611            <1>       mov    ax, [esi+LD_BPB+BytesPerSec]
2293 0000A89D 66F7E3              <1>       mul    bx
2294 0000A8A0 6609D2              <1>       or     dx, dx ; If bytes per cluster > 32KB it is invalid
2295 0000A8A3 755D                <1>       jnz    short loc_lcde_invalid_format
2296                              <1>       ;mov   ecx, eax
2297 0000A8A5 6689C1              <1>       mov    cx, ax ; BYTES PER CLUSTER
2298 0000A8A8 A1[A45D0100]        <1>       mov    eax, [LCDE_ByteOffset]
2299                              <1>       ;sub   edx, edx
2300 0000A8AD F7F1                <1>       div    ecx
2301 0000A8AF 3DFFFF0000          <1>       cmp    eax, 65535
2302 0000A8B4 774C                <1>       ja     short loc_lcde_invalid_format
2303                              <1>
2304                              <1>       ; cluster sequence number of directory (< 65536)
2305 0000A8B6 66A3[9E5D0100]      <1>       mov    [LCDE_ClusterSN], ax
2306                              <1>
2307 0000A8BC 6689D0              <1>       mov    ax, dx ; byte offset in cluster (directory buffer)
2308 0000A8BF 66BB2000            <1>       mov    bx, 32 ; ; 1 dir entry = 32 bytes
2309 0000A8C3 6629D2              <1>         sub     dx, dx  ; 0
2310 0000A8C6 66F7F3              <1>       div    bx
2311 0000A8C9 66A3[9C5D0100]      <1>       mov    [LCDE_EntryIndex], ax ; dir entry index/sequence number
2312                              <1>                                     ; (in directory buffer/cluster)
2313                              <1> loc_lcde_get_current_sub_dir_fcluster:
2314 0000A8CF A1[C0520100]        <1>       mov    eax, [Current_Dir_FCluster]
2315                              <1>
2316                              <1> loc_lcde_get_next_cluster:
2317 0000A8D4 66833D[9E5D0100]00  <1>       cmp    word [LCDE_ClusterSN], 0
2318 0000A8DC 763E                <1>       jna    short loc_lcde_check_dir_buffer_cluster
2319 0000A8DE A3[A05D0100]        <1>       mov    [LCDE_Cluster], eax
2320 0000A8E3 E835100000          <1>       call   get_next_cluster
2321 0000A8E8 7220                <1>       jc     short loc_lcde_check_gnc_error
2322 0000A8EA 66FF0D[9E5D0100]    <1>       dec    word [LCDE_ClusterSN]
2323 0000A8F1 EBE1                <1>       jmp    short loc_lcde_get_next_cluster
2324                              <1>
2325                              <1> loc_lcde_reload_current_directory:
2326 0000A8F3 51                  <1>       push   ecx
2327 0000A8F4 E812F6FFFF          <1>       call   reload_current_directory
2328 0000A8F9 59                  <1>       pop    ecx
2329 0000A8FA 0F8361FFFFFF        <1>         jnc     loc_lcde_cdl_check
2330 0000A900 5E                  <1>       pop    esi
2331 0000A901 C3                  <1>       retn
2332                              <1>
2333                              <1> loc_lcde_invalid_format:
2334                              <1>       ; 15/10/2016 (0Bh -> 28)
2335 0000A902 B81C000000          <1>       mov    eax, 28 ; Invalid Format !
2336                              <1> loc_lcde_drive_not_ready_read_err:
2337 0000A907 F9                  <1>       stc
2338 0000A908 5E                  <1>       pop    esi
2339 0000A909 C3                  <1>       retn
2340                              <1>
2341                              <1> loc_lcde_check_gnc_error:
2342 0000A90A 09C0                <1>       or     eax, eax
2343 0000A90C 75F9                <1>       jnz    short loc_lcde_drive_not_ready_read_err
2344 0000A90E 66FF0D[9E5D0100]    <1>       dec    word [LCDE_ClusterSN]
2345 0000A915 75EB                <1>       jnz    short loc_lcde_invalid_format
2346 0000A917 A1[A05D0100]        <1>       mov    eax, [LCDE_Cluster]
2347                              <1>
2348                              <1> loc_lcde_check_dir_buffer_cluster:
2349 0000A91C 3B05[F55A0100]      <1>       cmp    eax, [DirBuff_Cluster]
2350 0000A922 755C                <1>       jne    short loc_lcde_load_dir_cluster
2351 0000A924 803D[F05A0100]00    <1>       cmp    byte [DirBuff_ValidData], 0
2352 0000A92B 7727                <1>       ja     short lcde_check_dir_buffer_cluster_next
2353 0000A92D 803D[C4520100]00    <1>       cmp    byte [Current_Dir_Level], 0
2354 0000A934 775F                <1>       ja     short loc_lcde_load_dir_cluster_0
2355                              <1>       ; 27/02/2016
2356                              <1>       ; TRDOS v1 has bug here for FAT32 fs !
2357 0000A936 807E0303            <1>       cmp    byte [esi+LD_FATType], 3  ; FAT32
2358 0000A93A 7359                <1>       jnb    short loc_lcde_load_dir_cluster_0
2359                              <1>       ;
2360 0000A93C 0FB74E17            <1>       movzx  ecx, word [esi+LD_BPB+RootDirEnts]
2361 0000A940 6683C10F            <1>       add    cx, 15 ; round up (16 entries per sector)
2362 0000A944 66C1E904            <1>       shr    cx, 4 ; 1 sector contains 16 dir entries
2363                              <1>
2364 0000A948 8B4664              <1>         mov     eax, [esi+LD_ROOTBegin]
2365 0000A94B EB54                <1>       jmp    short loc_lcde_load_dir_cluster_1
2366                              <1>
2367                              <1> loc_lcde_validate_dirBuff:
2368 0000A94D C605[F05A0100]01    <1>       mov    byte [DirBuff_ValidData], 1
2369                              <1>
2370                              <1> lcde_check_dir_buffer_cluster_next:
```

```
2371 0000A954 0FB71D[9C5D0100]    <1>       movzx  ebx, word [LCDE_EntryIndex]
2372 0000A95B 663B1D[F35A0100]    <1>       cmp    bx, [DirBuff_LastEntry]
2373 0000A962 779E                <1>       ja     short loc_lcde_invalid_format
2374 0000A964 B820000000          <1>       mov    eax, 32
2375 0000A969 F7E3                <1>       mul    ebx
2376                              <1>       ;or    edx, edx
2377                              <1>       ;jnz   short loc_lcde_invalid_format
2378                              <1>
2379 0000A96B BF00000800          <1>       mov    edi, Directory_Buffer
2380 0000A970 01C7                <1>       add    edi, eax ; add entry offset to buffer address
2381                              <1>
2382                              <1> loc_lcde_dir_buffer_last_check:
2383 0000A972 A1[F55A0100]        <1>       mov    eax, [DirBuff_Cluster]
2384 0000A977 0FB60D[F05A0100]    <1>       movzx  ecx, byte [DirBuff_ValidData]
2385                              <1>
2386                              <1> loc_lcde_retn:
2387 0000A97E 5E                  <1>       pop    esi
2388 0000A97F C3                  <1>       retn
2389                              <1>
2390                              <1> loc_lcde_load_dir_cluster:
2391                              <1>       ;cmp   byte [DirBuff_ValidData], 2
2392                              <1>       ;jne   short loc_lcde_load_dir_cluster_n2
2393 0000A980 50                  <1>       push   eax
2394 0000A981 E8E4FCFFFF          <1>       call   save_directory_buffer
2395 0000A986 58                  <1>       pop    eax
2396 0000A987 72F5                <1>       jc     short loc_lcde_retn
2397                              <1>
2398                              <1> loc_lcde_load_dir_cluster_n2:
2399 0000A989 C605[F05A0100]00    <1>       mov    byte [DirBuff_ValidData], 0
2400 0000A990 A3[F55A0100]        <1>       mov    [DirBuff_Cluster], eax
2401                              <1>
2402                              <1> loc_lcde_load_dir_cluster_0:
2403 0000A995 83E802              <1>       sub    eax, 2
2404 0000A998 0FB64E13            <1>       movzx  ecx, byte [esi+LD_BPB+SecPerClust]
2405 0000A99C F7E1                <1>       mul    ecx
2406 0000A99E 034668              <1>         add     eax, [esi+LD_DATABegin]
2407                              <1>
2408                              <1> loc_lcde_load_dir_cluster_1:
2409 0000A9A1 BB00000800          <1>       mov    ebx, Directory_Buffer
2410                              <1>       ; ecx = sector count
2411 0000A9A6 E807480000          <1>       call   disk_read
2412 0000A9AB 73A0                <1>       jnc    short loc_lcde_validate_dirBuff
2413                              <1>
2414                              <1>       ; 15/10/2016
2415                              <1>       ; (Disk read error instead of drv not ready err)
2416 0000A9AD B811000000          <1>       mov    eax, 17 ; Drive not ready or read error !
2417 0000A9B2 EBCA                <1>       jmp    short loc_lcde_retn
2418                              <1>
2419                              <1>
2420                              <1> remove_file:
2421                              <1>       ; 15/10/2016
2422                              <1>       ; 28/02/2016 (TRDOS 386 = TRDOS v2.0)
2423                              <1>       ; 10/04/2011 (FILE.ASM, 'proc_delete_file')
2424                              <1>       ; 09/08/2010
2425                              <1>       ; INPUT ->
2426                              <1>       ;      EDI = Directory Buffer Entry Address
2427                              <1>       ;       CX = Directory Buffer Entry Counter/Index
2428                              <1>       ;       BL = Longname Entry Length
2429                              <1>       ;       BH = Logical DOS Drive Number
2430                              <1>
2431 0000A9B4 29C0                <1>       sub    eax, eax
2432 0000A9B6 88FC                <1>       mov    ah, bh
2433 0000A9B8 BE00010900          <1>       mov    esi, Logical_DOSDisks
2434 0000A9BD 01C6                <1>       add    esi, eax
2435                              <1>
2436 0000A9BF 807E0301            <1>       cmp    byte [esi+LD_FATType], 1
2437 0000A9C3 7312                <1>       jnb    short loc_del_fat_file
2438                              <1>
2439 0000A9C5 807E04A1            <1>       cmp    byte [esi+LD_FSType], 0A1h
2440 0000A9C9 7406                <1>       je     short loc_del_fs_file
2441                              <1>
2442                              <1> loc_del_file_invalid_format:
2443 0000A9CB 30E4                <1>       xor    ah, ah
2444                              <1>       ; 15/10/2016 (0Bh -> 28)
2445 0000A9CD B01C                <1>       mov    al, 28  ; Invalid Format
2446 0000A9CF F9                  <1>       stc
2447 0000A9D0 C3                  <1>       retn
2448                              <1>
2449                              <1> loc_del_fs_file:
2450 0000A9D1 E8400F0000          <1>       call   delete_fs_file
2451 0000A9D6 C3                  <1>       retn
2452                              <1>
2453                              <1> loc_del_fat_file:
2454 0000A9D7 E808000000          <1>       call   delete_directory_entry
2455 0000A9DC 7205                <1>       jc     short loc_del_file_err_retn
2456                              <1>
2457                              <1> loc_delfile_unlink_cluster_chain:
2458 0000A9DE E864170000          <1>       call   truncate_cluster_chain
2459                              <1>       ;jc    short loc_del_file_err_retn
2460                              <1>
2461                              <1> loc_delfile_return:
2462                              <1> loc_del_file_err_retn:
2463 0000A9E3 C3                  <1>       retn
2464                              <1>
2465                              <1> delete_directory_entry:
2466                              <1>       ; 15/10/2016
2467                              <1>       ; 28/02/2016 (TRDOS 386 = TRDOS v2.0)
2468                              <1>       ; 01/08/2011 (DIR.ASM, 'proc_delete_directory_entry')
2469                              <1>       ; 10/04/2011
2470                              <1>       ; INPUT ->
2471                              <1>       ;      ESI = Logical Dos Drive Descripton Table Address
2472                              <1>       ;      EDI = Directory Buffer Entry Address
```

```
2473                                    <1>       ;      CX = Directory Buffer Entry Counter/Index
2474                                    <1>       ;      BL = Longname Entry Length
2475                                    <1>       ; OUTPUT ->
2476                                    <1>       ;      ESI = Logical dos drive descripton table address
2477                                    <1>       ;      EAX = First cluster to be truncated/unlinked
2478                                    <1>       ;       CF = 1 -> Error code in EAX (AL)
2479                                    <1>       ;       CF = 0 & BH <> 0 -> LMDT write error  (BH = 1)
2480                                    <1>       ;       CF = 0 & BL <> 0 -> Long name delete error (BL = FFh)
2481                                    <1>       ;
2482                                    <1>       ;  (EDI, EBX, ECX register contents will be changed)
2483                                    <1>
2484 0000A9E4 881D[325D0100]           <1>       mov    [DelFile_LNEL], bl
2485 0000A9EA 66890D[305D0100]         <1>       mov    [DelFile_EntryCounter], cx
2486                                    <1>
2487 0000A9F1 668B4714                 <1>       mov    ax, [edi+20] ; First Cluster High Word
2488 0000A9F5 C1E010                   <1>       shl    eax, 16
2489 0000A9F8 668B471A                 <1>       mov    ax, [edi+26] ; First Cluster Low Word
2490                                    <1>
2491 0000A9FC A3[2C5D0100]             <1>       mov    [DelFile_FCluster], eax
2492                                    <1>
2493                                    <1> loc_del_short_name:
2494 0000AA01 C607E5                   <1>       mov    byte [edi], 0E5h  ; Deleted sign
2495                                    <1>
2496 0000AA04 C605[F05A0100]02         <1>       mov    byte [DirBuff_ValidData], 2
2497 0000AA0B E85AFCFFFF               <1>       call   save_directory_buffer
2498 0000AA10 723D                     <1>       jc     short loc_delete_direntry_err_return
2499                                    <1>
2500                                    <1> loc_del_long_name:
2501 0000AA12 0FB615[325D0100]         <1>       movzx  edx, byte [DelFile_LNEL]
2502 0000AA19 08D2                     <1>       or     dl, dl
2503 0000AA1B 7416                     <1>       jz     short loc_del_dir_entry_update_parent_dir_lm_date
2504                                    <1>
2505 0000AA1D 8835[325D0100]           <1>       mov    byte [DelFile_LNEL], dh ; 0
2506                                    <1>
2507 0000AA23 0FB705[305D0100]         <1>       movzx  eax,  word [DelFile_EntryCounter]
2508 0000AA2A 29D0                     <1>       sub    eax, edx
2509                                    <1>       ;jnc   short loc_del_long_name_continue
2510 0000AA2C 7205                     <1>       jc     short loc_del_dir_entry_update_parent_dir_lm_date
2511                                    <1>
2512                                    <1> ;loc_del_direntry_inv_data_return: ; 15/10/2016 (0Dh -> 29)
2513                                    <1> ;     mov    eax, 29 ; 0Dh (TRDOS 8086) ; Invalid data
2514                                    <1> ;     retn
2515                                    <1>
2516                                    <1> loc_del_long_name_continue:
2517                                    <1>       ; AX = Directory Entry Number of the long name last entry
2518 0000AA2E E897FDFFFF               <1>       call   delete_longname
2519                                    <1>       ;jc    short loc_delete_direntry_err_return
2520                                    <1>
2521                                    <1> loc_del_dir_entry_update_parent_dir_lm_date:
2522 0000AA33 801D[325D0100]00         <1>       sbb    byte [DelFile_LNEL], 0 ; 0FFh if cf = 1
2523                                    <1>
2524 0000AA3A E8C6FCFFFF               <1>       call   update_parent_dir_lmdt
2525 0000AA3F B700                     <1>       mov    bh, 0
2526 0000AA41 80D700                   <1>       adc    bh, 0
2527                                    <1>
2528 0000AA44 8A1D[325D0100]           <1>       mov    bl, byte [DelFile_LNEL]
2529                                    <1>
2530                                    <1> loc_delete_direntry_return:
2531 0000AA4A A1[2C5D0100]             <1>       mov    eax, [DelFile_FCluster]
2532                                    <1> loc_delete_direntry_err_return:
2533 0000AA4F C3                       <1>       retn
2534                                    <1>
2535                                    <1> rename_directory_entry:
2536                                    <1>       ; 15/10/2016
2537                                    <1>       ; 06/03/2016 (TRDOS 386 = TRDOS v2.0)
2538                                    <1>       ; 01/08/2011 (DIR.ASM, 'proc_rename_directory_entry')
2539                                    <1>       ; 19/11/2010
2540                                    <1>       ; INPUT -> (Current Directory)
2541                                    <1>       ;      CX = Directory Entry Number
2542                                    <1>       ;       EAX = First Cluster number of file or directory
2543                                    <1>       ;       EBX = Longname Length (dir entry count) (< 256)
2544                                    <1>       ;       ESI = New file (or directory) name (no path).
2545                                    <1>       ;            (ASCIIZ string)
2546                                    <1>       ; OUTPUT ->
2547                                    <1>       ;      CF = 0 -> successfull
2548                                    <1>       ;      CF = 1 -> error code in EAX (AL)
2549                                    <1>       ;
2550                                    <1>       ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)
2551                                    <1>
2552 0000AA50 803D[C5520100]00         <1>       cmp    byte [Current_FATType], 0
2553 0000AA57 7706                     <1>       ja     short loc_rename_directory_entry
2554                                    <1>
2555 0000AA59 E8B90E0000               <1>       call   rename_fs_file_or_directory
2556 0000AA5E C3                       <1>       retn
2557                                    <1>
2558                                    <1> loc_rename_directory_entry:
2559 0000AA5F 881D[325D0100]           <1>       mov    [DelFile_LNEL], bl
2560 0000AA65 66890D[305D0100]         <1>       mov    [DelFile_EntryCounter], cx
2561 0000AA6C A3[2C5D0100]             <1>       mov    [DelFile_FCluster], eax
2562                                    <1>
2563 0000AA71 0FB7C1                   <1>       movzx  eax, cx
2564 0000AA74 E8BBFDFFFF               <1>       call   locate_current_dir_entry
2565 0000AA79 7308                     <1>       jnc    short loc_rename_direntry_check_fcluster
2566                                    <1>
2567                                    <1> loc_rename_direntry_pop_retn:
2568 0000AA7B C3                       <1>       retn
2569                                    <1>
2570                                    <1> loc_rename_direntry_pop_invd_retn:
2571 0000AA7C F9                       <1>       stc
2572                                    <1> loc_rename_direntry_invd_retn:
2573                                    <1>       ; 15/10/2016 (0Dh -> 29)
2574 0000AA7D B81D000000               <1>       mov    eax, 29 ; Invalid data
```

```
                                <1> loc_rename_retn:
0000AA82 C3                     <1>         retn
                                <1>
                                <1> loc_rename_direntry_check_fcluster:
0000AA83 668B5714               <1>         mov    dx, [edi+20] ; First Cluster HW
0000AA87 66C1E210               <1>         shl    dx, 16
0000AA8B 668B571A               <1>         mov    dx, [edi+26] ; First Cluster LW
0000AA8F 3B15[2C5D0100]         <1>         cmp    edx, [DelFile_FCluster]
0000AA95 75E5                   <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.)
0000AA97 E89FF6FFFF             <1>         call   convert_file_name
                                <1>
0000AA9C C605[F05A0100]02       <1>           mov     byte [DirBuff_ValidData], 2
0000AAA3 E8C2FBFFFF             <1>         call   save_directory_buffer
0000AAA8 72D8                   <1>         jc     short loc_rename_retn
                                <1>
                                <1> loc_rename_direntry_del_ln:
0000AAAA 0FB615[325D0100]       <1>         movzx  edx, byte [DelFile_LNEL]
0000AAB1 08D2                   <1>         or     dl, dl
0000AAB3 7410                   <1>         jz     short loc_rename_direntry_update_parent_dir_lm_date
                                <1>
0000AAB5 0FB705[305D0100]       <1>         movzx  eax, word [DelFile_EntryCounter]
0000AABC 29D0                   <1>         sub    eax, edx
0000AABE 72BD                   <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
0000AAC0 E805FDFFFF             <1>         call   delete_longname
                                <1>
                                <1> loc_rename_direntry_update_parent_dir_lm_date:
0000AAC5 E83BFCFFFF             <1>         call   update_parent_dir_lmdt
0000AACA 31C0                   <1>         xor    eax, eax
0000AACC 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>
0000AACD 3C02                   <1>         cmp    al, 2
0000AACF 0F847F010000           <1>         je     msftdf_df2_check_directory
0000AAD5 A2[B25E0100]           <1>         mov    [move_cmd_phase], al
                                <1>
                                <1> msftdf_parse_sf_path:
                                <1>         ; ESI = ASCIIZ pathname (Source)
0000AADA 57                     <1>         push   edi
0000AADB BF[B05D0100]           <1>         mov    edi, SourceFile_Drv
0000AAE0 E821F7FFFF             <1>         call   parse_path_name
0000AAE5 5E                     <1>         pop    esi
0000AAE6 7211                   <1>         jc     short msftdf_psf_retn
                                <1>
                                <1> msftdf_parse_df_path:
                                <1>         ; ESI = ASCIIZ pathname   (Destination)
0000AAE8 BF[305E0100]           <1>         mov    edi, DestinationFile_Drv
0000AAED E814F7FFFF             <1>         call   parse_path_name
0000AAF2 7306                   <1>         jnc    short msftdf_check_sf_drv
                                <1>
0000AAF4 3C01                   <1>         cmp    al, 1 ; File or directory name is not existing
0000AAF6 7602                   <1>         jna    short msftdf_check_sf_drv
                                <1>
```

```
2677                                    <1> msftdf_stc_retn:
2678 0000AAF8 F9                       <1>         stc
2679                                    <1> msftdf_psf_retn:
2680 0000AAF9 C3                       <1>         retn
2681                                    <1>
2682                                    <1> msftdf_check_sf_drv:
2683 0000AAFA A0[B05D0100]             <1>         mov    al, [SourceFile_Drv]
2684                                    <1>
2685                                    <1> msftdf_check_df_drv:
2686 0000AAFF 8A15[305E0100]           <1>         mov    dl, [DestinationFile_Drv]
2687                                    <1>
2688                                    <1> msftdf_compare_sf_df_drv:
2689 0000AB05 29DB                     <1>         sub    ebx, ebx
2690 0000AB07 8A3D[C6520100]           <1>         mov    bh, [Current_Drv]
2691 0000AB0D 38C2                     <1>         cmp    dl, al
2692 0000AB0F 7409                     <1>         je     short msftdf_check_sf_df_drv_ok
2693                                    <1>
2694                                    <1> msftdf_not_same_drv:
2695                                    <1>           ; DL = source file's drive number
2696 0000AB11 88C6                     <1>         mov    dh, al ; destination file's drive number
2697                                    <1>         ; 15/10/2016 (11h -> 21)
2698 0000AB13 B815000000               <1>         mov    eax, 21 ; Not the same drive
2699 0000AB18 F9                       <1>         stc
2700 0000AB19 C3                       <1>         retn
2701                                    <1>
2702                                    <1> msftdf_check_sf_df_drv_ok:
2703 0000AB1A 8815[B35E0100]           <1>         mov    [msftdf_sf_df_drv], dl
2704                                    <1>
2705 0000AB20 29C0                     <1>           sub  eax, eax
2706 0000AB22 88D4                     <1>         mov    ah, dl
2707 0000AB24 0500010900               <1>         add    eax, Logical_DOSDisks
2708 0000AB29 A3[B45E0100]             <1>         mov    [msftdf_drv_offset], eax
2709                                    <1>
2710 0000AB2E 38FA                     <1>         cmp    dl, bh ; byte [Current_Drv]
2711 0000AB30 7407                     <1>         je     short msftdf_df_check_directory
2712                                    <1>
2713                                    <1> msftdf_change_drv:
2714 0000AB32 E826C1FFFF               <1>         call   change_current_drive
2715 0000AB37 726D                     <1>         jc     short msftdf_df_error_retn
2716                                    <1>
2717                                    <1> msftdf_check_destination_file:
2718                                    <1> msftdf_df_check_directory:
2719 0000AB39 BE[315E0100]             <1>         mov    esi, DestinationFile_Directory
2720 0000AB3E 803E20                   <1>         cmp    byte [esi], 20h
2721 0000AB41 760F                     <1>         jna    short msftdf_df_find_1
2722                                    <1>
2723                                    <1> msftdf_df_change_directory:
2724 0000AB43 FE05[AD060100]           <1>         inc    byte [Restore_CDIR]
2725 0000AB49 30E4                     <1>         xor    ah, ah ; CD_COMMAND sign -> 0
2726 0000AB4B E8A2F0FFFF               <1>         call   change_current_directory
2727 0000AB50 7254                     <1>         jc     short msftdf_df_error_retn
2728                                    <1>
2729                                    <1> ;msftdf_df_change_prompt_dir_string:
2730                                    <1> ;       call   change_prompt_dir_string
2731                                    <1>
2732                                    <1> msftdf_df_find_1:
2733 0000AB52 BE[725E0100]             <1>           mov     esi, DestinationFile_Name
2734 0000AB57 803E20                   <1>         cmp    byte [esi], 20h
2735 0000AB5A 7631                     <1>         jna    short msftdf_df_copy_sf_name
2736                                    <1>
2737                                    <1> msftdf_df_find_2:
2738 0000AB5C 6631C0                   <1>         xor    ax, ax ; DestinationFile_AttributesMask -> any/zero
2739 0000AB5F E88BD4FFFF               <1>         call   find_first_file
2740 0000AB64 0F838D000000             <1>         jnc    msftdf_permission_denied_retn
2741                                    <1>
2742                                    <1> msftdf_df_check_error_code:
2743                                    <1>         ;cmp   eax, 2 ; File not found error
2744 0000AB6A 3C02                     <1>         cmp    al, 2
2745 0000AB6C 7537                     <1>         jne    short msftdf_df_stc_retn
2746                                    <1>
2747                                    <1> msftdf_df_check_fname:
2748                                    <1>         ; 15/10/2016
2749 0000AB6E BE[725E0100]             <1>         mov    esi, DestinationFile_Name ; *
2750 0000AB73 E837D8FFFF               <1>         call   check_filename
2751 0000AB78 7307                     <1>         jnc    short msftdf_convert_df_direntry_name
2752                                    <1>         ; invalid file name chars !
2753 0000AB7A B81A000000               <1>         mov    eax, ERR_INV_FILE_NAME  ; 26
2754 0000AB7F EB24                     <1>         jmp    short msftdf_df_stc_retn
2755                                    <1>
2756                                    <1> msftdf_convert_df_direntry_name:
2757                                    <1>         ; mov  esi, DestinationFile_Name ; *
2758 0000AB81 BF[825E0100]             <1>         mov    edi, DestinationFile_DirEntry
2759 0000AB86 E8B0F5FFFF               <1>         call   convert_file_name
2760 0000AB8B EB1A                     <1>         jmp    short msftdf_restore_current_dir_1
2761                                    <1>
2762                                    <1> msftdf_df_copy_sf_name:
2763 0000AB8D 89F7                     <1>         mov    edi, esi
2764 0000AB8F 57                       <1>         push   edi
2765 0000AB90 BE[F25D0100]             <1>           mov     esi, SourceFile_Name
2766 0000AB95 B90C000000               <1>         mov    ecx, 12
2767                                    <1> msftdf_df_copy_sf_name_loop:
2768 0000AB9A AC                       <1>         lodsb
2769 0000AB9B AA                       <1>           stosb
2770 0000AB9C 08C0                     <1>         or     al, al
2771 0000AB9E 7402                     <1>         jz     short msftdf_df_copy_sf_name_ok
2772 0000ABA0 E2F8                     <1>           loop    msftdf_df_copy_sf_name_loop
2773                                    <1> msftdf_df_copy_sf_name_ok:
2774 0000ABA2 5E                       <1>         pop    esi
2775 0000ABA3 EBB7                     <1>         jmp    short msftdf_df_find_2
2776                                    <1>
2777                                    <1> msftdf_df_stc_retn:
2778 0000ABA5 F9                       <1>         stc
```

```
2779                                    <1> msftdf_restore_cdir_failed:
2780                                    <1> msftdf_df_error_retn:
2781 0000ABA6 C3                       <1>       retn
2782                                    <1>
2783                                    <1> msftdf_restore_current_dir_1:
2784 0000ABA7 803D[AD060100]00          <1>       cmp    byte [Restore_CDIR], 0
2785 0000ABAE 760D                     <1>       jna    short msftdf_sf_check_directory
2786 0000ABB0 8B35[B45E0100]           <1>       mov    esi, [msftdf_drv_offset]
2787 0000ABB6 E859C1FFFF               <1>       call   restore_current_directory
2788 0000ABBB 72E9                     <1>       jc     short msftdf_restore_cdir_failed
2789                                    <1>
2790                                    <1> msftdf_sf_check_directory:
2791 0000ABBD BE[B15D0100]             <1>       mov    esi, SourceFile_Directory
2792 0000ABC2 803E20                   <1>       cmp    byte [esi], 20h
2793 0000ABC5 760F                     <1>       jna    short msftdf_sf_find
2794                                    <1> msftdf_sf_change_directory:
2795 0000ABC7 FE05[AD060100]           <1>       inc    byte [Restore_CDIR]
2796 0000ABCD 30E4                     <1>       xor    ah, ah ; CD_COMMAND sign -> 0
2797 0000ABCF E81EF0FFFF               <1>       call   change_current_directory
2798 0000ABD4 7227                     <1>       jc     short msftdf_return
2799                                    <1>
2800                                    <1> ;msftdf_sf_change_prompt_dir_string:
2801                                    <1> ;     call   change_prompt_dir_string
2802                                    <1>
2803                                    <1> msftdf_sf_find:
2804 0000ABD6 BE[F25D0100]             <1>         mov     esi, SourceFile_Name  ; Offset 66
2805 0000ABDB 66B80018                 <1>       mov    ax, 1800h ; Only files
2806 0000ABDF E80BD4FFFF               <1>       call   find_first_file
2807 0000ABE4 7217                     <1>       jc     short msftdf_return
2808                                    <1>
2809                                    <1> msftdf_sf_ambgfn_check:
2810 0000ABE6 6609D2                   <1>       or     dx, dx ; Ambiguous filename chars used sign (DX>0)
2811 0000ABE9 7407                     <1>       jz     short msftdf_sf_found
2812                                    <1>
2813                                    <1> msftdf_ambiguous_file_name_error:
2814 0000ABEB B802000000               <1>       mov    eax, 2 ; File not found error
2815 0000ABF0 F9                       <1>       stc
2816 0000ABF1 C3                       <1>       retn
2817                                    <1>
2818                                    <1> msftdf_sf_found:
2819 0000ABF2 80E31F                   <1>       and    bl, 1Fh ; Attributes, D-V-S-H-R
2820 0000ABF5 7416                     <1>       jz     short msftdf_save_sf_structure
2821                                    <1>
2822                                    <1> msftdf_permission_denied_retn:
2823 0000ABF7 B805000000               <1>       mov    eax, 05h ; Access (Permission) denied !
2824 0000ABFC F9                       <1>       stc
2825                                    <1> msftdf_rest_cdir_err_retn:
2826                                    <1> msftdf_return:
2827 0000ABFD C3                       <1>       retn
2828                                    <1>
2829                                    <1> msftdf_phase_1_return:
2830 0000ABFE 31C0                     <1>       xor    eax, eax
2831 0000AC00 A2[B25E0100]             <1>       mov    [move_cmd_phase], al ; 0
2832 0000AC05 FEC0                     <1>       inc    al ; mov al, 1
2833 0000AC07 BB[54AC0000]             <1>       mov    ebx, msftdf_df2_check_directory
2834                                    <1>       ;mov   edx, 0FFFFFFFFh
2835 0000AC0C C3                       <1>       retn
2836                                    <1>
2837                                    <1> msftdf_save_sf_structure:
2838 0000AC0D BE[BC5C0100]             <1>       mov    esi, FindFile_DirEntry
2839 0000AC12 BF[025E0100]             <1>       mov    edi, SourceFile_DirEntry
2840 0000AC17 B908000000               <1>       mov    ecx, 8
2841 0000AC1C F3A5                     <1>       rep    movsd
2842                                    <1>
2843                                    <1> msftdf_df_copy_sf_parameters:
2844 0000AC1E BE0B000000               <1>       mov    esi, 11
2845 0000AC23 89F7                     <1>       mov    edi, esi
2846 0000AC25 81C6[025E0100]           <1>       add    esi, SourceFile_DirEntry
2847 0000AC2B 81C7[825E0100]           <1>       add    edi, DestinationFile_DirEntry
2848                                    <1>       ;mov   ecx, 21
2849 0000AC31 B115                     <1>       mov    cl, 21
2850 0000AC33 F3A4                     <1>       rep    movsb
2851                                    <1>
2852                                    <1> msftdf_restore_current_dir_2:
2853 0000AC35 803D[AD060100]00          <1>       cmp    byte [Restore_CDIR], 0
2854 0000AC3C 760D                     <1>       jna    short msftdf_df2_check_move_cmd_phase
2855 0000AC3E 8B35[B45E0100]           <1>       mov    esi, [msftdf_drv_offset]
2856 0000AC44 E8CBC0FFFF               <1>       call   restore_current_directory
2857 0000AC49 72B2                     <1>       jc     short msftdf_rest_cdir_err_retn
2858                                    <1>
2859                                    <1> msftdf_df2_check_move_cmd_phase:
2860 0000AC4B 803D[B25E0100]01          <1>       cmp    byte [move_cmd_phase], 1
2861 0000AC52 74AA                     <1>       je     short msftdf_phase_1_return
2862                                    <1>
2863                                    <1> msftdf_df2_check_directory:
2864 0000AC54 BE[315E0100]             <1>       mov    esi, DestinationFile_Directory
2865 0000AC59 803E20                   <1>       cmp    byte [esi], 20h
2866 0000AC5C 760F                     <1>       jna    short msftdf_make_dfde_locate_ffe_on_directory
2867                                    <1> msftdf_df2_change_directory:
2868 0000AC5E FE05[AD060100]           <1>       inc    byte [Restore_CDIR]
2869 0000AC64 30E4                     <1>       xor    ah, ah ; CD_COMMAND sign -> 0
2870 0000AC66 E887EFFFFF               <1>       call   change_current_directory
2871 0000AC6B 7290                     <1>       jc     short msftdf_return
2872                                    <1>
2873                                    <1> ;msftdf_df2_change_prompt_dir_string:
2874                                    <1> ;     call   change_prompt_dir_string
2875                                    <1>
2876                                    <1> msftdf_make_dfde_locate_ffe_on_directory:
2877                                    <1>       ; Current directory fcluster <> Directory buffer cluster
2878                                    <1>       ; Current directory will be reloaded by
2879                                    <1>       ; 'locate_current_dir_file' procedure
2880                                    <1>       ;
```

```
2881                                    <1>       ;xor   ax, ax
2882 0000AC6D 31C0                      <1>       xor    eax, eax
2883 0000AC6F 89C1                      <1>       mov    ecx, eax
2884 0000AC71 6649                      <1>       dec    cx ; FFFFh
2885                                    <1>              ; CX = FFFFh -> find first deleted or free entry
2886                                    <1>              ; ESI would be ASCIIZ filename address if the call
2887                                    <1>              ; would not be for first free or deleted dir entry
2888 0000AC73 E8CEF1FFFF                <1>       call   locate_current_dir_file
2889 0000AC78 733F                      <1>       jnc    msftdf_make_dfde_set_ff_dir_entry
2890                                    <1>
2891                                    <1>       ;cmp   eax, 2
2892 0000AC7A 3C02                      <1>         cmp  al, 2
2893 0000AC7C 7537                      <1>       jne    short msftdf_error_retn
2894                                    <1>
2895                                    <1> msftdf_add_new_dir_entry_check_fs:
2896 0000AC7E 8B35[B45E0100]            <1>       mov    esi, [msftdf_drv_offset]
2897 0000AC84 A1[F55A0100]              <1>       mov    eax, [DirBuff_Cluster]
2898 0000AC89 807E0300                  <1>       cmp    byte [esi+LD_FATType], 0
2899 0000AC8D 7711                      <1>       ja     short msftdf_add_new_subdir_cluster
2900                                    <1>
2901                                    <1> msftdf_add_new_fs_subdir_section:
2902                                    <1>       ;CL=0, CH=E5h --> deleted entry, CH=0 --> free entry
2903                                    <1>         ;xor cx, cx
2904 0000AC8F 30ED                      <1>       xor    ch, ch ; cx = 0 --> add a new subdir section
2905 0000AC91 E8830C0000                <1>       call   add_new_fs_section
2906 0000AC96 721E                      <1>         jc   short msftdf_dsfde_error_retn
2907                                    <1>       ;mov   [createfile_LastDirCluster], eax
2908                                    <1>
2909 0000AC98 E8A30E0000                <1>       call   load_FS_sub_directory
2910                                    <1>       ;mov   ebx, Directory_Buffer
2911 0000AC9D 7318                      <1>       jnc    short msftdf_add_new_fs_subdir_section_ok
2912 0000AC9F C3                        <1>       retn
2913                                    <1>
2914                                    <1> msftdf_add_new_subdir_cluster:
2915 0000ACA0 E881150000                <1>       call   add_new_cluster
2916 0000ACA5 720F                      <1>       jc     short msftdf_dsfde_error_retn
2917                                    <1>
2918                                    <1>       ;mov   [createfile_LastDirCluster], eax
2919                                    <1>
2920 0000ACA7 E8570E0000                <1>       call   load_FAT_sub_directory
2921 0000ACAC 7309                      <1>       jnc    short msftdf_add_new_subdir_cluster_ok
2922                                    <1>       ; EBX = Directory buffer address
2923                                    <1>
2924                                    <1> msftdf_ansdc_update_parent_dir_lmdt:
2925                                    <1> msftdf_make_dfde_err_upd_pdir_lmdt:
2926 0000ACAE 50                        <1>       push   eax
2927 0000ACAF E851FAFFFF                <1>       call   update_parent_dir_lmdt
2928 0000ACB4 58                        <1>       pop    eax
2929                                    <1>
2930                                    <1> msftdf_error_retn:
2931 0000ACB5 F9                        <1>       stc
2932                                    <1> msftdf_dsfde_restore_cdir_failed:
2933                                    <1> msftdf_dsfde_error_retn:
2934 0000ACB6 C3                        <1>       retn
2935                                    <1>
2936                                    <1> msftdf_add_new_fs_subdir_section_ok:
2937                                    <1> msftdf_add_new_subdir_cluster_ok:
2938 0000ACB7 89DF                      <1>       mov    edi, ebx ; Directory buffer address
2939                                    <1>
2940                                    <1> msftdf_make_dfde_set_ff_dir_entry:
2941 0000ACB9 8B15[C0520100]            <1>       mov    edx, [Current_Dir_FCluster]
2942 0000ACBF 8915[185F0100]            <1>       mov    [createfile_FFCluster], edx
2943                                    <1>       ; EDI = Directory entry offset
2944 0000ACC5 BE[825E0100]              <1>       mov    esi, DestinationFile_DirEntry
2945 0000ACCA B908000000                <1>       mov    ecx, 8
2946 0000ACCF F3A5                      <1>       rep    movsd
2947                                    <1>
2948 0000ACD1 C605[F05A0100]02          <1>       mov    byte [DirBuff_ValidData], 2
2949 0000ACD8 E88DF9FFFF                <1>       call   save_directory_buffer
2950 0000ACDD 72CF                      <1>       jc     short msftdf_make_dfde_err_upd_pdir_lmdt
2951                                    <1>
2952                                    <1> msftdf_make_dfde_update_pdir_lmdt:
2953 0000ACDF E821FAFFFF                <1>       call   update_parent_dir_lmdt
2954                                    <1>
2955                                    <1> msftdf_dsfde_restore_current_dir_1:
2956 0000ACE4 803D[AD060100]00          <1>       cmp    byte [Restore_CDIR], 0
2957 0000ACEB 760D                      <1>       jna    short msftdf_dsfde_check_directory
2958 0000ACED 8B35[B45E0100]            <1>       mov    esi, [msftdf_drv_offset]
2959 0000ACF3 E81CC0FFFF                <1>       call   restore_current_directory
2960 0000ACF8 72BC                      <1>       jc     short msftdf_dsfde_restore_cdir_failed
2961                                    <1>
2962                                    <1> msftdf_dsfde_check_directory:
2963 0000ACFA BE[B15D0100]              <1>       mov    esi, SourceFile_Directory
2964 0000ACFF 803E20                    <1>       cmp    byte [esi], 20h
2965 0000AD02 760F                      <1>       jna    short msftdf_dsfde_find_file
2966                                    <1>
2967                                    <1> msftdf_dsfde_change_directory:
2968 0000AD04 FE05[AD060100]            <1>       inc    byte [Restore_CDIR]
2969 0000AD0A 28E4                      <1>       sub    ah, ah ; CD_COMMAND sign -> 0
2970 0000AD0C E8E1EEFFFF                <1>       call   change_current_directory
2971 0000AD11 72A3                      <1>       jc     short msftdf_dsfde_error_retn
2972                                    <1>
2973                                    <1> ;msftdf_dsfde_sf_change_prompt_dir_string:
2974                                    <1> ;     call   change_prompt_dir_string
2975                                    <1>
2976                                    <1> msftdf_dsfde_find_file:
2977 0000AD13 BE[F25D0100]              <1>       mov    esi, SourceFile_Name  ; Offset 66
2978 0000AD18 668B460E                  <1>       mov    ax, [esi+14] ; 80 -> SourceFile_AttributesMask
2979 0000AD1C E8CED2FFFF                <1>       call   find_first_file
2980 0000AD21 7293                      <1>       jc     short msftdf_dsfde_error_retn
2981                                    <1>
2982                                    <1> msftdf_dsfde_delete_direntry:
```

```
2983 0000AD23 8B35[B45E0100]      <1>       mov    esi, [msftdf_drv_offset]
2984                              <1>
2985 0000AD29 807E0300            <1>       cmp    byte [esi+LD_FATType], 0
2986 0000AD2D 770A                <1>       ja     short msftdf_delete_FAT_direntry
2987                              <1>
2988 0000AD2F 30DB                <1>       xor    bl, bl
2989                              <1>       ; BL = 0 -> File
2990                              <1>       ; EDI -> Directory buffer entry offset/address
2991 0000AD31 E8E40B0000          <1>       call   delete_fs_directory_entry
2992 0000AD36 7315                <1>       jnc    short msftdf_dsfde_restore_current_dir_2
2993 0000AD38 C3                  <1>       retn
2994                              <1>
2995                              <1> msftdf_delete_FAT_direntry:
2996 0000AD39 8A1D[B95C0100]      <1>       mov    bl, [FindFile_LongNameEntryLength]
2997 0000AD3F 668B0D[E45C0100]    <1>       mov    cx, [FindFile_DirEntryNumber]
2998                              <1>       ; ESI = Logical DOS drive description table address
2999                              <1>       ; EDI = Directory buffer entry offset/address
3000 0000AD46 E899FCFFFF          <1>       call   delete_directory_entry
3001 0000AD4B 721C                <1>       jc     short msftdf_retn
3002                              <1>
3003                              <1> msftdf_dsfde_restore_current_dir_2:
3004 0000AD4D 803D[AD060100]00    <1>       cmp    byte [Restore_CDIR], 0
3005 0000AD54 7607                <1>       jna    short msftdf_new_dir_fcluster_retn
3006                              <1>       ;mov   esi, [msftdf_drv_offset]
3007 0000AD56 E8B9BFFFFF          <1>       call   restore_current_directory
3008 0000AD5B 720C                <1>       jc     short msftdf_retn
3009                              <1>
3010                              <1> msftdf_new_dir_fcluster_retn:
3011 0000AD5D 31C9                <1>       xor    ecx, ecx
3012 0000AD5F A1[185F0100]        <1>       mov    eax, [createfile_FFCluster]
3013 0000AD64 BB[305E0100]        <1>       mov    ebx, DestinationFile_Drv
3014                              <1>
3015                              <1> msftdf_retn:
3016 0000AD69 C3                  <1>       retn
3017                              <1>
3018                              <1>
3019                              <1> copy_source_file_to_destination_file:
3020                              <1>       ; 17/10/2016
3021                              <1>       ; 16/10/2016
3022                              <1>       ; 15/10/2016
3023                              <1>       ; 30/03/2016, 31/03/2016
3024                              <1>       ; 24/03/2016, 25/03/2016, 28/03/2016
3025                              <1>       ; 21/03/2016, 22/03/2016, 23/03/2016
3026                              <1>       ; 16/03/2016, 17/03/2016, 18/03/2016
3027                              <1>       ; 15/03/2016 (TRDOS 386 = TRDOS v2.0)
3028                              <1>       ; 02/09/2011 (FILE.ASM 'copy_source_file_to_destination_file')
3029                              <1>       ; 01/08/2010 - 18/05/2011
3030                              <1>       ;
3031                              <1>       ;   Command Interpreter phase 1 enter ->
3032                              <1>       ;           AL = 1 -> Caller is command interpreter
3033                              <1>       ;           AL = 2 -> The second call, re-enter/continue
3034                              <1>       ;   Phase 1 -> Check source file
3035                              <1>       ;              'found' is required
3036                              <1>       ;   Phase 2 -> Check destination file,
3037                              <1>       ;              save 'found' or 'not found' status
3038                              <1>       ;              'permission denied' error will be return
3039                              <1>       ;              if attributes have not for ordinary file
3040                              <1>       ;              without readonly attribute
3041                              <1>       ;   Command Interpreter phase 1 return ->
3042                              <1>       ;              DH = Source file attributes
3043                              <1>       ;              DL = Destination file found status
3044                              <1>       ;              EAX = 0
3045                              <1>       ;   Command Interpreter phase 2 enter ->
3046                              <1>       ;              AL = 2 -> Continue from the last position
3047                              <1>       ;              AH =
3048                              <1>       ;   Phase 3 -> Load source file or use read/write cluster method
3049                              <1>       ;   Phase 4 -> Create destination file if it is not found
3050                              <1>       ;   Phase 5 -> Open destination file
3051                              <1>       ;   Phase 6 -> Read from source and write to destination
3052                              <1>       ;   Phase 7 -> Unload source file, if it is loaded at memory
3053                              <1>       ;       cf = 1 causes to return before the phase 7
3054                              <1>       ;             but loaded file will be unloaded
3055                              <1>       ;            (allocated memory block will be deallocated)
3056                              <1>       ;
3057                              <1>       ; INPUT ->
3058                              <1>       ;      ESI = Source File Pathname (Asciiz)
3059                              <1>       ;       EDI = Destination File Pathname (Asciiz)
3060                              <1>       ;       AL = 0 --> Interrupt (System call)
3061                              <1>       ;       AL > 0 --> Command Interpreter (Question)
3062                              <1>       ;       AL = 1 --> Question Phase
3063                              <1>       ;       AL = 2 --> Progress Phase
3064                              <1>       ;
3065                              <1>       ; OUTPUT ->
3066                              <1>       ;     cf = 0 -> OK
3067                              <1>       ;     EAX = Destination file first cluster
3068                              <1>       ;
3069                              <1>       ;       CL > 0 if there is file reading error before EOF
3070                              <1>       ;            (incomplete copy)
3071                              <1>       ;       CH > 0 if file is (full) loaded at memory
3072                              <1>       ;
3073                              <1>       ;     cf = 1 -> Error code in AL (EAX)
3074                              <1>       ;
3075                              <1>       ; (EBX, ECX, ESI, EDI register contents will be changed)
3076                              <1>
3077                              <1>
3078 0000AD6A 3C02                <1>       cmp    al, 2
3079 0000AD6C 0F845A020000        <1>       je     csftdf2_check_cdrv
3080                              <1>
3081                              <1> ; Phase 1
3082                              <1>
3083 0000AD72 A2[D85E0100]        <1>       mov    byte [copy_cmd_phase], al
3084                              <1>
```

```
3085 0000AD77 57                <1>       push   edi ; *
3086                            <1>
3087                            <1> csftdf_parse_sf_path:
3088 0000AD78 BF[B05D0100]      <1>       mov    edi, SourceFile_Drv
3089 0000AD7D E884F4FFFF        <1>       call   parse_path_name
3090 0000AD82 721C              <1>       jc     short csftdf_parse_sf_path_failed
3091                            <1>
3092                            <1> csftdf_parse_df_path:
3093 0000AD84 5E                <1>       pop    esi ; * (pushed edi)
3094                            <1>
3095                            <1> csftdf_sf_check_filename_exists:
3096 0000AD85 803D[F25D0100]21  <1>       cmp    byte [SourceFile_Name], 21h
3097 0000AD8C 7215              <1>       jb     short csftdf_sf_file_not_found_error
3098                            <1>
3099 0000AD8E BF[305E0100]      <1>       mov    edi, DestinationFile_Drv
3100 0000AD93 E86EF4FFFF        <1>       call   parse_path_name
3101 0000AD98 7310              <1>       jnc    short csftdf_check_sf_cdrv
3102                            <1>
3103 0000AD9A 3C01              <1>       cmp    al, 1 ; File or directory name is not existing
3104 0000AD9C 760C              <1>       jna    short csftdf_check_sf_cdrv
3105                            <1>
3106                            <1> csftdf_parse_df_path_failed:
3107 0000AD9E F9                <1>       stc
3108                            <1> csftdf_sf_error_retn:
3109 0000AD9F C3                <1>       retn
3110                            <1>
3111                            <1> csftdf_parse_sf_path_failed:
3112 0000ADA0 5F                <1>       pop    edi ; *
3113 0000ADA1 EBFC              <1>       jmp    short csftdf_sf_error_retn
3114                            <1>
3115                            <1> csftdf_sf_file_not_found_error:
3116 0000ADA3 B802000000        <1>       mov    eax, 2 ; File not found
3117 0000ADA8 EBF5              <1>       jmp    short csftdf_sf_error_retn
3118                            <1>
3119                            <1> csftdf_check_sf_cdrv:
3120 0000ADAA 8A3D[C6520100]    <1>       mov    bh, [Current_Drv]
3121                            <1>
3122 0000ADB0 883D[DB5E0100]    <1>       mov    [csftdf_cdrv], bh ; 23/03/2016
3123                            <1>
3124 0000ADB6 8A15[B05D0100]    <1>       mov    dl, [SourceFile_Drv]
3125 0000ADBC 38FA              <1>       cmp    dl, bh ; byte [Current_Drv]
3126 0000ADBE 7407              <1>       je     short csftdf_sf_check_directory
3127                            <1>
3128 0000ADC0 E898BEFFFF        <1>       call   change_current_drive
3129 0000ADC5 72D8              <1>       jc     short csftdf_sf_error_retn
3130                            <1>
3131                            <1> csftdf_sf_check_directory:
3132 0000ADC7 BE[B15D0100]      <1>       mov    esi, SourceFile_Directory
3133 0000ADCC 803E20            <1>       cmp    byte [esi], 20h
3134 0000ADCF 760F              <1>       jna    short csftdf_find_sf
3135                            <1>
3136                            <1> csftdf_sf_change_directory:
3137 0000ADD1 FE05[AD060100]    <1>       inc    byte [Restore_CDIR]
3138 0000ADD7 30E4              <1>       xor    ah, ah ; CD_COMMAND sign -> 0
3139 0000ADD9 E814EEFFFF        <1>       call   change_current_directory
3140 0000ADDE 72BF              <1>       jc     short csftdf_sf_error_retn
3141                            <1>
3142                            <1> ;csftdf_sf_change_prompt_dir_string:
3143                            <1> ;     call   change_prompt_dir_string
3144                            <1>
3145                            <1> csftdf_find_sf:
3146 0000ADE0 BE[F25D0100]      <1>       mov    esi, SourceFile_Name
3147 0000ADE5 66B80018          <1>       mov    ax, 1800h ; Except volume label and dirs
3148 0000ADE9 E801D2FFFF        <1>       call   find_first_file
3149 0000ADEE 72AF              <1>       jc     short csftdf_sf_error_retn
3150                            <1>
3151                            <1> csftdf_sf_ambgfn_check:
3152 0000ADF0 6621D2            <1>       and    dx, dx ; Ambiguous filename chars used sign (DX>0)
3153 0000ADF3 7407              <1>       jz     short csftdf_sf_found
3154                            <1>
3155                            <1> csftdf_ambiguous_file_name_error:
3156 0000ADF5 B802000000        <1>       mov    eax, 2 ; File not found error
3157 0000ADFA F9                <1>       stc
3158 0000ADFB C3                <1>       retn
3159                            <1>
3160                            <1> csftdf_sf_found:
3161 0000ADFC A3[DC5E0100]      <1>       mov    [csftdf_filesize], eax
3162                            <1>
3163 0000AE01 09C0              <1>       or     eax, eax
3164 0000AE03 7507              <1>       jnz    short csftdf_set_source_file_direntry
3165                            <1>
3166                            <1> csftdf_sf_file_size_zero:
3167 0000AE05 B814000000        <1>       mov    eax, 20 ; TRDOS zero length (file size) error
3168 0000AE0A F9                <1>       stc
3169 0000AE0B C3                <1>       retn
3170                            <1>
3171                            <1> csftdf_set_source_file_direntry:
3172 0000AE0C BE[BC5C0100]      <1>       mov    esi, FindFile_DirEntry
3173 0000AE11 BF[025E0100]      <1>       mov    edi, SourceFile_DirEntry
3174 0000AE16 B908000000        <1>       mov    ecx, 8
3175 0000AE1B F3A5              <1>       rep    movsd
3176                            <1>
3177                            <1> csftdf_sf_restore_cdrv:
3178                            <1>       ; 22/03/2016
3179 0000AE1D 8A15[DB5E0100]    <1>       mov    dl, [csftdf_cdrv]
3180 0000AE23 3A15[C6520100]    <1>       cmp    dl, [Current_Drv]
3181 0000AE29 7407              <1>       je     short csftdf_sf_restore_cdir
3182 0000AE2B E82DBEFFFF        <1>       call   change_current_drive
3183 0000AE30 724F              <1>       jc     short csftdf_df_error_retn ; 30/03/2016
3184                            <1>
3185                            <1> csftdf_sf_restore_cdir:
3186 0000AE32 803D[AD060100]00  <1>       cmp    byte [Restore_CDIR], 0
```

```
3187 0000AE39 7612                <1>       jna    short csftdf_df_check_filename_exists
3188 0000AE3B 29C0                <1>       sub    eax, eax
3189 0000AE3D BE00010900          <1>       mov    esi, Logical_DOSDisks
3190 0000AE42 88D4                <1>       mov    ah, dl ; byte [csftdf_cdrv]
3191 0000AE44 01C6                <1>       add    esi, eax
3192 0000AE46 E8C9BEFFFF          <1>       call   restore_current_directory
3193 0000AE4B 7234                <1>       jc     short csftdf_df_error_retn
3194                              <1>
3195                              <1> csftdf_df_check_filename_exists:
3196 0000AE4D 803D[725E0100]20    <1>       cmp    byte [DestinationFile_Name], 20h
3197 0000AE54 7716                <1>       ja     short csftdf_check_df_cdrv
3198                              <1>
3199                              <1> csftdf_copy_sf_name:
3200 0000AE56 BF[725E0100]        <1>       mov    edi, DestinationFile_Name
3201 0000AE5B BE[F25D0100]        <1>       mov    esi, SourceFile_Name
3202 0000AE60 B10C                <1>       mov    cl, 12
3203                              <1>
3204                              <1> csftdf_df_copy_sf_name_loop:
3205 0000AE62 AC                  <1>       lodsb
3206 0000AE63 AA                  <1>       stosb
3207 0000AE64 08C0                <1>       or     al, al
3208 0000AE66 7404                <1>       jz     short csftdf_check_df_cdrv
3209 0000AE68 FEC9                <1>       dec    cl
3210 0000AE6A 75F6                <1>       jnz    csftdf_df_copy_sf_name_loop
3211                              <1>
3212                              <1> csftdf_check_df_cdrv:
3213 0000AE6C 8A15[305E0100]      <1>       mov    dl, [DestinationFile_Drv]
3214 0000AE72 3A15[C6520100]      <1>       cmp    dl, [Current_Drv]
3215 0000AE78 7408                <1>       je     short csftdf_df_check_directory
3216                              <1>
3217 0000AE7A E8DEBDFFFF          <1>       call   change_current_drive
3218 0000AE7F 7301                <1>       jnc    short csftdf_df_check_directory
3219                              <1>
3220                              <1> csftdf_df_error_retn:
3221 0000AE81 C3                  <1>       retn
3222                              <1>
3223                              <1> csftdf_df_check_directory:
3224 0000AE82 BE[315E0100]        <1>       mov    esi, DestinationFile_Directory
3225 0000AE87 803E20              <1>         cmp     byte [esi], 20h
3226 0000AE8A 760F                <1>       jna    short csftdf_find_df
3227                              <1>
3228                              <1> csftdf_df_change_directory:
3229 0000AE8C FE05[AD060100]      <1>       inc    byte [Restore_CDIR]
3230 0000AE92 28E4                <1>       sub    ah, ah ; CD_COMMAND sign -> 0
3231 0000AE94 E859EDFFFF          <1>       call   change_current_directory
3232 0000AE99 72E6                <1>       jc     short csftdf_df_error_retn
3233                              <1>
3234                              <1> ;csftdf_df_change_prompt_dir_string:
3235                              <1> ;     call   change_prompt_dir_string
3236                              <1>
3237                              <1> csftdf_find_df:
3238                              <1>       ; 23/03/2016
3239 0000AE9B 29DB                <1>       sub    ebx, ebx
3240 0000AE9D 8A3D[305E0100]      <1>       mov    bh,  [DestinationFile_Drv]
3241 0000AEA3 81C300010900        <1>       add    ebx, Logical_DOSDisks
3242 0000AEA9 891D[085F0100]      <1>       mov    [csftdf_df_drv_dt], ebx
3243                              <1>
3244 0000AEAF BE[725E0100]        <1>       mov    esi, DestinationFile_Name
3245 0000AEB4 6631C0              <1>       xor    ax, ax
3246                              <1>              ; DestinationFile_AttributesMask -> any/zero
3247 0000AEB7 E833D1FFFF          <1>       call   find_first_file
3248 0000AEBC 7218                <1>       jc     short csftdf_df_check_error_code
3249                              <1>
3250                              <1> csftdf_df_ambgfn_check:
3251 0000AEBE 6609D2              <1>       or     dx, dx ; Ambiguous filename chars used sign (DX>0)
3252 0000AEC1 752A                <1>       jnz    short csftdf_df_error_inv_fname
3253                              <1>
3254                              <1> csftdf_df_found:
3255 0000AEC3 C605[DA5E0100]01    <1>       mov    byte [DestinationFileFound], 1
3256                              <1>       ; 17/10/2016 (cl -> bl)
3257 0000AECA 80E31F              <1>       and    bl, 1Fh ; Attributes, D-V-S-H-R
3258 0000AECD 745F                <1>       jz     short csftdf_df_save_first_cluster
3259                              <1>
3260                              <1> csftdf_df_permission_denied_retn:
3261 0000AECF B805000000          <1>       mov    eax, 05h ; Access/Permisson denied.
3262                              <1> csftdf_df_error_stc_retn:
3263 0000AED4 F9                  <1>       stc
3264 0000AED5 C3                  <1>       retn
3265                              <1>
3266                              <1> csftdf_df_check_error_code:
3267                              <1>       ;cmp   eax, 2
3268 0000AED6 3C02                <1>       cmp    al, 2
3269 0000AED8 75FA                <1>       jne    short csftdf_df_error_stc_retn
3270                              <1>
3271 0000AEDA C605[DA5E0100]00    <1>       mov    byte [DestinationFileFound], 0
3272                              <1>
3273                              <1>       ; 15/10/2016
3274 0000AEE1 BE[AC5C0100]        <1>       mov    esi, FindFile_Name ; *
3275 0000AEE6 E8C4D4FFFF          <1>       call   check_filename
3276 0000AEEB 7307                <1>       jnc    short csftdf_df_valid_fname
3277                              <1> csftdf_df_error_inv_fname: ; 'invalid file name !'
3278 0000AEED B81A000000          <1>       mov    eax, ERR_INV_FILE_NAME  ; 26
3279 0000AEF2 F9                  <1>       stc
3280 0000AEF3 C3                  <1>       retn
3281                              <1>
3282                              <1> csftdf_df_valid_fname:
3283                              <1>       ; 21/03/2016
3284                              <1>       ; (Capitalized file name)
3285                              <1>       ;mov   esi, FindFile_Name ; * ; 15/10/2016
3286 0000AEF4 BF[725E0100]        <1>       mov    edi, DestinationFile_Name
3287 0000AEF9 A5                  <1>       movsd
3288 0000AEFA A5                  <1>       movsd
```

```
3289 0000AEFB A5                <1>       movsd
3290                            <1>       ;movsb
3291                            <1>
3292                            <1> csftdf_check_disk_free_size_0:
3293 0000AEFC A1[1E5E0100]      <1>       mov    eax, [SourceFile_DirEntry+DirEntry_FileSize]
3294                            <1>
3295                            <1> csftdf_check_disk_free_size_1:
3296                            <1>       ;sub   ebx, ebx
3297                            <1>       ;mov   esi, Logical_DOSDisks
3298                            <1>       ;mov   bh,  [DestinationFile_Drv]
3299                            <1>       ;add   esi, ebx
3300                            <1>
3301 0000AF01 8B35[085F0100]    <1>       mov    esi, [csftdf_df_drv_dt] ; 23/03/2016
3302                            <1>
3303 0000AF07 0FB74E11          <1>       movzx  ecx, word [esi+LD_BPB+BytesPerSec] ; 17, LD_BPB + 0Bh
3304 0000AF0B 01C8              <1>       add    eax, ecx
3305 0000AF0D 48                <1>       dec    eax  ; file size (additional bytes) + 511 (round up)
3306                            <1> csftdf_check_disk_free_size_3: ; 16/03/2016
3307 0000AF0E 29D2              <1>       sub    edx, edx
3308 0000AF10 F7F1              <1>       div    ecx ; bytes per sector
3309                            <1>
3310                            <1> csftdf_check_disk_free_size:
3311 0000AF12 3B4674            <1>       cmp    eax, [esi+LD_FreeSectors]
3312 0000AF15 0F8294000000      <1>         jb      csftdf_check_disk_free_size_ok
3313 0000AF1B 770A              <1>       ja     short csftdf_df_insufficient_disk_space
3314                            <1>
3315 0000AF1D 807E0300          <1>       cmp    byte [esi+LD_FATType], 0 ; FS needs FDT sector also.
3316 0000AF21 0F8788000000      <1>         ja      csftdf_check_disk_free_size_ok
3317                            <1>
3318                            <1> csftdf_df_insufficient_disk_space:
3319 0000AF27 B827000000        <1>       mov    eax, 27h ; insufficient disk space
3320 0000AF2C EBA6              <1>       jmp    short csftdf_df_error_stc_retn
3321                            <1>
3322                            <1> csftdf_df_save_first_cluster:
3323                            <1>       ; ESI = FindFile_DirEntry (for the old destination file)
3324                            <1>       ; EAX = Old destination file size
3325                            <1>       ; 24/03/2016
3326                            <1>       ; EDI = Directory entry address (within Dir Buffer boundaries)
3327 0000AF2E 81EF00000800        <1>       sub    edi, Directory_Buffer  ; (<65536)
3328 0000AF34 66C1EF05          <1>       shr    di, 5 ; Convert entry offset to entry index/number
3329 0000AF38 66893D[AA5E0100]  <1>       mov    [DestinationFile_DirEntryNumber], di ; (<2048)
3330                            <1>
3331                            <1> csftdf_df_check_sf_df_fcluster:
3332 0000AF3F 668B5614          <1>       mov    dx, [esi+DirEntry_FstClusHI]
3333 0000AF43 C1E210            <1>       shl    edx, 16
3334 0000AF46 668B561A          <1>       mov    dx, [esi+DirEntry_FstClusLO]
3335 0000AF4A 8915[EC5E0100]    <1>       mov    [csftdf_df_cluster], edx
3336                            <1> csftdf_df_check_sf_df_fcluster_1:
3337 0000AF50 668B15[165E0100]  <1>       mov    dx, [SourceFile_DirEntry+DirEntry_FstClusHI]
3338 0000AF57 C1E210            <1>       shl    edx, 16
3339 0000AF5A 668B15[1C5E0100]  <1>       mov    dx, [SourceFile_DirEntry+DirEntry_FstClusLO]
3340 0000AF61 3B15[EC5E0100]    <1>       cmp    edx, [csftdf_df_cluster]
3341 0000AF67 7512              <1>       jne    short csftdf_df_check_sf_df_fcluster_ok
3342                            <1> csftdf_df_check_sf_df_drv:
3343 0000AF69 8A15[B05D0100]    <1>       mov    dl, [SourceFile_Drv]
3344 0000AF6F 3A15[305E0100]    <1>       cmp    dl, [DestinationFile_Drv]
3345 0000AF75 7504              <1>       jne    short csftdf_df_check_sf_df_fcluster_ok
3346                            <1>
3347                            <1>       ; source and destination files are same !
3348                            <1>       ; (they have same first cluster value on same logical disk)
3349                            <1>
3350 0000AF77 31C0              <1>       xor    eax, eax ; mov eax, 0 -> Bad command or file name !
3351 0000AF79 F9                <1>       stc
3352 0000AF7A C3                <1>       retn
3353                            <1>
3354                            <1> csftdf_df_check_sf_df_fcluster_ok:
3355                            <1> csftdf_df_move_findfile_struct:
3356                            <1>       ; mov  esi, FindFile_DirEntry
3357 0000AF7B BF[825E0100]      <1>       mov    edi, DestinationFile_DirEntry
3358 0000AF80 B908000000        <1>       mov    ecx, 8
3359 0000AF85 F3A5              <1>       rep    movsd
3360                            <1>
3361                            <1> csftdf_check_disk_free_size_2:
3362 0000AF87 89C2              <1>       mov    edx, eax ; Old destination file size
3363                            <1>
3364                            <1>       ;mov   eax, [SourceFile_DirEntry+DirEntry_FileSize]
3365 0000AF89 A1[DC5E0100]      <1>       mov    eax, [csftdf_filesize] ; 23/03/2016
3366                            <1>
3367                            <1>       ;;sub  ecx, ecx ; 0
3368                            <1>       ;mov   esi, Logical_DOSDisks
3369                            <1>       ;mov   ch,  [DestinationFile_Drv]
3370                            <1>       ;add   esi, ecx
3371                            <1>       ;
3372                            <1>       ;mov   [csftdf_df_drv_dt], esi
3373                            <1>
3374 0000AF8E 8B35[085F0100]    <1>       mov    esi, [csftdf_df_drv_dt] ; 23/03/2016
3375                            <1>
3376 0000AF94 668B4E11          <1>       mov    cx, [esi+LD_BPB+BytesPerSec] ; 17, LD_BPB + 0Bh
3377 0000AF98 01CA              <1>       add    edx, ecx ; + 512
3378 0000AF9A 01C8              <1>       add    eax, ecx ; + 512
3379 0000AF9C 4A                <1>       dec    edx ; old file size + 511 (round up)
3380 0000AF9D 48                <1>       dec    eax ; new file size + 511 (round up)
3381 0000AF9E F7D9              <1>       neg    ecx ; -512 ; 0FFFFFE00h
3382 0000AFA0 21CA              <1>       and    edx, ecx ; = old sector count * 512
3383 0000AFA2 21C8              <1>       and    eax, ecx ; = new sector count * 512
3384                            <1>
3385 0000AFA4 29D0              <1>       sub    eax, edx ; new file size - old file size (on disk)
3386 0000AFA6 7607              <1>       jna    short csftdf_check_disk_free_size_ok
3387                            <1>
3388 0000AFA8 F7D9              <1>       neg    ecx ; 512 (bytes per sector) ; 200h
3389                            <1>       ; check free space for additional sectors
3390                            <1>       ; eax = number of additional sectors * bytes per sector
```

```
                                <1>       ; esi = Logical DOS drive number (of destination disk)
0000AFAA E95FFFFFFF              <1>         jmp     csftdf_check_disk_free_size_3
                                <1>
                                <1> csftdf_check_disk_free_size_ok:
                                <1>       ; 18/03/2016
                                <1> csftdf_df_check_copy_cmd_phase:
0000AFAF A0[D85E0100]            <1>       mov   al, [copy_cmd_phase]
0000AFB4 3C01                    <1>       cmp   al, 1
0000AFB6 7514                    <1>       jne   short csftdf2_check_cdrv
                                <1>
0000AFB8 31C0                    <1>       xor   eax, eax
0000AFBA A2[D85E0100]            <1>       mov   [copy_cmd_phase], al ; 0
                                <1>
0000AFBF 8A15[DA5E0100]          <1>       mov   dl, [DestinationFileFound]
0000AFC5 8A35[0D5E0100]          <1>       mov   dh, [SourceFile_DirEntry+11] ; Attributes
                                <1>
                                <1> csftdf_return:
0000AFCB C3                      <1>       retn
                                <1>
                                <1> ; Phase 2
                                <1>
                                <1> csftdf2_check_cdrv:
                                <1>       ; 18/03/2016
                                <1>       ; Here, destination drive and directory are ready !
                                <1>       ; (checking/restoring is not needed)
                                <1>       ; (Since at the end of the phase 1)
                                <1>
                                <1> ;     mov   dl, [DestinationFile_Drv]
                                <1> ;     cmp   dl, [Current_Drv]
                                <1> ;     je    short csftdf2_df_check_directory
                                <1> ;
                                <1> ;     call  change_current_drive
                                <1> ;     jc    short csftdf2_read_error
                                <1> ;
                                <1> ;csftdf2_df_check_directory:
                                <1> ;     mov   esi, DestinationFile_Directory
                                <1> ;     cmp   byte [esi], 20h
                                <1> ;     jna   short csftdf2_df_check_found_or_not
                                <1> ;
                                <1> ;csftdf2_df_change_directory:
                                <1> ;     inc   byte [Restore_CDIR]
                                <1> ;     xor   ah, ah ; CD_COMMAND sign -> 0
                                <1> ;     call  change_current_directory
                                <1> ;     jc    short csftdf2_stc_return
                                <1> ;
                                <1> ;;csftdf2_df_change_prompt_dir_string:
                                <1> ;;    call  change_prompt_dir_string
                                <1>
                                <1> csftdf2_df_check_found_or_not:
                                <1>       ; 21/03/2016
0000AFCC 803D[DA5E0100]00        <1>       cmp   byte [DestinationFileFound], 0
0000AFD3 7739                    <1>       ja    short csftdf2_set_sf_percentage
                                <1>
                                <1> csftdf2_create_file:
0000AFD5 BE[725E0100]            <1>       mov   esi, DestinationFile_Name
0000AFDA A1[DC5E0100]            <1>       mov   eax, [csftdf_filesize]
0000AFDF 30C9                    <1>       xor   cl, cl ; 0
                                <1>
0000AFE1 31DB                    <1>       xor   ebx, ebx ; 0
0000AFE3 4B                      <1>       dec   ebx ; 0FFFFFFFFh
                                <1>
                                <1>       ; INPUT ->
                                <1>       ;     EAX -> File Size
                                <1>       ;     ESI = ASCIIZ File name
                                <1>       ;      CL = File attributes
                                <1>       ;     EBX = FFFFFFFFh -> empty file sign for FAT fs
                                <1>       ;     EBX <> FFFFFFFFh -> use file size for FAT fs
                                <1>       ;
                                <1>       ; OUTPUT ->
                                <1>       ;     EAX = New file's first cluster
                                <1>       ;     ESI = Logical Dos Drv Descr. Table Addr.
                                <1>       ;     EBX = CreateFile_Size address
                                <1>       ;     ECX = Sectors per cluster (<256)
                                <1>       ;     EDX = Directory Entry Index/Number (<65536)
                                <1>       ;
                                <1>       ;     cf = 1 -> error code in AL (EAX)
                                <1>
0000AFE4 E8EC050000              <1>       call  create_file
                                <1>       ;pop  esi
0000AFE9 0F82A3050000            <1>         jc      csftdf2_rw_error
                                <1>
                                <1> csftdf2_create_file_OK:
0000AFEF A3[EC5E0100]            <1>       mov   [csftdf_df_cluster], eax
                                <1>
                                <1>       ; 24/03/2016
0000AFF4 668915[AA5E0100]        <1>       mov   [DestinationFile_DirEntryNumber], dx
                                <1>
                                <1>       ; 21/03/2016
0000AFFB BE00000800              <1>       mov   esi, Directory_Buffer
0000B000 C1E205                  <1>       shl   edx, 5 ; 32 * index number
0000B003 01D6                    <1>       add   esi, edx
0000B005 BF[825E0100]            <1>       mov   edi, DestinationFile_DirEntry
0000B00A B108                    <1>       mov   cl, 8 ; 32 bytes
0000B00C F3A5                    <1>       rep   movsd
                                <1>
                                <1> csftdf2_set_sf_percentage:
                                <1>       ; 17/03/2016
0000B00E 31C0                    <1>       xor   eax, eax
0000B010 A2[005F0100]            <1>       mov   [csftdf_percentage], al ; 0, reset
                                <1>
0000B015 A3[F85E0100]            <1>       mov   [csftdf_sf_rbytes], eax ; 0, reset
0000B01A A3[FC5E0100]            <1>       mov   [csftdf_df_wbytes], eax ; 0, reset
```

```
3493                                    <1>
3494 0000B01F 8A25[B05D0100]            <1>       mov    ah, [SourceFile_Drv]
3495 0000B025 BE00010900                <1>       mov    esi, Logical_DOSDisks
3496 0000B02A 01C6                      <1>       add    esi, eax
3497                                    <1>
3498 0000B02C 8935[045F0100]            <1>       mov    [csftdf_sf_drv_dt], esi ; 23/03/2016
3499                                    <1>
3500 0000B032 668B15[165E0100]          <1>       mov    dx, [SourceFile_DirEntry+DirEntry_FstClusHI]
3501 0000B039 C1E210                    <1>       shl    edx, 16
3502 0000B03C 668B15[1C5E0100]          <1>       mov    dx, [SourceFile_DirEntry+DirEntry_FstClusLO]
3503 0000B043 8915[E85E0100]            <1>       mov    [csftdf_sf_cluster], edx
3504                                    <1>
3505                                    <1>       ; 16/03/2016
3506                                    <1>       ; Note: Singlix FS boot sector parameters (for cluster
3507                                    <1>       ;      related calculations) has same offset
3508                                    <1>       ;      values from LD_BPB as in FAT file system.
3509                                    <1>       ;      [esi+LD_BPB+SecPerClust] is 1 for Singlix FS.
3510                                    <1>       ;
3511 0000B049 0FB64E13                  <1>       movzx  ecx, byte [esi+LD_BPB+SecPerClust]
3512 0000B04D 880D[2E5E0100]            <1>       mov    [SourceFile_SecPerClust], cl
3513                                    <1>
3514                                    <1>       ; 17/03/2016
3515 0000B053 386E03                    <1>       cmp    [esi+LD_FATType], ch ; 0
3516 0000B056 7707                      <1>       ja     short csftdf2_set_sf_percent_rsize1
3517                                    <1>
3518 0000B058 B800000100                <1>       mov    eax, 65536 ; read/write buffer size for Singlix FS
3519 0000B05D EB06                      <1>       jmp    short csftdf2_set_sf_percent_rsize2
3520                                    <1>
3521                                    <1> csftdf2_set_sf_percent_rsize1:
3522 0000B05F 668B4611                  <1>       mov    ax, [esi+LD_BPB+BytesPerSec]
3523 0000B063 F7E1                      <1>       mul    ecx
3524                                    <1>       ;sub   edx, edx
3525                                    <1> csftdf2_set_sf_percent_rsize2:
3526 0000B065 A3[F05E0100]              <1>       mov    [csftdf_r_size], eax
3527                                    <1>
3528                                    <1> csftdf2_set_df_percentage:
3529                                    <1>       ;sub   eax, eax
3530                                    <1>       ;mov   ah, [DestinationFile_Drv]
3531                                    <1>       ;mov   edi, Logical_DOSDisks
3532                                    <1>       ;add   edi, eax
3533                                    <1>       ;mov   [csftdf_df_drv_dt], edi ; 17/03/2016
3534                                    <1>
3535 0000B06A 8B3D[085F0100]            <1>       mov    edi, [csftdf_df_drv_dt] ; 23/03/2016
3536                                    <1>
3537                                    <1>       ; 16/03/2016
3538                                    <1>       ; Note: Singlix FS boot sector parameters (for cluster
3539                                    <1>       ;      related calculations) has same offset
3540                                    <1>       ;      values from LD_BPB as in FAT file system.
3541                                    <1>       ;      [edi+LD_BPB+SecPerClust] is 1 for Singlix FS.
3542                                    <1>       ;
3543                                    <1>       ;movzx ecx, byte [edi+LD_BPB+SecPerClust]
3544 0000B070 8A4F13                  <1>       mov    cl, [edi+LD_BPB+SecPerClust]
3545 0000B073 880D[AE5E0100]            <1>       mov    [DestinationFile_SecPerClust], cl
3546                                    <1>
3547                                    <1>       ; 17/03/2016
3548 0000B079 386F03                    <1>       cmp    [edi+LD_FATType], ch ; 0
3549 0000B07C 7707                      <1>       ja     short csftdf2_set_df_percent_wsize1
3550                                    <1>
3551 0000B07E B800000100                <1>       mov    eax, 65536 ; read/write buffer size for Singlix FS
3552 0000B083 EB06                      <1>       jmp    short csftdf2_set_df_percent_wsize2
3553                                    <1>
3554                                    <1> csftdf2_set_df_percent_wsize1:
3555 0000B085 0FB74711                  <1>       movzx  eax, word [edi+LD_BPB+BytesPerSec]
3556 0000B089 F7E1                      <1>       mul    ecx
3557                                    <1>       ;sub   edx, edx
3558                                    <1> csftdf2_set_df_percent_wsize2:
3559 0000B08B A3[F45E0100]              <1>       mov    [csftdf_w_size], eax
3560                                    <1>
3561 0000B090 A1[DC5E0100]              <1>       mov    eax, [csftdf_filesize]
3562                                    <1>
3563 0000B095 3D00000100                <1>       cmp    eax, 65536 ; 64KB   ; small file
3564 0000B09A 721F                      <1>       jb     short csftdf2_load_file ; do not display percentage
3565                                    <1>
3566                                    <1> csftdf2_reset_wf_percent_ptr_chk_64k:
3567 0000B09C B201                      <1>       mov    dl, 1 ; 25/03/2016
3568                                    <1>
3569 0000B09E 3D00000400                <1>       cmp    eax, 65536*4 ; 256KB
3570 0000B0A3 7310                      <1>       jnb    short csftdf2_enable_percentage_display ; big file
3571                                    <1>
3572                                    <1>       ; 64-128KB file size for floppy disks
3573 0000B0A5 3815[B05D0100]            <1>       cmp    byte [SourceFile_Drv], dl ; 1 ; read from floppy disk ?
3574 0000B0AB 7608                      <1>       jna    short csftdf2_enable_percentage_display
3575                                    <1>
3576 0000B0AD 3815[305E0100]            <1>       cmp    byte [DestinationFile_Drv], dl ; 1 ; write to floppy disk ?
3577 0000B0B3 7706                      <1>       ja     short csftdf2_load_file
3578                                    <1>
3579                                    <1> csftdf2_enable_percentage_display:
3580 0000B0B5 8815[005F0100]            <1>       mov    [csftdf_percentage], dl ; 1
3581                                    <1>
3582                                    <1> csftdf2_load_file:
3583                                    <1>       ; 13/05/2016
3584                                    <1>       ; 19/03/2016
3585                                    <1>       ; 18/03/2016
3586                                    <1>       ; 17/03/2016
3587 0000B0BB B40F                      <1>       mov    ah, 0Fh
3588 0000B0BD E8D863FFFF                <1>       call   _int10h
3589                                    <1>       ; 13/05/2016
3590 0000B0C2 883D[015F0100]            <1>       mov    [csftdf_videopage], bh ; active video page
3591 0000B0C8 B403                      <1>       mov    ah, 03h
3592 0000B0CA E8CB63FFFF                <1>       call   _int10h
3593 0000B0CF 668915[025F0100]          <1>       mov    [csftdf_cursorpos], dx
3594                                    <1>
```

```
3595 0000B0D6 29C0                <1>       sub    eax, eax
3596 0000B0D8 A2[D95E0100]        <1>       mov    [csftdf_rw_err], al ; 0
3597                              <1>
3598                              <1> ; ///
3599                              <1> csftdf_sf_amb: ; 15/03/2016
3600 0000B0DD 8B0D[DC5E0100]      <1>       mov    ecx, [csftdf_filesize]    ; 23/03/2016
3601                              <1>
3602                              <1>       ; TRDOS 386 (TRDOS v2.0)
3603                              <1>       ; Allocate contiguous memory block for loading the file
3604                              <1>
3605                              <1>       ;mov   ecx, [SourceFile_DirEntry+DirEntry_FileSize]
3606                              <1>
3607                              <1>       ;sub   eax, eax ; First free memory aperture
3608                              <1>
3609                              <1>       ; eax = 0 (Allocate memory from the beginning)
3610                              <1>       ; ecx = File (Allocation) size in bytes
3611                              <1>
3612 0000B0E3 E83BA3FFFF          <1>       call   allocate_memory_block
3613 0000B0E8 7304                <1>       jnc    short loc_check_sf_save_loading_parms
3614                              <1>
3615 0000B0EA 29C0                <1>       sub    eax, eax
3616 0000B0EC 29C9                <1>       sub    ecx, ecx
3617                              <1>
3618                              <1> loc_check_sf_save_loading_parms:
3619 0000B0EE A3[E05E0100]        <1>       mov    [csftdf_sf_mem_addr], eax ; loading address
3620 0000B0F3 890D[E45E0100]      <1>       mov    [csftdf_sf_mem_bsize], ecx ; block size
3621                              <1> ; ///
3622                              <1>       ; 19/03/2016
3623 0000B0F9 8B35[045F0100]      <1>       mov    esi, [csftdf_sf_drv_dt] ; logical dos drv desc. tbl.
3624                              <1>
3625                              <1>       ; 17/03/2016
3626 0000B0FF 09C0                <1>       or     eax, eax ; contiguous free memory block address
3627 0000B101 0F845B010000        <1>         jz     csftdf2_read_sf_cluster
3628                              <1>
3629                              <1>       ; 18/03/2016
3630 0000B107 8B1D[E05E0100]      <1>       mov    ebx, [csftdf_sf_mem_addr] ; memory block address
3631                              <1>
3632 0000B10D 807E0300            <1>       cmp    byte [esi+LD_FATType], 0
3633 0000B111 0F8605020000        <1>         jna    csftdf2_load_fs_file
3634                              <1>
3635                              <1> csftdf2_load_fat_file:
3636 0000B117 53                  <1>       push   ebx ; *
3637                              <1>
3638                              <1> csftdf2_load_fat_file_next:
3639 0000B118 BE[FD0C0100]        <1>       mov    esi, msg_reading
3640 0000B11D E83BB2FFFF          <1>       call   print_msg
3641                              <1>
3642 0000B122 803D[005F0100]00    <1>       cmp    byte [csftdf_percentage], 0
3643 0000B129 7605                <1>       jna    short csftdf2_load_fat_file_1
3644                              <1>
3645 0000B12B E87C000000          <1>       call   csftdf2_print_percentage ; 19/03/2016
3646                              <1>
3647                              <1> csftdf2_load_fat_file_1:
3648 0000B130 8B35[045F0100]      <1>       mov    esi, [csftdf_sf_drv_dt]
3649 0000B136 5B                  <1>       pop    ebx ; *
3650                              <1>
3651                              <1> csftdf2_load_fat_file_2:
3652 0000B137 E8B8000000          <1>       call   csftdf2_read_fat_file_sectors ; 19/03/2016
3653 0000B13C 0F8250040000        <1>         jc     csftdf2_rw_error ; eocc! or disk error!
3654                              <1>
3655 0000B142 09D2                <1>       or     edx, edx ; edx > 0 -> EOF
3656 0000B144 7520                <1>       jnz    short csftdf2_load_fat_file_ok
3657                              <1>
3658 0000B146 803D[005F0100]00    <1>       cmp    byte [csftdf_percentage], 0
3659 0000B14D 76E8                <1>       jna    short csftdf2_load_fat_file_2
3660                              <1>
3661 0000B14F 53                  <1>       push   ebx ; *
3662                              <1>
3663                              <1>       ; Set cursor position
3664                              <1>       ; AH= 02h, BH= Page Number, DH= Row, DL= Column
3665 0000B150 8A3D[015F0100]      <1>       mov    bh, [csftdf_videopage]
3666 0000B156 668B15[025F0100]    <1>       mov    dx, [csftdf_cursorpos]
3667 0000B15D B402                <1>       mov    ah, 2
3668 0000B15F E83663FFFF          <1>       call   _int10h
3669 0000B164 EBB2                <1>       jmp    short csftdf2_load_fat_file_next
3670                              <1>
3671                              <1> csftdf2_load_fat_file_ok:
3672 0000B166 803D[005F0100]00    <1>       cmp    byte [csftdf_percentage], 0
3673 0000B16D 0F8651020000        <1>         jna    csftdf2_save_file ; 25/03/2016
3674                              <1>
3675                              <1>       ; "Reading... 100%"
3676 0000B173 BF[150D0100]        <1>       mov    edi, percentagestr
3677 0000B178 B031                <1>       mov    al, '1'
3678 0000B17A AA                  <1>       stosb
3679 0000B17B B030                <1>       mov    al, '0'
3680 0000B17D AA                  <1>       stosb
3681 0000B17E AA                  <1>       stosb
3682                              <1>
3683 0000B17F 8A3D[015F0100]      <1>       mov    bh, [csftdf_videopage]
3684 0000B185 668B15[025F0100]    <1>       mov    dx, [csftdf_cursorpos]
3685 0000B18C B402                <1>       mov    ah, 2
3686 0000B18E E80763FFFF          <1>       call   _int10h
3687                              <1>
3688 0000B193 BE[FD0C0100]        <1>       mov    esi, msg_reading
3689 0000B198 E8C0B1FFFF          <1>       call   print_msg
3690                              <1>
3691 0000B19D BE[150D0100]        <1>       mov    esi, percentagestr
3692 0000B1A2 E8B6B1FFFF          <1>       call   print_msg
3693                              <1>
3694 0000B1A7 E918020000          <1>         jmp    csftdf2_save_file ; 25/03/2016
3695                              <1>
3696                              <1> csftdf2_print_percentage:
```

```
3697                                    <1>       ; 19/03/2016
3698                                    <1>       ; 18/03/2016
3699 0000B1AC B020                      <1>       mov    al, 20h
3700 0000B1AE BF[150D0100]              <1>       mov    edi, percentagestr
3701 0000B1B3 AA                        <1>       stosb
3702 0000B1B4 AA                        <1>       stosb
3703 0000B1B5 A1[F85E0100]              <1>       mov    eax, [csftdf_sf_rbytes]
3704 0000B1BA BA64000000                <1>       mov    edx, 100
3705 0000B1BF F7E2                      <1>       mul    edx
3706 0000B1C1 8B0D[DC5E0100]            <1>       mov    ecx, [csftdf_filesize]
3707 0000B1C7 F7F1                      <1>       div    ecx
3708 0000B1C9 B10A                      <1>       mov    cl, 10
3709 0000B1CB F6F1                      <1>       div    cl
3710 0000B1CD 80C430                    <1>       add    ah, '0'
3711 0000B1D0 8827                      <1>       mov    [edi], ah
3712 0000B1D2 20C0                      <1>       and    al, al
3713 0000B1D4 740A                      <1>       jz     short csftdf2_print_percent_1
3714 0000B1D6 4F                        <1>       dec    edi
3715 0000B1D7 6698                      <1>       cbw
3716 0000B1D9 F6F1                      <1>       div    cl
3717 0000B1DB 80C430                    <1>       add    ah, '0'
3718 0000B1DE 8827                      <1>       mov    [edi], ah
3719                                    <1>       ;and   al, al
3720                                    <1>       ;jz    short csftdf2_print_percent_1
3721                                    <1>       ;dec   edi
3722                                    <1>       ;mov   [edi], '1' ; 100%
3723                                    <1>
3724                                    <1> csftdf2_print_percent_1:
3725 0000B1E0 BE[150D0100]              <1>       mov    esi, percentagestr
3726                                    <1>       ;call  print_msg
3727                                    <1>       ;retn
3728 0000B1E5 E973B1FFFF                <1>       jmp    print_msg
3729                                    <1>
3730                                    <1> csftdf2_read_file_sectors:
3731                                    <1>       ; 19/03/2016
3732 0000B1EA 807E0300                  <1>       cmp    byte [esi+LD_FATType], 0
3733 0000B1EE 0F8627070000              <1>         jna     csftdf2_read_fs_file_sectors
3734                                    <1>
3735                                    <1> csftdf2_read_fat_file_sectors:
3736                                    <1>       ; 19/03/2016
3737                                    <1>       ; 18/03/2016
3738                                    <1>       ; return:
3739                                    <1>       ;   CF = 0 & EDX > 0 -> END OF FILE
3740                                    <1>       ;   CF = 0 & EDX = 0 -> not EOF
3741                                    <1>       ;   CF = 1 -> read error (error code in AL)
3742                                    <1>
3743                                    <1> csftdf2_read_fat_file_secs_0:
3744 0000B1F4 8B15[DC5E0100]            <1>       mov    edx, [csftdf_filesize]
3745 0000B1FA 2B15[F85E0100]            <1>       sub    edx, [csftdf_sf_rbytes]
3746 0000B200 3B15[F05E0100]            <1>       cmp    edx, [csftdf_r_size]
3747 0000B206 7306                      <1>       jnb    short csftdf2_read_fat_file_secs_1
3748 0000B208 8915[F05E0100]            <1>       mov    [csftdf_r_size], edx
3749                                    <1>
3750                                    <1> csftdf2_read_fat_file_secs_1:
3751 0000B20E A1[F05E0100]              <1>       mov    eax, [csftdf_r_size]
3752 0000B213 29D2                      <1>       sub    edx, edx
3753 0000B215 0FB74E11                  <1>       movzx  ecx, word [esi+LD_BPB+BytesPerSec]
3754 0000B219 01C8                      <1>       add    eax, ecx
3755 0000B21B 48                        <1>       dec    eax
3756 0000B21C F7F1                      <1>       div    ecx
3757 0000B21E 89C1                      <1>       mov    ecx, eax ; sector count
3758 0000B220 A1[E85E0100]              <1>       mov    eax, [csftdf_sf_cluster]
3759                                    <1>
3760                                    <1>       ; EBX = memory block address (current)
3761                                    <1>
3762 0000B225 E821090000                <1>       call   read_fat_file_sectors
3763 0000B22A 7235                      <1>       jc     short csftdf2_read_fat_file_secs_3
3764                                    <1>
3765                                    <1>       ; EBX = next memory address
3766                                    <1>
3767 0000B22C A1[F85E0100]              <1>       mov    eax, [csftdf_sf_rbytes]
3768 0000B231 0305[F05E0100]            <1>       add    eax, [csftdf_r_size]
3769 0000B237 8B15[DC5E0100]            <1>       mov    edx, [csftdf_filesize]
3770 0000B23D 39D0                      <1>       cmp    eax, edx
3771 0000B23F 7320                      <1>       jnb    short csftdf2_read_fat_file_secs_3 ; edx > 0
3772 0000B241 A3[F85E0100]              <1>       mov    [csftdf_sf_rbytes], eax
3773                                    <1>
3774 0000B246 53                        <1>       push   ebx ; *
3775                                    <1>       ; get next cluster (csftdf_r_size! bytes)
3776 0000B247 A1[E85E0100]              <1>       mov    eax, [csftdf_sf_cluster]
3777 0000B24C E8CC060000                <1>       call   get_next_cluster
3778 0000B251 5B                        <1>       pop    ebx ; *
3779 0000B252 7306                      <1>       jnc    short csftdf2_read_fat_file_secs_2
3780                                    <1>
3781                                    <1>       ; 15/10/2016
3782                                    <1>       ;Disk read error instad of drv not ready err
3783 0000B254 B811000000                <1>       mov    eax, 17 ; Read error !
3784 0000B259 C3                        <1>       retn
3785                                    <1>
3786                                    <1> csftdf2_read_fat_file_secs_2:
3787 0000B25A 29D2                      <1>       sub    edx, edx ; 0
3788 0000B25C A3[E85E0100]              <1>       mov    [csftdf_sf_cluster], eax ; next cluster
3789                                    <1>
3790                                    <1> csftdf2_read_fat_file_secs_3:
3791 0000B261 C3                        <1>       retn
3792                                    <1>
3793                                    <1> csftdf2_read_sf_cluster:
3794                                    <1>       ; 19/03/2016
3795 0000B262 BB00000700                <1>       mov    ebx, Cluster_Buffer ; buffer address (64KB)
3796                                    <1>
3797 0000B267 803D[005F0100]00          <1>       cmp    byte [csftdf_percentage], 0
3798 0000B26E 760D                      <1>       jna    short csftdf2_read_sf_clust_2
```

```
3799                                    <1>
3800 0000B270 53                        <1>       push   ebx ; *
3801                                    <1>
3802                                    <1> csftdf2_read_sf_clust_next:
3803 0000B271 E836FFFFFF                <1>       call   csftdf2_print_percentage
3804                                    <1>
3805                                    <1> csftdf2_read_sf_clust_0:
3806 0000B276 8B35[045F0100]            <1>       mov    esi, [csftdf_sf_drv_dt]
3807                                    <1> csftdf2_read_sf_clust_1:
3808 0000B27C 5B                        <1>       pop    ebx ; *
3809                                    <1>
3810                                    <1> csftdf2_read_sf_clust_2:
3811 0000B27D 89DA                      <1>       mov    edx, ebx
3812 0000B27F 0315[F05E0100]            <1>       add    edx, [csftdf_r_size]
3813 0000B285 81FA00000800              <1>       cmp    edx, Cluster_Buffer + 65536
3814 0000B28B 772F                      <1>       ja     short csftdf2_write_df_cluster
3815                                    <1>
3816 0000B28D E858FFFFFF                <1>       call   csftdf2_read_file_sectors ; 19/03/2016
3817 0000B292 0F8280020000              <1>         jc     csftdf2_save_fat_file_err2 ; eocc! or disk error!
3818                                    <1>
3819 0000B298 09D2                      <1>       or     edx, edx ; edx > 0 -> EOF
3820 0000B29A 7520                      <1>       jnz    short csftdf2_write_df_cluster
3821                                    <1>
3822 0000B29C 803D[005F0100]00          <1>       cmp    byte [csftdf_percentage], 0
3823 0000B2A3 76D8                      <1>       jna    short csftdf2_read_sf_clust_2
3824                                    <1>
3825 0000B2A5 53                        <1>       push   ebx ; *
3826                                    <1>
3827                                    <1>       ; Set cursor position
3828                                    <1>       ; AH= 02h, BH= Page Number, DH= Row, DL= Column
3829 0000B2A6 8A3D[015F0100]            <1>       mov    bh, [csftdf_videopage]
3830 0000B2AC 668B15[025F0100]          <1>       mov    dx, [csftdf_cursorpos]
3831 0000B2B3 B402                      <1>       mov    ah, 2
3832 0000B2B5 E8E061FFFF                <1>       call   _int10h
3833 0000B2BA EBB5                      <1>       jmp    short csftdf2_read_sf_clust_next
3834                                    <1>
3835                                    <1> csftdf2_write_df_cluster:
3836                                    <1>       ; 19/03/2016
3837 0000B2BC 8B35[085F0100]            <1>       mov    esi, [csftdf_df_drv_dt]
3838 0000B2C2 BB00000700                <1>       mov    ebx, Cluster_Buffer ; buffer address (64KB)
3839                                    <1>
3840                                    <1> csftdf2_write_df_clust_next:
3841 0000B2C7 E855000000                <1>       call   csftdf2_write_file_sectors ; 19/03/2016
3842 0000B2CC 0F8246020000              <1>         jc     csftdf2_save_fat_file_err2 ; eocc! or disk error!
3843                                    <1>
3844 0000B2D2 09D2                      <1>       or     edx, edx ; edx > 0 -> EOF
3845 0000B2D4 750A                      <1>       jnz    short csftdf2_rw_f_clust_ok
3846                                    <1>
3847 0000B2D6 81FB00000800              <1>       cmp    ebx, Cluster_Buffer + 65536
3848 0000B2DC 72E9                      <1>       jb     short csftdf2_write_df_clust_next
3849                                    <1>
3850 0000B2DE EB82                      <1>       jmp    short csftdf2_read_sf_cluster
3851                                    <1>
3852                                    <1> csftdf2_rw_f_clust_ok:
3853 0000B2E0 803D[005F0100]00          <1>       cmp    byte [csftdf_percentage], 0
3854 0000B2E7 0F86B2010000              <1>         jna    csftdf2_save_fat_file_4 ; 25/03/2016
3855                                    <1>
3856                                    <1>       ; "100%"
3857 0000B2ED BF[150D0100]              <1>       mov    edi, percentagestr
3858 0000B2F2 B031                      <1>       mov    al, '1'
3859 0000B2F4 AA                        <1>       stosb
3860 0000B2F5 B030                      <1>       mov    al, '0'
3861 0000B2F7 AA                        <1>       stosb
3862 0000B2F8 AA                        <1>       stosb
3863                                    <1>
3864 0000B2F9 8A3D[015F0100]            <1>       mov    bh, [csftdf_videopage]
3865 0000B2FF 668B15[025F0100]          <1>       mov    dx, [csftdf_cursorpos]
3866 0000B306 B402                      <1>       mov    ah, 2
3867 0000B308 E88D61FFFF                <1>       call   _int10h
3868                                    <1>
3869 0000B30D BE[150D0100]              <1>       mov    esi, percentagestr
3870 0000B312 E846B0FFFF                <1>       call   print_msg
3871                                    <1>
3872 0000B317 E983010000                <1>         jmp    csftdf2_save_fat_file_4
3873                                    <1>
3874                                    <1> csftdf2_load_fs_file:
3875                                    <1>       ; temporary - 18/03/2016
3876 0000B31C E96F020000                <1>         jmp    csftdf2_read_error
3877                                    <1>
3878                                    <1> csftdf2_write_file_sectors:
3879                                    <1>       ; 19/03/2016
3880 0000B321 807E0300                  <1>       cmp    byte [esi+LD_FATType], 0
3881 0000B325 0F86F1050000              <1>         jna    csftdf2_write_fs_file_sectors
3882                                    <1>
3883                                    <1> csftdf2_write_fat_file_sectors:
3884                                    <1>       ; 19/03/2016
3885                                    <1>       ; 18/03/2016
3886                                    <1>       ; return:
3887                                    <1>       ;   CF = 0 & EDX > 0 -> END OF FILE
3888                                    <1>       ;   CF = 0 & EDX = 0 -> not EOF
3889                                    <1>       ;   CF = 1 -> write error (error code in AL)
3890                                    <1>
3891                                    <1> csftdf2_write_fat_file_secs_0:
3892 0000B32B 8B15[DC5E0100]            <1>       mov    edx, [csftdf_filesize]
3893 0000B331 2B15[FC5E0100]            <1>       sub    edx, [csftdf_df_wbytes]
3894 0000B337 3B15[F45E0100]            <1>       cmp    edx, [csftdf_w_size]
3895 0000B33D 7306                      <1>       jnb    short csftdf2_write_fat_file_secs_1
3896 0000B33F 8915[F45E0100]            <1>       mov    [csftdf_w_size], edx
3897                                    <1>
3898                                    <1> csftdf2_write_fat_file_secs_1:
3899 0000B345 A1[F45E0100]              <1>       mov    eax, [csftdf_w_size]
3900 0000B34A 29D2                      <1>       sub    edx, edx
```

```
3901 0000B34C 0FB74E11            <1>       movzx  ecx, word [esi+LD_BPB+BytesPerSec]
3902 0000B350 01C8                <1>       add    eax, ecx
3903 0000B352 48                  <1>       dec    eax
3904 0000B353 F7F1                <1>       div    ecx
3905 0000B355 89C1                <1>       mov    ecx, eax ; sector count
3906 0000B357 A1[EC5E0100]        <1>       mov    eax, [csftdf_df_cluster]
3907                              <1>
3908                              <1>       ; EBX = memory block address (current)
3909                              <1>
3910 0000B35C E8A20F0000          <1>       call   write_fat_file_sectors
3911 0000B361 7259                <1>       jc     short csftdf2_write_fat_file_secs_4
3912                              <1>
3913                              <1>       ; EBX = next memory address
3914                              <1>
3915 0000B363 A1[FC5E0100]        <1>       mov    eax, [csftdf_df_wbytes]
3916 0000B368 0305[F45E0100]      <1>       add    eax, [csftdf_w_size]
3917 0000B36E 8B15[DC5E0100]      <1>       mov    edx, [csftdf_filesize]
3918 0000B374 39D0                <1>       cmp    eax, edx
3919 0000B376 7344                <1>       jnb    short csftdf2_write_fat_file_secs_4
3920 0000B378 A3[FC5E0100]        <1>       mov    [csftdf_df_wbytes], eax
3921                              <1>       ;
3922 0000B37D A3[9E5E0100]        <1>       mov    [DestinationFile_DirEntry+DirEntry_FileSize], eax
3923                              <1>
3924 0000B382 53                  <1>       push   ebx ; *
3925                              <1>
3926 0000B383 803D[DA5E0100]01    <1>       cmp    byte [DestinationFileFound], 1
3927 0000B38A 7210                <1>       jb     short csftdf2_write_fat_file_secs_2
3928                              <1>
3929                              <1>       ; get next cluster (csftdf_w_size! bytes)
3930 0000B38C A1[EC5E0100]        <1>       mov    eax, [csftdf_df_cluster]
3931 0000B391 E887050000          <1>       call   get_next_cluster
3932 0000B396 731C                <1>       jnc    short csftdf2_write_fat_file_secs_3
3933                              <1>
3934 0000B398 21C0                <1>       and    eax, eax ; end of cluster chain!?
3935 0000B39A 7521                <1>       jnz    short csftdf2_write_fat_file_secs_5 ; disk error !
3936                              <1>
3937                              <1> csftdf2_write_fat_file_secs_2:
3938 0000B39C A1[EC5E0100]        <1>       mov    eax, [csftdf_df_cluster] ; last cluster
3939 0000B3A1 E8800E0000          <1>       call   add_new_cluster
3940 0000B3A6 7215                <1>       jc     short csftdf2_write_fat_file_secs_5
3941                              <1>
3942                              <1>       ; NOTE: Destination file size may be bigger than
3943                              <1>       ; source file size when the last reading fails after here.
3944                              <1>       ; (The last -empty- cluster of destination file must be
3945                              <1>       ; truncated and LMDT must be current date&time for partial
3946                              <1>       ; copy result!)
3947 0000B3A8 8B15[F45E0100]      <1>       mov    edx, [csftdf_w_size] ; bytes per cluster
3948 0000B3AE 0115[9E5E0100]      <1>       add    [DestinationFile_DirEntry+DirEntry_FileSize], edx
3949                              <1>
3950                              <1> csftdf2_write_fat_file_secs_3:
3951 0000B3B4 5B                  <1>       pop    ebx ; *
3952 0000B3B5 29D2                <1>       sub    edx, edx ; 0
3953 0000B3B7 A3[EC5E0100]        <1>       mov    [csftdf_df_cluster], eax ; next cluster
3954                              <1>
3955                              <1> csftdf2_write_fat_file_secs_4:
3956 0000B3BC C3                  <1>       retn
3957                              <1>
3958                              <1> csftdf2_write_fat_file_secs_5:
3959 0000B3BD 5B                  <1>       pop    ebx ; *
3960                              <1>       ; 16/10/2016 (1Dh -> 18)
3961 0000B3BE B812000000          <1>       mov    eax, 18 ; Write error !
3962 0000B3C3 C3                  <1>       retn
3963                              <1>
3964                              <1> csftdf2_save_file:
3965                              <1>       ; 25/03/2016
3966                              <1>       ; 19/03/2016
3967                              <1>       ; 18/03/2016
3968 0000B3C4 8B35[085F0100]      <1>       mov    esi, [csftdf_df_drv_dt] ; logical dos drv desc. tbl.
3969                              <1>
3970 0000B3CA 8B1D[E05E0100]      <1>       mov    ebx, [csftdf_sf_mem_addr] ; memory block address
3971                              <1>
3972 0000B3D0 807E0300            <1>       cmp    byte [esi+LD_FATType], 0
3973 0000B3D4 0F86F4010000        <1>         jna     csftdf2_save_fs_file
3974                              <1>
3975                              <1> csftdf2_save_fat_file:
3976 0000B3DA 53                  <1>       push   ebx; *
3977                              <1>
3978 0000B3DB 803D[005F0100]00    <1>       cmp    byte [csftdf_percentage], 0
3979 0000B3E2 7724                <1>       ja     short csftdf2_save_fat_file_0
3980                              <1>
3981                              <1>       ; Set cursor position
3982                              <1>       ; AH= 02h, BH= Page Number, DH= Row, DL= Column
3983 0000B3E4 8A3D[015F0100]      <1>       mov    bh, [csftdf_videopage]
3984 0000B3EA 668B15[025F0100]    <1>       mov    dx, [csftdf_cursorpos]
3985 0000B3F1 B402                <1>       mov    ah, 2
3986 0000B3F3 E8A260FFFF          <1>       call   _int10h
3987                              <1>
3988 0000B3F8 BE[090D0100]        <1>       mov    esi, msg_writing
3989 0000B3FD E85BAFFFFF          <1>       call   print_msg
3990                              <1>
3991                              <1> csftdf2_save_fat_file_next:
3992 0000B402 8B35[085F0100]      <1>       mov    esi, [csftdf_df_drv_dt] ; 25/03/2016
3993                              <1>
3994                              <1> csftdf2_save_fat_file_0:
3995 0000B408 5B                  <1>       pop    ebx ; *
3996                              <1>
3997                              <1> csftdf2_save_fat_file_1:
3998 0000B409 E813FFFFFF          <1>       call   csftdf2_write_file_sectors ; 19/03/2016
3999 0000B40E 0F827E010000        <1>         jc      csftdf2_rw_error ; eocc! or disk error!
4000                              <1>
4001 0000B414 09D2                <1>       or     edx, edx ; edx > 0 -> EOF
4002 0000B416 756D                <1>         jnz     short csftdf2_save_fat_file_3 ; 25/03/2016
```

```
4003                                <1>
4004 0000B418 803D[005F0100]00      <1>       cmp    byte [csftdf_percentage], 0
4005 0000B41F 76E8                  <1>       jna    short csftdf2_save_fat_file_1
4006                                <1>
4007 0000B421 B020                  <1>       mov    al, 20h
4008 0000B423 BF[150D0100]          <1>       mov    edi, percentagestr
4009 0000B428 AA                    <1>       stosb
4010 0000B429 AA                    <1>       stosb
4011 0000B42A A1[FC5E0100]          <1>       mov    eax, [csftdf_df_wbytes]
4012 0000B42F BA64000000            <1>       mov    edx, 100
4013 0000B434 F7E2                  <1>       mul    edx
4014 0000B436 8B0D[DC5E0100]        <1>       mov    ecx, [csftdf_filesize]
4015 0000B43C F7F1                  <1>       div    ecx
4016 0000B43E B10A                  <1>       mov    cl, 10
4017 0000B440 F6F1                  <1>       div    cl
4018 0000B442 80C430                <1>       add    ah, '0'
4019 0000B445 8827                  <1>       mov    [edi], ah
4020 0000B447 20C0                  <1>       and    al, al
4021 0000B449 740A                  <1>       jz     short csftdf2_save_fat_file_2
4022 0000B44B 4F                    <1>       dec    edi
4023 0000B44C 6698                  <1>       cbw
4024 0000B44E F6F1                  <1>       div    cl
4025 0000B450 80C430                <1>       add    ah, '0'
4026 0000B453 8827                  <1>       mov    [edi], ah
4027                                <1>       ;and   al, al
4028                                <1>       ;jz    short csftdf2_save_fat_file_2
4029                                <1>       ;dec   edi
4030                                <1>       ;mov   [edi], '1' ; 100%
4031                                <1>
4032                                <1> csftdf2_save_fat_file_2:
4033 0000B455 53                    <1>       push   ebx ; *
4034                                <1>
4035 0000B456 E802000000            <1>       call   csftdf2_print_wr_percentage ; 25/03/2016
4036                                <1>
4037 0000B45B EBA5                  <1>         jmp     csftdf2_save_fat_file_next
4038                                <1>
4039                                <1> csftdf2_print_wr_percentage:
4040                                <1>       ; Set cursor position
4041                                <1>       ; AH= 02h, BH= Page Number, DH= Row, DL= Column
4042 0000B45D 8A3D[015F0100]        <1>       mov    bh, [csftdf_videopage]
4043 0000B463 668B15[025F0100]      <1>       mov    dx, [csftdf_cursorpos]
4044 0000B46A B402                  <1>       mov    ah, 2
4045 0000B46C E82960FFFF            <1>       call   _int10h
4046                                <1>
4047 0000B471 BE[090D0100]          <1>       mov    esi, msg_writing
4048 0000B476 E8E2AEFFFF            <1>       call   print_msg
4049                                <1>
4050 0000B47B BE[150D0100]          <1>       mov    esi, percentagestr
4051                                <1>       ;call  print_msg
4052                                <1>       ;retn
4053 0000B480 E9D8AEFFFF            <1>       jmp    print_msg
4054                                <1>
4055                                <1> csftdf2_save_fat_file_3:
4056 0000B485 803D[005F0100]00      <1>       cmp    byte [csftdf_percentage], 0
4057 0000B48C 7611                  <1>         jna     csftdf2_save_fat_file_4 ; 25/03/2016
4058                                <1>
4059                                <1>       ; "100%"
4060 0000B48E BF[150D0100]          <1>       mov    edi, percentagestr
4061 0000B493 B031                  <1>       mov    al, '1'
4062 0000B495 AA                    <1>       stosb
4063 0000B496 B030                  <1>       mov    al, '0'
4064 0000B498 AA                    <1>       stosb
4065 0000B499 AA                    <1>       stosb
4066                                <1>
4067 0000B49A E8BEFFFFFF            <1>       call   csftdf2_print_wr_percentage
4068                                <1>
4069                                <1> csftdf2_save_fat_file_4:
4070 0000B49F 803D[DA5E0100]00      <1>       cmp    byte [DestinationFileFound], 0
4071 0000B4A6 7647                  <1>       jna    short csftdf2_save_fat_file_6
4072                                <1>
4073 0000B4A8 8B35[085F0100]        <1>       mov    esi, [csftdf_df_drv_dt] ; 31/03/2016
4074                                <1>
4075 0000B4AE A1[EC5E0100]          <1>       mov    eax, [csftdf_df_cluster] ; last cluster
4076 0000B4B3 E865040000            <1>       call   get_next_cluster
4077 0000B4B8 7235                  <1>       jc     short csftdf2_save_fat_file_6 ; eocc! or disk error!
4078                                <1>
4079 0000B4BA A1[EC5E0100]          <1>       mov    eax, [csftdf_df_cluster] ; last cluster
4080                                <1>       ;xor   ecx, ecx
4081                                <1>       ;mov   [FAT_ClusterCounter], ecx ; 0 ; reset
4082                                <1>       ;dec   ecx ; 0FFFFFFFFh
4083                                <1>       ;shr   ecx, 4 ; 28 bit ; 0FFFFFFFh
4084 0000B4BF B9FFFFFF0F           <1>       mov    ecx, 0FFFFFFFh
4085 0000B4C4 E87E070000            <1>       call   update_cluster
4086 0000B4C9 7224                  <1>       jc     short csftdf2_save_fat_file_6 ; really last cluster!?
4087                                <1>
4088 0000B4CB A3[EC5E0100]          <1>       mov    [csftdf_df_cluster], eax ; next cluster
4089                                <1>
4090                                <1>       ; byte [FAT_BuffValidData] = 2
4091 0000B4D0 E82F0A0000            <1>       call   save_fat_buffer
4092 0000B4D5 730E                  <1>       jnc    short csftdf2_save_fat_file_5
4093                                <1>
4094 0000B4D7 8B15[DC5E0100]        <1>       mov    edx, [csftdf_filesize]
4095 0000B4DD 8915[9E5E0100]        <1>       mov    [DestinationFile_DirEntry+DirEntry_FileSize], edx
4096 0000B4E3 EB58                  <1>       jmp    short csftdf2_save_fat_file_err3
4097                                <1>
4098                                <1> csftdf2_save_fat_file_5:
4099 0000B4E5 A1[EC5E0100]          <1>       mov    eax, [csftdf_df_cluster]
4100                                <1>
4101                                <1>       ; EAX = First cluster to be truncated/unlinked
4102                                <1>       ; ESI = Logical dos drive description table address
4103 0000B4EA E8580C0000            <1>       call   truncate_cluster_chain
4104                                <1>
```

```
4105                                <1> csftdf2_save_fat_file_6:
4106                                <1>       ; 28/03/2016
4107 0000B4EF BE[0D5E0100]         <1>       mov    esi, SourceFile_DirEntry+DirEntry_Attr ; +11 to + 18
4108 0000B4F4 BF[8D5E0100]         <1>       mov    edi, DestinationFile_DirEntry+DirEntry_Attr ; +11 to + 18
4109 0000B4F9 A4                   <1>       movsb ; +11
4110 0000B4FA A5                   <1>       movsd ; +12 .. +15
4111 0000B4FB 66A5                 <1>       movsw ; +16 .. +17
4112                                <1>             ; + 18
4113 0000B4FD 83C604               <1>       add    esi, 4
4114 0000B500 83C704               <1>       add    edi, 4
4115 0000B503 A5                   <1>       movsd  ; DirEntry_WrtTime ; +22 .. +25
4116                                <1>
4117 0000B504 8B15[DC5E0100]       <1>       mov    edx, [csftdf_filesize]
4118 0000B50A 8915[9E5E0100]       <1>       mov    [DestinationFile_DirEntry+DirEntry_FileSize], edx
4119                                <1>
4120 0000B510 E8B7F0FFFF           <1>       call   convert_current_date_time
4121                                <1>       ; DX = Date in dos dir entry format
4122                                <1>       ; AX = Time in dos dir entry format
4123 0000B515 EB4D                 <1>       jmp    short csftdf2_save_fat_file_7
4124                                <1>
4125                                <1> csftdf2_save_fat_file_err1:
4126 0000B517 5B                   <1>       pop    ebx ; *
4127                                <1> csftdf2_save_fat_file_err2:
4128 0000B518 A1[FC5E0100]         <1>       mov    eax, [csftdf_df_wbytes]
4129 0000B51D 8B15[9E5E0100]       <1>       mov    edx, [DestinationFile_DirEntry+DirEntry_FileSize]
4130 0000B523 39C2                 <1>       cmp    edx, eax
4131 0000B525 7616                 <1>       jna    short csftdf2_save_fat_file_err3
4132 0000B527 A1[EC5E0100]         <1>       mov    eax, [csftdf_df_cluster] ; last (empty) cluster
4133                                <1>       ; ESI = Logical dos drive description table address
4134 0000B52C E8160C0000           <1>       call   truncate_cluster_chain
4135 0000B531 720A                 <1>       jc     short csftdf2_save_fat_file_err3
4136 0000B533 A1[FC5E0100]         <1>       mov    eax, [csftdf_df_wbytes]
4137 0000B538 A3[9E5E0100]         <1>       mov    [DestinationFile_DirEntry+DirEntry_FileSize], eax
4138                                <1> csftdf2_save_fat_file_err3:
4139 0000B53D E88AF0FFFF           <1>       call   convert_current_date_time
4140                                <1>       ; DX = Date in dos dir entry format
4141                                <1>       ; AX = Time in dos dir entry format
4142 0000B542 C605[8F5E0100]00     <1>       mov    byte [DestinationFile_DirEntry+DirEntry_CrtTimeTenth], 0
4143 0000B549 66A3[905E0100]       <1>       mov    [DestinationFile_DirEntry+DirEntry_CrtTime], ax
4144 0000B54F 668915[925E0100]     <1>       mov    [DestinationFile_DirEntry+DirEntry_CrtDate], dx
4145 0000B556 66A3[985E0100]       <1>       mov    [DestinationFile_DirEntry+DirEntry_WrtTime], ax
4146 0000B55C 668915[9A5E0100]     <1>       mov    [DestinationFile_DirEntry+DirEntry_WrtDate], dx
4147 0000B563 F9                   <1>       stc
4148                                <1> csftdf2_save_fat_file_7:
4149 0000B564 9C                   <1>       pushf
4150 0000B565 668915[945E0100]     <1>       mov    [DestinationFile_DirEntry+DirEntry_LastAccDate], dx
4151 0000B56C BE[825E0100]         <1>       mov    esi, DestinationFile_DirEntry
4152 0000B571 BF00000800           <1>       mov    edi, Directory_Buffer
4153 0000B576 0FB70D[AA5E0100]     <1>       movzx  ecx, word [DestinationFile_DirEntryNumber] ; (<2048)
4154 0000B57D 66C1E105             <1>       shl    cx, 5 ; 32 * directory entry number
4155 0000B581 01CF                 <1>       add    edi, ecx
4156                                <1>       ;mov   ecx, 8
4157 0000B583 66B90800             <1>       mov    cx, 8
4158 0000B587 F3A5                 <1>       rep    movsd
4159 0000B589 9D                   <1>       popf
4160 0000B58A 730B                 <1>       jnc    short csftdf2_write_file_OK
4161                                <1>
4162                                <1> csftdf2_write_error:
4163                                <1>       ; 18/03/2016
4164 0000B58C B01D                 <1>       mov    al, 1Dh ; write error
4165 0000B58E EB02                 <1>       jmp    short csftdf2_rw_error
4166                                <1>
4167                                <1>       ; 16/03/2016
4168                                <1> csftdf2_read_error:
4169 0000B590 B011                 <1>       mov    al, 17 ; ; Drive not ready or read error!
4170                                <1> csftdf2_rw_error:
4171 0000B592 A2[D95E0100]         <1>       mov    [csftdf_rw_err], al
4172                                <1>
4173                                <1> csftdf2_write_file_OK:
4174                                <1>       ; 18/03/2016
4175 0000B597 C605[F05A0100]02     <1>       mov    byte [DirBuff_ValidData], 2
4176 0000B59E E8C7F0FFFF           <1>       call   save_directory_buffer
4177                                <1>
4178                                <1>       ; Update last modification date&time of destination
4179                                <1>       ; file's (parent) directory
4180 0000B5A3 E85DF1FFFF           <1>       call   update_parent_dir_lmdt
4181                                <1>       ;
4182 0000B5A8 A1[E05E0100]         <1>       mov    eax, [csftdf_sf_mem_addr] ; start address
4183                                <1>
4184 0000B5AD 21C0                 <1>       and    eax, eax
4185 0000B5AF 750E                 <1>       jnz    short csftdf2_dealloc_mblock
4186                                <1>
4187 0000B5B1 88C5                 <1>       mov    ch, al ; 0 (Cluster r/w, not full loading)
4188                                <1> csftdf2_dealloc_retn:
4189 0000B5B3 8A0D[D95E0100]       <1>       mov    cl, [csftdf_rw_err]
4190 0000B5B9 A1[EC5E0100]         <1>       mov    eax, [csftdf_df_cluster]
4191 0000B5BE C3                   <1>       retn
4192                                <1>
4193                                <1> csftdf2_dealloc_mblock:
4194 0000B5BF 8B0D[E45E0100]       <1>       mov    ecx, [csftdf_sf_mem_bsize] ; block size
4195 0000B5C5 E866A0FFFF           <1>       call   deallocate_memory_block
4196 0000B5CA B5FF                 <1>         mov     ch, 0FFh ; (File was full loaded at memory)
4197 0000B5CC EBE5                 <1>       jmp    short csftdf2_dealloc_retn
4198                                <1>
4199                                <1> csftdf2_save_fs_file:
4200                                <1>       ; 16/10/2016 (1Dh -> 18)
4201                                <1>       ; temporary - (21/03/2016)
4202 0000B5CE B812000000           <1>       mov    eax, 18 ; write error
4203 0000B5D3 F9                   <1>       stc
4204 0000B5D4 C3                   <1>       retn
4205                                <1>
4206                                <1> create_file:
```

```
                                <1>       ; 16/10/2016
                                <1>       ; 24/03/2016, 31/03/2016
                                <1>       ; 20/03/2016, 21/03/2016, 23/03/2016
                                <1>       ; 19/03/2016 (TRDOS 396 = TRDOS v2.0)
                                <1>       ; 03/09/2011 (FILE.ASM, 'proc_create_file')
                                <1>       ; 09/08/2010
                                <1>       ;
                                <1>       ; INPUT ->
                                <1>       ;     EAX = File Size
                                <1>       ;     ESI = ASCIIZ File Name
                                <1>       ;     CL = File Attributes
                                <1>       ;     EBX = FFFFFFFFh -> create empty file
                                <1>       ;                  (only for FAT fs)
                                <1>       ; OUTPUT ->
                                <1>       ;     CF = 0 ->
                                <1>       ;     EAX = New file's first cluster
                                <1>       ;     ESI = Logical Dos Drv Descr. Table Addr.
                                <1>       ;     EBX = offset CreateFile_Size
                                <1>       ;     ECX = Sectors per cluster (<256)
                                <1>       ;     EDX = Directory entry index/number (<65536)
                                <1>       ;     CF = 1 -> error code in AL
                                <1>
                                <1> ;     test   cl, 18h (directory or volume name)
                                <1> ;     jnz    short loc_createfile_access_denied
0000B5D5 80E107                 <1>       and    cl, 07h ; S, H, R
0000B5D8 880D[285F0100]         <1>         mov     [createfile_attrib], cl
                                <1>
0000B5DE 89D9                   <1>       mov    ecx, ebx
0000B5E0 89F3                   <1>       mov    ebx, esi ; ASCIIZ File Name address
0000B5E2 29D2                   <1>       sub    edx, edx
0000B5E4 8A35[C6520100]         <1>         mov     dh, [Current_Drv]
0000B5EA BE00010900             <1>         mov     esi, Logical_DOSDisks
0000B5EF 01D6                   <1>       add    esi, edx
                                <1>
0000B5F1 8815[335F0100]         <1>       mov    [createfile_UpdatePDir], dl ; 0 ; 31/03/2016
                                <1>
                                <1>       ; LD_DiskType = 0 for write protection (read only)
0000B5F7 807E0101               <1>       cmp    byte [esi+LD_DiskType], 1 ; 0 = Invalid
0000B5FB 730A                   <1>       jnb    short loc_createfile_check_file_sytem
                                <1>       ; 16/10/2016 (TRDOS Error code: 30, disk write protected)
0000B5FD B81E000000             <1>       mov    eax, 30 ; 13h, MSDOS err : Disk write-protected
0000B602 66BA0000               <1>       mov    dx, 0
                                <1>       ; err retn: EDX = 0, EBX = File name offset
                                <1>       ; ESI -> Dos drive description table address
0000B606 C3                     <1>       retn
                                <1>
                                <1> ;loc_createfile_access_denied:
                                <1> ;     mov    eax, 05h ; access denied (invalid attributes input)
                                <1> ;     stc
                                <1> ;     retn
                                <1>
                                <1> loc_createfile_check_file_sytem:
0000B607 807E0301               <1>       cmp    byte [esi+LD_FATType], 1
0000B60B 730A                   <1>       jnb    short loc_createfile_chk_empty_FAT_file_sign1
                                <1>
0000B60D A3[145F0100]           <1>       mov    [createfile_size], eax
                                <1>       ; ESI = Logical Dos Drive Description Table address
                                <1>       ; EBX = ASCIIZ File Name address
0000B612 E9FE020000             <1>       jmp    create_fs_file
                                <1>
                                <1> loc_createfile_chk_empty_FAT_file_sign1:
                                <1>       ; ECX = FFFFFFFFh -> create empty file if drive has FAT fs
0000B617 41                     <1>       inc    ecx
0000B618 7506                   <1>       jnz    short loc_createfile_chk_empty_FAT_file_sign2
0000B61A 890D[145F0100]         <1>       mov    [createfile_size], ecx ; 0 ; empty file
                                <1>
                                <1> loc_createfile_chk_empty_FAT_file_sign2:
                                <1>       ; 23/03/2016
0000B620 668B4E11               <1>       mov    cx, [esi+LD_BPB+BytesPerSec]
0000B624 66890D[305F0100]       <1>       mov    [createfile_BytesPerSec], cx
                                <1>
                                <1>       ; EBX = ASCIIZ File Name address
0000B62B 0FB65613               <1>       movzx  edx, byte [esi+LD_BPB+SecPerClust]
0000B62F 8815[295F0100]         <1>       mov    [createfile_SecPerClust], dl
0000B635 8B4E74                 <1>       mov    ecx, [esi+LD_FreeSectors]
0000B638 39D1                   <1>       cmp    ecx, edx ; byte [createfile_SecPerClust]
0000B63A 7306                   <1>       jnb    short loc_create_fat_file
                                <1>
                                <1> loc_createfile_insufficient_disk_space:
0000B63C B827000000             <1>       mov    eax, 27h
                                <1> loc_createfile_gffc_retn:
0000B641 C3                     <1>       retn
                                <1>
                                <1> loc_create_fat_file:
0000B642 891D[0C5F0100]         <1>       mov    [createfile_Name_Offset], ebx
0000B648 890D[105F0100]         <1>       mov    [createfile_FreeSectors], ecx
                                <1>
                                <1> loc_createfile_gffc_1:
0000B64E E821050000             <1>       call   get_first_free_cluster
0000B653 72EC                   <1>       jc     short loc_createfile_gffc_retn
                                <1>
0000B655 A3[185F0100]           <1>       mov    [createfile_FFCluster], eax
                                <1>
                                <1> loc_createfile_locate_ffe_on_directory:
                                <1>       ; Current directory fcluster <> Directory buffer cluster
                                <1>       ; Current directory will be reloaded by
                                <1>       ; 'locate_current_dir_file' procedure
                                <1>       ;
                                <1>       ; ESI = Logical Dos Drv Desc. Table Adress
0000B65A 56                     <1>       push   esi ; *
0000B65B 31C0                   <1>       xor    eax, eax
                                <1>
```

```
4309 0000B65D A3[E65A0100]        <1>       mov    dword [FAT_ClusterCounter], eax ; 0
4310                              <1>       ; 21/03/2016
4311 0000B662 A2[325F0100]        <1>       mov    byte [createfile_wfc], al ; 0
4312                              <1>
4313 0000B667 89C1                <1>       mov    ecx, eax
4314 0000B669 6649                <1>       dec    cx ; FFFFh
4315                              <1>       ; CX = FFFFh -> find first deleted or free entry
4316                              <1>       ; ESI would be ASCIIZ filename address if the call
4317                              <1>       ; would not be for first free or deleted dir entry
4318 0000B66B E8D6E7FFFF          <1>       call   locate_current_dir_file
4319 0000B670 0F83EE000000        <1>       jnc    loc_createfile_set_ff_dir_entry
4320 0000B676 5E                  <1>       pop    esi ; *
4321                              <1>        ; ESI = Logical DOS Drv. Description Table Address
4322 0000B677 83F802              <1>       cmp    eax, 2
4323 0000B67A 7402                <1>       je     short loc_createfile_add_new_cluster
4324                              <1> loc_createfile_locate_file_stc_retn:
4325 0000B67C F9                  <1>       stc
4326 0000B67D C3                  <1>       retn
4327                              <1>
4328                              <1> loc_createfile_add_new_cluster:
4329 0000B67E 803D[C5520100]02    <1>       cmp    byte [Current_FATType], 2
4330                              <1>       ;cmp   byte [esi+LD_FATType], 2
4331 0000B685 770C                <1>       ja     short loc_createfile_add_new_cluster_check_fsc
4332 0000B687 803D[C4520100]01    <1>       cmp    byte [Current_Dir_Level], 1
4333                              <1>       ;cmp   byte [esi+LD_CDirLevel], 1
4334 0000B68E 7303                <1>       jnb    short loc_createfile_add_new_cluster_check_fsc
4335                              <1>
4336                              <1>       ;mov   eax, 12
4337 0000B690 B00C                <1>       mov    al, 12 ; No more files
4338                              <1>
4339                              <1> loc_createfile_anc_retn:
4340 0000B692 C3                  <1>       retn
4341                              <1>
4342                              <1> loc_createfile_add_new_cluster_check_fsc:
4343 0000B693 8B0D[105F0100]      <1>       mov    ecx, [createfile_FreeSectors]
4344 0000B699 0FB605[295F0100]    <1>       movzx  eax, byte [createfile_SecPerClust]
4345 0000B6A0 66D1E0              <1>       shl    ax, 1 ; AX = 2 * AX
4346 0000B6A3 39C1                <1>       cmp    ecx, eax
4347 0000B6A5 7295                <1>         jb   short loc_createfile_insufficient_disk_space
4348                              <1>
4349                              <1> loc_createfile_add_new_subdir_cluster:
4350 0000B6A7 8B15[F55A0100]      <1>       mov    edx, [DirBuff_Cluster]
4351 0000B6AD 8915[1C5F0100]      <1>       mov    [createfile_LastDirCluster], edx
4352                              <1>
4353 0000B6B3 A1[185F0100]        <1>       mov    eax, [createfile_FFCluster]
4354 0000B6B8 E846040000          <1>       call   load_FAT_sub_directory
4355 0000B6BD 72D3                <1>       jc     short loc_createfile_anc_retn
4356                              <1>
4357                              <1> pass_createfile_add_new_subdir_cluster:
4358                              <1>       ;movzx eax, word [esi+LD_BPB+BytesPerSec]
4359 0000B6BF 0FB705[305F0100]    <1>       movzx  eax, word [createfile_BytesPerSec] ; 23/03/2016
4360 0000B6C6 F7E1                <1>       mul    ecx ; ecx = directory buffer sector count
4361 0000B6C8 89C1                <1>       mov    ecx, eax
4362 0000B6CA C1E902              <1>       shr    ecx, 2 ; dword count
4363 0000B6CD 29C0                <1>       sub    eax, eax ; 0
4364 0000B6CF F3AB                <1>       rep    stosd
4365                              <1>       ;
4366 0000B6D1 C605[F05A0100]02    <1>       mov    byte [DirBuff_ValidData], 2
4367 0000B6D8 E88DEFFFFF          <1>       call   save_directory_buffer
4368 0000B6DD 72B3                <1>       jc     short loc_createfile_anc_retn
4369                              <1>
4370                              <1> loc_createfile_save_added_subdir_cluster:
4371 0000B6DF A1[1C5F0100]        <1>       mov    eax, [createfile_LastDirCluster]
4372 0000B6E4 8B0D[185F0100]      <1>       mov    ecx, [createfile_FFCluster]
4373 0000B6EA E858050000          <1>       call   update_cluster
4374 0000B6EF 7304                <1>       jnc    short loc_createfile_save_fat_buffer_0
4375 0000B6F1 09C0                <1>       or     eax, eax ; EAX = 0 -> cluster value is 0 or eocc
4376 0000B6F3 751A                <1>       jnz    short loc_createfile_save_fat_buffer_stc_retn
4377                              <1>
4378                              <1> loc_createfile_save_fat_buffer_0:
4379 0000B6F5 A1[185F0100]        <1>       mov    eax, [createfile_FFCluster]
4380 0000B6FA A3[1C5F0100]        <1>       mov    [createfile_LastDirCluster], eax
4381 0000B6FF B9FFFFFF0F          <1>       mov    ecx, 0FFFFFFFh ; 28 bit
4382 0000B704 E83E050000          <1>       call   update_cluster
4383 0000B709 7306                <1>       jnc    short loc_createfile_save_fat_buffer_1
4384 0000B70B 09C0                <1>       or     eax, eax ; Was it free cluster
4385 0000B70D 7402                <1>       jz     short loc_createfile_save_fat_buffer_1
4386                              <1>
4387                              <1> loc_createfile_save_fat_buffer_stc_retn:
4388 0000B70F F9                  <1>       stc
4389                              <1> loc_createfile_save_fat_buffer_retn:
4390                              <1> loc_createfile_gffc_2_stc_retn:
4391 0000B710 C3                  <1>       retn
4392                              <1>
4393                              <1> loc_createfile_save_fat_buffer_1:
4394                              <1>       ; byte [FAT_BuffValidData] = 2
4395 0000B711 E8EE070000          <1>       call   save_fat_buffer
4396 0000B716 72F8                <1>       jc     short loc_createfile_save_fat_buffer_retn
4397                              <1>
4398 0000B718 803D[E65A0100]01    <1>       cmp    byte [FAT_ClusterCounter], 1
4399 0000B71F 7222                <1>       jb     short loc_createfile_save_fat_buffer_2
4400                              <1>
4401                              <1>       ; ESI = Logical DOS Drive Description Table address
4402 0000B721 A1[E65A0100]        <1>       mov    eax, [FAT_ClusterCounter]
4403                              <1>
4404 0000B726 C605[E65A0100]00    <1>       mov    byte [FAT_ClusterCounter], 0 ; 21/03/2016
4405                              <1>
4406 0000B72D 66BB01FF            <1>       mov    bx, 0FF01h ; add free clusters
4407 0000B731 E863080000          <1>       call   calculate_fat_freespace
4408                              <1>
4409                              <1>       ;inc   eax ; 0FFFFFFFFh -> 0 ; recalculation is needed!
4410                              <1>       ;jnz   short loc_createfile_save_fat_buffer_2
```

```
4411                                    <1>
4412                                    <1>       ; ecx > 0 -> Recalculation is needed
4413 0000B736 09C9                     <1>       or     ecx, ecx
4414 0000B738 7409                     <1>       jz     short loc_createfile_save_fat_buffer_2
4415                                    <1>
4416 0000B73A 66BB00FF                 <1>       mov    bx, 0FF00h ; ; recalculate free space
4417 0000B73E E856080000               <1>       call   calculate_fat_freespace
4418                                    <1>
4419                                    <1> loc_createfile_save_fat_buffer_2:
4420                                    <1>       ;call  update_parent_dir_lmdt
4421                                    <1>
4422                                    <1> loc_createfile_gffc_2:
4423 0000B743 E82C040000               <1>       call   get_first_free_cluster
4424 0000B748 72C6                     <1>       jc     short loc_createfile_gffc_2_stc_retn
4425                                    <1>
4426 0000B74A A3[185F0100]             <1>       mov    [createfile_FFCluster], eax
4427                                    <1>
4428 0000B74F A1[1C5F0100]             <1>       mov    eax, [createfile_LastDirCluster]
4429                                    <1>
4430 0000B754 E8AA030000               <1>       call   load_FAT_sub_directory
4431 0000B759 72B5                     <1>       jc     short loc_createfile_gffc_2_stc_retn
4432                                    <1>
4433 0000B75B BF00000800               <1>       mov    edi, Directory_Buffer
4434                                    <1>
4435 0000B760 6629DB                   <1>       sub    bx, bx ; directory entry index/number = 0
4436                                    <1>
4437 0000B763 56                       <1>       push   esi ; * ; 23/03/2016
4438                                    <1>
4439                                    <1> loc_createfile_set_ff_dir_entry:
4440 0000B764 66891D[2A5F0100]         <1>       mov    [createfile_DirIndex], bx
4441                                    <1>
4442                                    <1>         ; EDI = Directory entry address
4443 0000B76B 8B35[0C5F0100]           <1>       mov    esi, [createfile_Name_Offset]
4444 0000B771 A1[185F0100]             <1>       mov    eax, [createfile_FFCluster]
4445 0000B776 A3[205F0100]             <1>       mov    [createfile_Cluster], eax ; 24/03/2016
4446 0000B77B B5FF                     <1>       mov    ch, 0FFh
4447 0000B77D 8A0D[285F0100]           <1>         mov  cl, [createfile_attrib] ; file attributes
4448                                    <1>       ; CH > 0 -> File size is in [EBX]
4449 0000B783 BB[145F0100]             <1>       mov    ebx, createfile_size
4450                                    <1>
4451 0000B788 E800EEFFFF               <1>       call   make_directory_entry
4452                                    <1>
4453 0000B78D 5E                       <1>       pop    esi ; * ; ESI = Logical Dos Drv Desc. Table address
4454                                    <1>
4455 0000B78E C605[F05A0100]02         <1>       mov    byte [DirBuff_ValidData], 2
4456 0000B795 E8D0EEFFFF               <1>       call   save_directory_buffer
4457 0000B79A 7221                     <1>       jc     short loc_createfile_set_ff_dir_entry_retn
4458                                    <1>
4459 0000B79C C605[335F0100]01         <1>       mov    byte [createfile_UpdatePDir], 1 ; 31/03/2016
4460                                    <1>
4461                                    <1> loc_createfile_get_set_write_file_cluster:
4462 0000B7A3 A1[145F0100]             <1>       mov    eax, [createfile_size]
4463 0000B7A8 09C0                     <1>       or     eax, eax
4464 0000B7AA 7570                     <1>       jnz    short loc_createfile_get_set_wfc_cont
4465 0000B7AC 40                       <1>       inc    eax
4466                                    <1>       ; 23/03/2016
4467 0000B7AD 0FB61D[295F0100]         <1>       movzx  ebx, byte [createfile_SecPerClust]
4468                                    <1>       ;movzx ecx, word [esi+LD_BPB+BytesPerSec] ; 512
4469 0000B7B4 0FB70D[305F0100]         <1>         movzx   ecx, word [createfile_BytesPerSec] ; 512
4470 0000B7BB EB7C                     <1>       jmp    loc_createfile_set_cluster_count
4471                                    <1>
4472                                    <1> loc_createfile_set_ff_dir_entry_retn:
4473 0000B7BD C3                       <1>       retn
4474                                    <1>
4475                                    <1> loc_createfile_write_fcluster_to_disk:
4476 0000B7BE 034668                   <1>       add    eax, [esi+LD_DATABegin] ; convert to physical address
4477 0000B7C1 BB00000700               <1>       mov    ebx, Cluster_Buffer
4478                                    <1>       ; ESI = Logical DOS Drv. Desc. Tbl. address
4479                                    <1>       ; EAX = Disk address
4480                                    <1>       ; EBX = Sector Buffer
4481                                    <1>       ; ECX = sectors per cluster
4482 0000B7C6 E8D8390000               <1>       call   disk_write
4483 0000B7CB 7211                     <1>       jc     short loc_createfile_dsk_wr_err
4484                                    <1>
4485                                    <1> loc_createfile_update_fat_cluster:
4486                                    <1>       ; 21/03/2016
4487 0000B7CD 803D[325F0100]00         <1>       cmp    byte [createfile_wfc], 0
4488 0000B7D4 7712                     <1>       ja     short loc_createfile_update_fat_cluster_n1
4489                                    <1>
4490 0000B7D6 FE05[325F0100]           <1>       inc    byte [createfile_wfc] ; 1
4491 0000B7DC EB24                     <1>       jmp    short loc_createfile_update_fat_cluster_n2
4492                                    <1>
4493                                    <1> loc_createfile_dsk_wr_err:
4494                                    <1>       ; 16/10/2016 (1Dh -> 18)
4495                                    <1>       ; 23/03/2016
4496 0000B7DE B812000000               <1>       mov    eax, 18 ; Drive not ready or write error !
4497 0000B7E3 E9BD000000               <1>       jmp    loc_createfile_stc_retn
4498                                    <1>
4499                                    <1> loc_createfile_update_fat_cluster_n1:
4500 0000B7E8 A1[245F0100]             <1>       mov    eax, [createfile_PCluster]
4501 0000B7ED 8B0D[205F0100]           <1>       mov    ecx, [createfile_Cluster]
4502 0000B7F3 E84F040000               <1>       call   update_cluster
4503 0000B7F8 7308                     <1>       jnc    short loc_createfile_update_fat_cluster_n2
4504 0000B7FA 09C0                     <1>       or     eax, eax ; EAX = 0 -> cluster value is 0 or eocc
4505 0000B7FC 0F85A3000000             <1>       jnz    loc_createfile_stc_retn
4506                                    <1>
4507                                    <1> loc_createfile_update_fat_cluster_n2:
4508 0000B802 A1[205F0100]             <1>         mov  eax, [createfile_Cluster]
4509 0000B807 B9FFFFFF0F               <1>       mov    ecx, 0FFFFFFFh
4510 0000B80C E836040000               <1>       call   update_cluster
4511 0000B811 734E                     <1>       jnc    short loc_createfile_save_fat_buffer_3
4512 0000B813 09C0                     <1>       or     eax, eax ; EAX = 0 -> cluster value is 0 or eocc
```

```
4513 0000B815 744A                <1>       jz     short loc_createfile_save_fat_buffer_3
4514                              <1>
4515                              <1> loc_createfile_upd_fat_fcluster_stc_retn:
4516 0000B817 E989000000          <1>       jmp    loc_createfile_stc_retn
4517                              <1>
4518                              <1> loc_createfile_get_set_wfc_cont:
4519                              <1>       ;movzx ecx, word [esi+LD_BPB+BytesPerSec] ; 512
4520 0000B81C 0FB70D[305F0100]    <1>       movzx  ecx, word [createfile_BytesPerSec] ; 512
4521 0000B823 01C8                <1>       add    eax, ecx
4522 0000B825 48                  <1>       dec    eax  ; add eax, 511
4523 0000B826 29D2                <1>       sub    edx, edx
4524 0000B828 F7F1                <1>       div    ecx
4525 0000B82A 0FB61D[295F0100]    <1>       movzx  ebx, byte [createfile_SecPerClust]
4526 0000B831 01D8                <1>       add    eax, ebx
4527 0000B833 48                  <1>       dec    eax  ; add eax, SecPerClust - 1
4528 0000B834 6631D2              <1>       xor    dx, dx
4529 0000B837 F7F3                <1>       div    ebx
4530                              <1>
4531                              <1> loc_createfile_set_cluster_count:
4532 0000B839 A3[2C5F0100]        <1>       mov    [createfile_CCount], eax
4533                              <1>
4534 0000B83E BF00000700          <1>       mov    edi, Cluster_Buffer
4535 0000B843 89C8                <1>       mov    eax, ecx ; Bytes per Sector
4536 0000B845 F7E3                <1>       mul    ebx ; Sectors per Cluster
4537                              <1>       ; EAX = Bytes per Cluster
4538 0000B847 89C1                <1>       mov    ecx, eax
4539 0000B849 C1E902              <1>       shr    ecx, 2 ; dword count
4540 0000B84C 31C0                <1>       xor    eax, eax
4541 0000B84E F3AB                <1>       rep    stosd ; clear cluster buffer
4542                              <1>
4543 0000B850 A1[205F0100]        <1>       mov    eax, [createfile_Cluster] ; 24/03/2016
4544                              <1>
4545 0000B855 89D9                <1>       mov    ecx, ebx
4546                              <1>
4547                              <1> loc_createfile_get_set_wf_fclust_cont:
4548 0000B857 83E802              <1>       sub    eax, 2
4549 0000B85A F7E1                <1>       mul    ecx
4550                              <1>       ; EAX = Logical DOS disk address (offset)
4551 0000B85C E95DFFFFFF          <1>         jmp     loc_createfile_write_fcluster_to_disk
4552                              <1>
4553                              <1> loc_createfile_save_fat_buffer_3:
4554                              <1>       ; byte [FAT_BuffValidData] = 2
4555 0000B861 E89E060000          <1>       call   save_fat_buffer
4556 0000B866 723D                <1>       jc     loc_createfile_stc_retn
4557                              <1>
4558                              <1>       ; 21/03/2016
4559 0000B868 803D[E65A0100]01    <1>       cmp    byte [FAT_ClusterCounter], 1
4560 0000B86F 721B                <1>       jb     short loc_createfile_save_fat_buffer_4
4561                              <1>
4562                              <1>       ; ESI = Logical DOS Drive Description Table address
4563 0000B871 A1[E65A0100]        <1>       mov    eax, [FAT_ClusterCounter]
4564 0000B876 66BB01FF            <1>       mov    bx, 0FF01h ; add free clusters
4565 0000B87A E81A070000          <1>       call   calculate_fat_freespace
4566                              <1>
4567                              <1>       ;inc   eax ; 0FFFFFFFFh -> 0 ; recalculation is needed!
4568                              <1>       ;jnz   short loc_createfile_save_fat_buffer_4
4569                              <1>
4570                              <1>       ; ecx > 0 -> Recalculation is needed
4571 0000B87F 09C9                <1>       or     ecx, ecx
4572 0000B881 7409                <1>       jz     short loc_createfile_save_fat_buffer_4
4573                              <1>
4574 0000B883 66BB00FF            <1>       mov    bx, 0FF00h ; ; recalculate free space
4575 0000B887 E80D070000          <1>       call   calculate_fat_freespace
4576                              <1>
4577                              <1> loc_createfile_save_fat_buffer_4:
4578 0000B88C FF0D[2C5F0100]      <1>       dec    dword [createfile_CCount]
4579                              <1>       ;jz    short loc_createfile_upd_dir_modif_date_time
4580 0000B892 743F                <1>       jz     short loc_createfile_stc_retn_cc ; 31/03/2016
4581                              <1>
4582                              <1> loc_createfile_get_set_write_next_cluster:
4583 0000B894 E8DB020000          <1>       call   get_first_free_cluster
4584 0000B899 720A                <1>       jc     short loc_createfile_stc_retn
4585                              <1>
4586                              <1> loc_createfile_get_set_write_next_cluster_1:
4587 0000B89B 83F8FF              <1>       cmp    eax, 0FFFFFFFFh
4588 0000B89E 7213                <1>       jb     short loc_createfile_get_set_write_next_cluster_2
4589                              <1>
4590                              <1> loc_createfile_wnc_insufficient_disk_space:
4591 0000B8A0 B827000000          <1>       mov    eax, 27h ; Insufficient disk space
4592                              <1>
4593                              <1> loc_createfile_stc_retn:
4594 0000B8A5 803D[325F0100]01    <1>       cmp    byte [createfile_wfc], 1
4595 0000B8AC 7324                <1>       jnb    short loc_createfile_err_retn
4596 0000B8AE C3                  <1>       retn
4597                              <1>
4598                              <1> loc_createfile_wnc_inv_format_retn:
4599                              <1>       ;mov   eax, 28
4600 0000B8AF B01C                <1>       mov    al, 28 ; Invalid format
4601 0000B8B1 EBF2                <1>       jmp    short loc_createfile_stc_retn
4602                              <1>
4603                              <1> loc_createfile_get_set_write_next_cluster_2:
4604 0000B8B3 83F802              <1>       cmp    eax, 2
4605 0000B8B6 72F7                <1>       jb     short loc_createfile_wnc_inv_format_retn
4606                              <1>
4607                              <1> loc_createfile_get_set_write_next_cluster_3:
4608 0000B8B8 8B0D[205F0100]      <1>       mov    ecx, [createfile_Cluster]
4609 0000B8BE A3[205F0100]        <1>       mov    [createfile_Cluster], eax
4610 0000B8C3 890D[245F0100]      <1>       mov    [createfile_PCluster], ecx
4611 0000B8C9 0FB60D[295F0100]    <1>       movzx  ecx, byte [createfile_SecPerClust]
4612 0000B8D0 EB85                <1>       jmp    short loc_createfile_get_set_wf_fclust_cont
4613                              <1>
4614                              <1> loc_createfile_err_retn:
```

```
4615 0000B8D2 F9                <1>       stc
4616                            <1>
4617                            <1> ;loc_createfile_upd_dir_modif_date_time:
4618                            <1> loc_createfile_stc_retn_cc: ; 31/03/2016
4619 0000B8D3 9C                <1>       pushf  ; cpu is here for an error return or completion
4620 0000B8D4 50                <1>       push   eax ; error code if cf = 1
4621                            <1>
4622                            <1>       ;call  update_parent_dir_lmdt
4623                            <1>
4624                            <1> ;loc_createfile_stc_retn_cc:
4625 0000B8D5 A1[E65A0100]      <1>       mov    eax, [FAT_ClusterCounter]
4626 0000B8DA 09C0              <1>       or     eax, eax
4627 0000B8DC 741A              <1>       jz     short loc_createfile_stc_retn_pop_eax
4628 0000B8DE 8A3D[C6520100]    <1>       mov    bh, [Current_Drv]
4629 0000B8E4 B301              <1>       mov    bl, 01h ; BL = 1 -> add clusters
4630                            <1>       ; NOTE: EAX value will be added to Free Cluster Count
4631                            <1>       ; (If EAX value is negative, Free Cluster Count will be decreased)
4632 0000B8E6 E8AE060000        <1>       call   calculate_fat_freespace
4633                            <1>         ; ESI = Logical DOS Drive Description Table Address
4634                            <1>         ;jc  short loc_createfile_stc_retn_pop_eax_cf
4635 0000B8EB 21C9              <1>       and    ecx, ecx ; cx = 0 -> valid free sector count
4636 0000B8ED 7409              <1>       jz     short loc_createfile_stc_retn_pop_eax
4637                            <1>
4638                            <1> loc_createfile_stc_retn_recalc_FAT_freespace:
4639 0000B8EF 66BB00FF          <1>       mov    bx, 0FF00h ; bh = 0FFh ->
4640                            <1>       ; ESI = Logical DOS Drv DT Addr
4641                            <1>       ; BL = 0 -> Recalculate
4642 0000B8F3 E8A1060000        <1>       call   calculate_fat_freespace
4643                            <1>
4644                            <1> loc_createfile_stc_retn_pop_eax:
4645 0000B8F8 58                <1>       pop    eax
4646 0000B8F9 9D                <1>       popf
4647 0000B8FA 7218              <1>       jc     short loc_createfile_retn
4648                            <1>
4649                            <1> loc_createfile_retn_fcluster:
4650 0000B8FC A1[185F0100]      <1>       mov    eax, [createfile_FFCluster]
4651 0000B901 BB[145F0100]      <1>       mov    ebx, createfile_size
4652                            <1>       ;movzx ecx, byte [esi+LD_BPB+SecPerClust]
4653 0000B906 0FB60D[295F0100]  <1>       movzx  ecx, byte [createfile_SecPerClust] ; 23/03/2016
4654 0000B90D 0FB715[2A5F0100]  <1>       movzx  edx, word [createfile_DirIndex]
4655                            <1>
4656                            <1> loc_createfile_retn:
4657 0000B914 C3                <1>       retn
4658                            <1>
4659                            <1> create_fs_file:
4660                            <1>       ; temporary (21/03/2016)
4661 0000B915 C3                <1>       retn
4662                            <1>
4663                            <1> delete_fs_file:
4664                            <1>       ; temporary (28/02/2016)
4665 0000B916 C3                <1>       retn
4666                            <1>
4667                            <1> rename_fs_file_or_directory:
4668 0000B917 C3                <1>       retn
4669                            <1>
4670                            <1> make_fs_directory:
4671                            <1>       ; temporary (21/02/2016)
4672 0000B918 C3                <1>       retn
4673                            <1>
4674                            <1> add_new_fs_section:
4675                            <1>       ; temporary (11/03/2016)
4676 0000B919 C3                <1>       retn
4677                            <1>
4678                            <1> delete_fs_directory_entry:
4679                            <1>       ; temporary (11/03/2016)
4680 0000B91A C3                <1>       retn
4681                            <1>
4682                            <1> csftdf2_read_fs_file_sectors:
4683                            <1>       ; temporary (19/03/2016)
4684 0000B91B C3                <1>       retn
4685                            <1>
4686                            <1> csftdf2_write_fs_file_sectors:
4687                            <1>       ; temporary (19/03/2016)
4688 0000B91C C3                <1>       retn
2308                                %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
```

```
                        <1>       ;     cf = 1 & EAX > 0 -> Error
                        <1>       ;     ECX = Current/Previous cluster (if CF = 0)
                        <1>       ;     EAX = Next Cluster Number (32 bit)
                        <1>       ;
                        <1>       ; (Modified registers: EAX, ECX, EBX, EDX)
                        <1>
0000B91D A3[DA5A0100]   <1>       mov   [FAT_CurrentCluster], eax
                        <1> check_next_cluster_fat_type:
0000B922 29D2           <1>       sub   edx, edx ; 0
0000B924 807E0302       <1>         cmp     byte [esi+LD_FATType], 2
0000B928 7250           <1>       jb    short get_FAT12_next_cluster
0000B92A 0F87AF000000   <1>         ja      get_FAT32_next_cluster
                        <1> get_FAT16_next_cluster:
0000B930 BB00030000     <1>       mov   ebx, 300h ;768
0000B935 F7F3           <1>       div   ebx
                        <1>       ; EAX = Count of 3 FAT sectors
                        <1>       ; EDX = Cluster Offset (< 768)
0000B937 66D1E2         <1>       shl   dx, 1 ; Multiply by 2
0000B93A 89D3           <1>       mov   ebx, edx ; Byte Offset
0000B93C 81C3001C0900   <1>       add   ebx, FAT_Buffer
0000B942 66BA0300       <1>       mov   dx, 3
0000B946 F7E2           <1>       mul   edx
                        <1>       ; EAX = FAT Sector (<= 256)
                        <1>       ; EDX = 0
0000B948 8A0E           <1>       mov   cl, [esi+LD_Name]
0000B94A 803D[DE5A0100]00 <1>     cmp   byte [FAT_BuffValidData], 0
0000B951 0F86CC000000   <1>         jna     load_FAT_sectors0
0000B957 3A0D[DF5A0100] <1>       cmp   cl, [FAT_BuffDrvName]
0000B95D 0F85C0000000   <1>         jne     load_FAT_sectors0
0000B963 3B05[E25A0100] <1>       cmp   eax, [FAT_BuffSector]
0000B969 0F85BA000000   <1>         jne     load_FAT_sectors1
                        <1>       ;movzx eax, word [ebx]
0000B96F 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
0000B972 6683F8F7       <1>       cmp   ax, 0FFF7h
0000B976 725A           <1>       jb    short loc_pass_gnc_FAT16_eoc_check
                        <1>       ; ax >= FFF7h (cluster 0002h to FFF6h is valid, in use)
0000B978 EB56           <1>       jmp   short loc_pass_gnc_FAT16_eoc_check_xor_eax
                        <1>
                        <1> get_FAT12_next_cluster:
0000B97A BB00040000     <1>       mov   ebx, 400h ;1024
0000B97F F7F3           <1>       div   ebx
                        <1>       ; EAX = Count of 3 FAT sectors
                        <1>       ; EDX = Cluster Offset (< 1024)
0000B981 6650           <1>       push  ax
0000B983 66B80300       <1>       mov   ax, 3
0000B987 66F7E2         <1>       mul   dx    ; Multiply by 3
0000B98A 66D1E8         <1>       shr   ax, 1  ; Divide by 2
0000B98D 6689C3         <1>         mov  bx, ax       ; Byte Offset
0000B990 81C3001C0900   <1>       add   ebx, FAT_Buffer
0000B996 6658           <1>       pop   ax
0000B998 66BA0300       <1>       mov   dx, 3
0000B99C F7E2           <1>       mul   edx
                        <1>       ; EAX = FAT Sector (<= 12)
                        <1>       ; EDX = 0
0000B99E 8A0E           <1>       mov   cl, [esi+LD_Name]
0000B9A0 803D[DE5A0100]00 <1>     cmp   byte [FAT_BuffValidData], 0
0000B9A7 767A           <1>       jna   short load_FAT_sectors0
0000B9A9 3A0D[DF5A0100] <1>       cmp   cl, [FAT_BuffDrvName]
0000B9AF 7572           <1>       jne   short load_FAT_sectors0
0000B9B1 3B05[E25A0100] <1>       cmp   eax, [FAT_BuffSector]
0000B9B7 7570           <1>       jne   short load_FAT_sectors1
0000B9B9 A1[DA5A0100]   <1>       mov   eax, [FAT_CurrentCluster]
0000B9BE 66D1E8         <1>       shr   ax, 1
                        <1>       ;movzx eax, word [ebx]
0000B9C1 668B03         <1>       mov   ax, [ebx]
0000B9C4 7314           <1>       jnc   short get_FAT12_nc_even
0000B9C6 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
0000B9CA 663DF70F       <1>       cmp   ax, 0FF7h
0000B9CE 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:
0000B9D0 31C0           <1>       xor   eax, eax ; 0
                        <1> loc_pass_gnc_FAT16_eoc_check:
                        <1> loc_pass_gnc_FAT32_eoc_check:
0000B9D2 8B0D[DA5A0100] <1>       mov   ecx, [FAT_CurrentCluster]
0000B9D8 F5             <1>       cmc
0000B9D9 C3             <1>       retn
                        <1>
                        <1> get_FAT12_nc_even:
0000B9DA 80E40F         <1>       and   ah, 0Fh
0000B9DD EBEB           <1>       jmp   short loc_gnc_fat12_eoc_check
                        <1>
                        <1> get_FAT32_next_cluster:
0000B9DF BB80010000     <1>       mov   ebx, 180h ;384
0000B9E4 F7F3           <1>       div   ebx
                        <1>       ; EAX = Count of 3 FAT sectors
                        <1>       ; EDX = Cluster Offset (< 384)
0000B9E6 66C1E202       <1>       shl   dx, 2  ; Multiply by 4
```

```
130 0000B9EA 89D3                <1>       mov    ebx, edx ; Byte Offset
131 0000B9EC 81C3001C0900        <1>       add    ebx, FAT_Buffer
132 0000B9F2 66BA0300            <1>       mov    dx, 3
133 0000B9F6 F7E2                <1>       mul    edx
134                              <1>         ; EAX = FAT Sector (<= 2097152) ; (FFFFFF7h * 4) / 512
135                              <1>       ;      for 32KB cluster size:
136                              <1>       ;      EAX <= 1024 = (4GB / 32KB) * 4) / 512
137                              <1>       ; EDX = 0
138 0000B9F8 8A0E                <1>       mov    cl, [esi+LD_Name]
139 0000B9FA 803D[DE5A0100]00    <1>       cmp    byte [FAT_BuffValidData], 0
140 0000BA01 7620                <1>       jna    short load_FAT_sectors0
141 0000BA03 3A0D[DF5A0100]      <1>       cmp    cl, [FAT_BuffDrvName]
142 0000BA09 7518                <1>       jne    short load_FAT_sectors0
143 0000BA0B 3B05[E25A0100]      <1>       cmp    eax, [FAT_BuffSector] ; 0, 3, 6, 9 ...
144 0000BA11 7516                <1>       jne    short load_FAT_sectors1
145 0000BA13 8B03                <1>       mov    eax, [ebx]
146 0000BA15 25FFFFFF0F          <1>       and    eax, 0FFFFFFFh ; 28 bit Cluster
147 0000BA1A 3DF7FFFF0F          <1>       cmp    eax, 0FFFFFF7h
148 0000BA1F 72B1                <1>       jb     short loc_pass_gnc_FAT32_eoc_check
149                              <1>       ; eax >= FFFFFF7h (cluster 0002h to FFFFFF6h is valid)
150 0000BA21 EBAD                <1>       jmp    short loc_pass_gnc_FAT16_eoc_check_xor_eax
151                              <1>
152                              <1> load_FAT_sectors0:
153 0000BA23 880D[DF5A0100]      <1>       mov    [FAT_BuffDrvName], cl
154                              <1> load_FAT_sectors1:
155 0000BA29 A3[E25A0100]        <1>       mov    [FAT_BuffSector], eax
156 0000BA2E 89C3                <1>       mov    ebx, eax
157 0000BA30 034660              <1>         add     eax, [esi+LD_FATBegin]
158 0000BA33 807E0302            <1>       cmp    byte [esi+LD_FATType], 2
159 0000BA37 7706                <1>         ja      short load_FAT_sectors3
160 0000BA39 0FB74E1C            <1>       movzx  ecx, word [esi+LD_BPB+BPB_FATSz16]
161 0000BA3D EB03                <1>       jmp    short load_FAT_sectors4
162                              <1> load_FAT_sectors3:
163 0000BA3F 8B4E2A              <1>       mov    ecx, [esi+LD_BPB+BPB_FATSz32]
164                              <1> load_FAT_sectors4:
165 0000BA42 29D9                <1>       sub    ecx, ebx ; [FAT_BuffSector]
166 0000BA44 83F903              <1>         cmp     ecx, 3
167 0000BA47 7605                <1>         jna     short load_FAT_sectors5
168 0000BA49 B903000000          <1>       mov    ecx, 3
169                              <1> load_FAT_sectors5:
170 0000BA4E BB001C0900          <1>       mov    ebx, FAT_Buffer
171 0000BA53 E85A370000          <1>       call   disk_read
172 0000BA58 730D                <1>       jnc    short load_FAT_sectors_ok
173                              <1>       ; 15/10/2016 (15h -> 17)
174                              <1>       ; 23/03/2016 (15h)
175 0000BA5A B811000000          <1>       mov    eax, 17 ; Drive not ready or read error
176 0000BA5F C605[DE5A0100]00    <1>       mov    byte [FAT_BuffValidData], 0
177 0000BA66 C3                  <1>       retn
178                              <1> load_FAT_sectors_ok:
179 0000BA67 C605[DE5A0100]01    <1>       mov    byte [FAT_BuffValidData], 1
180 0000BA6E A1[DA5A0100]        <1>       mov    eax, [FAT_CurrentCluster]
181 0000BA73 E9AAFEFFFF          <1>         jmp     check_next_cluster_fat_type
182                              <1>
183                              <1> load_FAT_root_directory:
184                              <1>       ; 23/10/2016
185                              <1>       ; 15/10/2016
186                              <1>       ; 07/02/2016
187                              <1>       ; 02/02/2016
188                              <1>       ; 01/02/2016 (TRDOS 386 =  TRDOS v2.0)
189                              <1>       ; 21/05/2011
190                              <1>       ; 22/08/2009
191                              <1>       ;
192                              <1>       ; INPUT ->
193                              <1>       ;      ESI = Logical DOS Drive Description Table
194                              <1>       ; OUTPUT ->
195                              <1>       ;      cf = 1 -> Root directory could not be loaded
196                              <1>       ;          EAX > 0 -> Error number
197                              <1>       ;      cf = 0 -> EAX = 0
198                              <1>       ;      ECX = Directory buffer size in sectors (CL)
199                              <1>       ;      EBX = Directory buffer address
200                              <1>       ;      NOTE: DirBuffer_Size is in bytes ! (word)
201                              <1>       ;
202                              <1>       ; (Modified registers: EAX, ECX, EBX, EDX)
203                              <1>
204                              <1>       ; NOTE: Only for FAT12 and FAT16 file systems !
205                              <1>       ; (FAT32 fs root dir must be loaded as sub directory)
206                              <1>
207 0000BA78 8A1E                <1>       mov    bl, [esi+LD_Name]
208 0000BA7A 8A7E03              <1>       mov    bh, [esi+LD_FATType]
209                              <1>
210                              <1>       ;mov   [DirBuff_DRV], bl
211                              <1>       ;mov   [DirBuff_FATType], bh
212 0000BA7D 66891D[EE5A0100]    <1>       mov    [DirBuff_DRV], bx
213                              <1>
214                              <1>       ;cmp   bh, 2
215                              <1>       ;ja    short load_FAT32_root_dir0 ; FAT32 root dir
216                              <1>
217                              <1> load_FAT_root_dir0: ; 23/10/2016
218 0000BA84 0FB75617            <1>       movzx  edx, word [esi+LD_BPB+RootDirEnts]
219                              <1>
220                              <1>       ;or    dx, dx ; 0 for FAT32 file systems
221                              <1>       ;jz    short load_FAT32_root_dir0 ; FAT32 root dir
222                              <1>
223 0000BA88 6681FA0002           <1>       cmp    dx, 512 ; Number of Root Dir Entries
224 0000BA8D 7414                <1>       je     short lrd_mov_ecx_32
225 0000BA8F 89D0                <1>       mov    eax, edx
226                              <1>       ; 23/10/2016
227 0000BA91 89C1                <1>       mov    ecx, eax
228 0000BA93 6683C10F            <1>       add    cx, 15 ; round up
229 0000BA97 66C1E904            <1>       shr    cx, 4  ; 16 entries per sector (512/32)
230                              <1>       ; ecx = Root directory size in sectors
231 0000BA9B 66C1E005            <1>       shl    ax, 5 ; Root directory size in bytes
```

```
0000BA9F 664A                <1>       dec    dx    ; Last entry number of root dir
                             <1>       ; cx = Dir Buffer sector count
0000BAA1 EB0B                <1>       jmp    short lrd_check_dir_buffer
                             <1>
                             <1> lrd_mov_ecx_32:
0000BAA3 B920000000          <1>       mov    ecx, 32
0000BAA8 664A                <1>       dec    dx ; 511
0000BAAA 66B80040            <1>       mov    ax, 32*512
                             <1>
                             <1> lrd_check_dir_buffer:
0000BAAE 29DB                <1>       sub    ebx, ebx ; 0
0000BAB0 881D[F05A0100]      <1>       mov    [DirBuff_ValidData], bl ; 0
0000BAB6 668915[F35A0100]    <1>       mov    [DirBuff_LastEntry], dx
0000BABD 891D[F55A0100]      <1>       mov    [DirBuff_Cluster], ebx ; 0
0000BAC3 66A3[F95A0100]      <1>       mov    [DirBuffer_Size], ax
                             <1>
0000BAC9 8B4664              <1>       mov    eax, [esi+LD_ROOTBegin]
                             <1> read_directory:
0000BACC BB00000800          <1>       mov    ebx, Directory_Buffer
0000BAD1 51                  <1>       push   ecx ; Directory buffer sector count
0000BAD2 53                  <1>       push   ebx
0000BAD3 E8DA360000          <1>       call   disk_read
0000BAD8 5B                  <1>       pop    ebx
0000BAD9 720B                <1>       jc     short load_DirBuff_error
                             <1>
                             <1> validate_DirBuff_and_return:
0000BADB 59                  <1>       pop    ecx ; Number of loaded sectors
0000BADC C605[F05A0100]01    <1>       mov    byte [DirBuff_ValidData], 1
0000BAE3 31C0                <1>       xor    eax, eax ; 0 = no error
0000BAE5 C3                  <1>       retn
                             <1>
                             <1> load_DirBuff_error:
0000BAE6 89C8                <1>       mov    eax, ecx ; remaining sectors
0000BAE8 59                  <1>       pop    ecx ; sector count
0000BAE9 29C1                <1>       sub    ecx, eax ; Number of loaded sectors
                             <1>       ; 15/10/2016 (15h -> 17)
0000BAEB B811000000          <1>       mov    eax, 17 ; DRV NOT READY OR READ ERROR !
0000BAF0 F9                  <1>       stc
0000BAF1 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>
0000BAF2 8A1E                <1>       mov    bl, [esi+LD_Name]
0000BAF4 8A7E03              <1>       mov    bh, [esi+LD_FATType]
                             <1>
                             <1>       ;mov   [DirBuff_DRV], bl
                             <1>       ;mov   [DirBuff_FATType], bh
0000BAF7 66891D[EE5A0100]    <1>       mov    [DirBuff_DRV], bx
                             <1>
                             <1> load_FAT32_root_dir0:
0000BAFE 8B4632              <1>       mov    eax, [esi+LD_BPB+FAT32_RootFClust]
0000BB01 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>
0000BB03 8A1E                <1>       mov    bl, [esi+LD_Name]
0000BB05 8A7E03              <1>       mov    bh, [esi+LD_FATType]
                             <1>
                             <1>       ;mov   [DirBuff_DRV], bl
                             <1>       ;mov   [DirBuff_FATType], bh
0000BB08 66891D[EE5A0100]    <1>       mov    [DirBuff_DRV], bx
                             <1>
                             <1> load_FAT_sub_dir0:
0000BB0F 0FB64E13            <1>       movzx  ecx, byte [esi+LD_BPB+SecPerClust]
                             <1>
0000BB13 882D[F05A0100]      <1>       mov    [DirBuff_ValidData], ch ; 0
0000BB19 A3[F55A0100]        <1>       mov    [DirBuff_Cluster], eax
                             <1>
0000BB1E 0FB74611            <1>       movzx  eax, word [esi+LD_BPB+BytesPerSec]
0000BB22 F7E1                <1>       mul    ecx
0000BB24 C1E805              <1>       shr    eax, 5 ; directory entry count (dir size / 32)
```

```
334 0000BB27 6648               <1>       dec    ax ; last entry
335 0000BB29 66A3[F35A0100]     <1>       mov    [DirBuff_LastEntry], ax
336                             <1>
337 0000BB2F A1[F55A0100]       <1>       mov    eax, [DirBuff_Cluster]
338 0000BB34 83E802             <1>       sub    eax, 2
339 0000BB37 F7E1               <1>       mul    ecx
340 0000BB39 034668             <1>       add    eax, [esi+LD_DATABegin]
341                             <1>       ; ecx = sector per cluster (dir buffer size = 32 sectors)
342 0000BB3C EB8E               <1>       jmp    short read_directory
343                             <1>
344                             <1> ; DRV_FS.ASM
345                             <1>
346                             <1> load_current_FS_directory:
347 0000BB3E C3                 <1>       retn
348                             <1> load_FS_root_directory:
349 0000BB3F C3                 <1>       retn
350                             <1> load_FS_sub_directory:
351 0000BB40 C3                 <1>       retn
352                             <1>
353                             <1> read_cluster:
354                             <1>       ; 15/10/2016
355                             <1>       ; 18/03/2016
356                             <1>       ; 16/03/2016
357                             <1>       ; 17/02/2016
358                             <1>       ; 15/02/2016 (TRDOS 386 =  TRDOS v2.0)
359                             <1>       ;
360                             <1>       ; INPUT ->
361                             <1>       ;     EAX = Cluster Number (Sector index for SINGLIX FS)
362                             <1>       ;     ESI = Logical DOS Drive Description Table address
363                             <1>       ;     EBX = Cluster (File R/W) Buffer address (max. 64KB)
364                             <1>       ;     Only for SINGLIX FS:
365                             <1>       ;     EDX = File Number (The 1st FDT address)
366                             <1>       ; OUTPUT ->
367                             <1>       ;     cf = 1 -> Cluster can not be loaded at the buffer
368                             <1>       ;         EAX > 0 -> Error number
369                             <1>       ;     cf = 0 -> Cluster has been loaded at the buffer
370                             <1>       ;
371                             <1>       ; (Modified registers: EAX, ECX, EBX, EDX)
372                             <1>
373 0000BB41 0FB64E13           <1>       movzx  ecx, byte [esi+LD_BPB+BPB_SecPerClust]
374                             <1>       ; CL = 1 = [esi+LD_FS_Reserved2] ; SectPerClust for Singlix FS
375                             <1>
376                             <1> read_file_sectors: ; 16/03/2016
377 0000BB45 807E0300           <1>       cmp    byte [esi+LD_FATType], 0
378 0000BB49 761C               <1>       jna    short read_fs_cluster
379                             <1>
380                             <1> read_fat_file_sectors: ; 18/03/2016
381 0000BB4B 83E802             <1>       sub    eax, 2 ; Beginning cluster number is always 2
382 0000BB4E 0FB65613           <1>       movzx  edx, byte [esi+LD_BPB+BPB_SecPerClust] ; 18/03/2016
383 0000BB52 F7E2               <1>       mul    edx
384 0000BB54 034668             <1>       add    eax, [esi+LD_DATABegin] ; absolute address of the cluster
385                             <1>
386                             <1>       ; EAX = Disk sector address
387                             <1>       ; ECX = Sector count
388                             <1>       ; EBX = Buffer address
389                             <1>       ; (EDX = 0)
390                             <1>       ; ESI = Logical DOS drive description table address
391                             <1>
392 0000BB57 E856360000         <1>       call   disk_read
393 0000BB5C 7306               <1>       jnc    short rclust_retn
394                             <1>
395                             <1>       ; 15/10/2016 (15h -> 17)
396 0000BB5E B811000000         <1>       mov    eax, 17 ; Drive not ready or read error !
397 0000BB63 C3                 <1>       retn
398                             <1>
399                             <1> rclust_retn:
400 0000BB64 29C0               <1>       sub    eax, eax ; 0
401 0000BB66 C3                 <1>       retn
402                             <1>
403                             <1> read_fs_cluster:
404                             <1>       ; 15/02/2016 (TRDOS 386 =  TRDOS v2.0)
405                             <1>       ; Singlix FS
406                             <1>
407                             <1>       ; EAX = Cluster number is sector index number of the file (eax)
408                             <1>
409                             <1>       ; EDX = File number is the first File Descriptor Table address
410                             <1>       ;     of the file. (Absolute address of the FDT).
411                             <1>
412                             <1>       ; eax = sector index (0 for the first sector)
413                             <1>       ; edx = FDT0 address
414                             <1>             ; 64 KB buffer = 128 sectors (limit)
415 0000BB67 B980000000         <1>       mov    ecx, 128 ; maximum count of sectors (before eof)
416 0000BB6C E801000000         <1>       call   read_fs_sectors
417 0000BB71 C3                 <1>       retn
418                             <1>
419                             <1> read_fs_sectors:
420                             <1>       ; 15/02/2016 (TRDOS 386 =  TRDOS v2.0)
421 0000BB72 F9                 <1>       stc
422 0000BB73 C3                 <1>       retn
423                             <1>
424                             <1> get_first_free_cluster:
425                             <1>       ; 02/03/2016
426                             <1>       ; 21/02/2016 (TRDOS 386 =  TRDOS v2.0)
427                             <1>       ; 26/10/2010 (DRV_FAT.ASM, 'proc_get_first_free_cluster')
428                             <1>       ; 10/07/2010
429                             <1>       ; INPUT ->
430                             <1>       ;     ESI = Logical DOS Drive Description Table address
431                             <1>       ; OUTPUT ->
432                             <1>       ;     cf = 1 -> Error code in AL (EAX)
433                             <1>       ;     cf = 0 ->
434                             <1>       ;       EAX = Cluster number
435                             <1>       ;       If EAX = FFFFFFFFh -> no free space
```

```
                        <1>       ;     If the drive has FAT32 fs:
                        <1>       ;       EBX = FAT32 FSI sector buffer address (if > 0)
                        <1>
0000BB74 8B4678         <1>       mov   eax, [esi+LD_Clusters]
0000BB77 40             <1>       inc   eax ; add eax, 1
0000BB78 A3[785D0100]   <1>       mov   [gffc_last_free_cluster], eax
                        <1>
0000BB7D 31DB           <1>       xor   ebx, ebx ; 0 ; 02/03/2016
                        <1>
0000BB7F 807E0302       <1>       cmp   byte [esi+LD_FATType], 2
0000BB83 760E           <1>       jna   short loc_gffc_get_first_fat_free_cluster0
                        <1>
                        <1> loc_gffc_get_first_fat32_free_cluster:
                        <1>       ; 02/03/2016
0000BB85 E844060000     <1>       call  get_fat32_fsinfo_sector_parms
0000BB8A 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)
0000BB8C 89D0           <1>       mov   eax, edx ; FSI_Next_Free (First Free Cluster)
0000BB8E 83F8FF         <1>       cmp   eax, 0FFFFFFFFh ; invalid (unknown) !
0000BB91 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:
0000BB93 B802000000     <1>       mov   eax, 2
                        <1>       ;xor   edx, edx
                        <1>
                        <1> loc_gffc_get_first_fat_free_cluster1:
0000BB98 53             <1>       push  ebx ; 02/03/2016
                        <1>
                        <1> loc_gffc_get_first_fat_free_cluster2:
0000BB99 A3[745D0100]   <1>       mov   [gffc_first_free_cluster], eax
0000BB9E A3[705D0100]   <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:
0000BBA3 E875FDFFFF     <1>       call  get_next_cluster
0000BBA8 7307           <1>       jnc   short loc_gffc_get_first_fat_free_cluster4
0000BBAA 09C0           <1>       or    eax, eax
0000BBAC 740B           <1>       jz    short loc_gffc_first_free_fat_cluster_next
0000BBAE 5B             <1>       pop   ebx ; 02/03/2016
0000BBAF F5             <1>       cmc   ; stc
0000BBB0 C3             <1>       retn
                        <1>
                        <1> loc_gffc_get_first_fat_free_cluster4:
0000BBB1 21C0           <1>       and   eax, eax ; next cluster value
0000BBB3 7504           <1>       jnz   short loc_gffc_first_free_fat_cluster_next
0000BBB5 89C8           <1>       mov   eax, ecx ; current (previous cluster) value
0000BBB7 EB22           <1>       jmp   short loc_gffc_check_for_set
                        <1>
                        <1> loc_gffc_first_free_fat_cluster_next:
0000BBB9 A1[705D0100]   <1>       mov   eax, [gffc_next_free_cluster]
0000BBBE 3B05[785D0100] <1>       cmp   eax, [gffc_last_free_cluster]
0000BBC4 7308           <1>       jnb   short retn_stc_from_get_first_free_cluster
                        <1> pass_gffc_last_cluster_eax_check:
0000BBC6 40             <1>       inc   eax ; add eax, 1
0000BBC7 A3[705D0100]   <1>       mov   [gffc_next_free_cluster], eax
0000BBCC EBD5           <1>       jmp   short loc_gffc_get_first_fat_free_cluster3
                        <1>
                        <1> retn_stc_from_get_first_free_cluster:
0000BBCE A1[745D0100]   <1>       mov   eax, [gffc_first_free_cluster]
0000BBD3 83F802         <1>       cmp   eax, 2
0000BBD6 7709           <1>       ja    short loc_gffc_check_previous_clusters
0000BBD8 29C0           <1>       sub   eax, eax
0000BBDA 48             <1>       dec   eax ; FFFFFFFFh
                        <1>
                        <1> loc_gffc_check_for_set:
                        <1>       ; 02/03/2016
0000BBDB 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>
0000BBDC 09DB           <1>       or    ebx, ebx
0000BBDE 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:
0000BBE0 C3             <1>       retn
                        <1>
                        <1> loc_gffc_check_previous_clusters:
0000BBE1 48             <1>       dec   eax ; sub eax, 1
0000BBE2 A3[785D0100]   <1>       mov   [gffc_last_free_cluster], eax
0000BBE7 B802000000     <1>       mov   eax, 2
                        <1>       ;xor   edx, edx
0000BBEC 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'
0000BBEE 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>
0000BBF1 813B52526141           <1>       cmp     dword [ebx], 41615252h
0000BBF7 7540                   <1>       jne    short loc_sffc_read_fsinfo_sector
0000BBF9 81BBE4010000727241-    <1>       cmp    dword [ebx+484], 61417272h
0000BC02 61                     <1>
0000BC03 7534                   <1>       jne    short loc_sffc_read_fsinfo_sector
                                <1>
0000BC05 3B83EC010000           <1>       cmp    eax, [ebx+492]  ; FSI_Next_Free
0000BC0B 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'
0000BC0D 8983EC010000           <1>       mov    [ebx+492], eax
0000BC13 A1[885D0100]           <1>       mov    eax, [CFS_FAT32FSINFOSEC]
0000BC18 B901000000             <1>       mov    ecx, 1
0000BC1D 53                     <1>       push   ebx
0000BC1E E880350000             <1>       call   disk_write
0000BC23 7208                   <1>       jc      short loc_sffc_read_fsinfo_sector_err1
0000BC25 5B                     <1>       pop    ebx
                                <1>
0000BC26 8B83EC010000           <1>       mov    eax, [ebx+492] ; First (Next) Free Cluster
                                <1>
                                <1> loc_sffc_retn:
0000BC2C 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:
0000BC2D BB00000000             <1>       mov    ebx, 0
                                <1>       ; 15/10/2016 (1Dh -> 18)
                                <1>       ; 23/03/2016 (1Dh)
0000BC32 B812000000             <1>       mov    eax, 18 ; Drive not ready or write error
                                <1>
                                <1> loc_sffc_read_fsinfo_sector_err2:
0000BC37 5A                     <1>       pop    edx
0000BC38 C3                     <1>       retn
                                <1>
                                <1> loc_sffc_read_fsinfo_sector:
0000BC39 50                     <1>       push   eax
                                <1>
0000BC3A E88F050000             <1>       call   get_fat32_fsinfo_sector_parms
0000BC3F 72F6                   <1>       jc     short loc_sffc_read_fsinfo_sector_err2
                                <1>
0000BC41 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)
0000BC42 39D0                   <1>       cmp    eax, edx ; First free Cluster (eax = new value)
0000BC44 75C7                   <1>       jne    short loc_sffc_write_fsinfo_sector
                                <1>
0000BC46 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>
0000BC47 A3[DA5A0100]           <1>       mov   [FAT_CurrentCluster], eax
0000BC4C 890D[7C5D0100]         <1>       mov   [ClusterValue], ecx
                                <1>
                                <1> loc_update_cluster_check_fat_buffer:
0000BC52 8A1E                   <1>       mov   bl, [esi+LD_Name]
0000BC54 381D[DF5A0100]         <1>       cmp   [FAT_BuffDrvName], bl
0000BC5A 741A                   <1>       je    short loc_update_cluster_check_fat_type
0000BC5C 803D[DE5A0100]02       <1>       cmp   byte [FAT_BuffValidData], 2
0000BC63 0F84C2000000           <1>         je      loc_uc_save_fat_buffer
                                <1>
                                <1> loc_uc_reset_fat_buffer_validation:
0000BC69 C605[DE5A0100]00       <1>       mov   byte [FAT_BuffValidData], 0
                                <1>
                                <1> loc_uc_check_fat_type_reset_drvname:
0000BC70 881D[DF5A0100]         <1>       mov   [FAT_BuffDrvName], bl
                                <1>
                                <1> loc_update_cluster_check_fat_type:
0000BC76 29D2                   <1>       sub   edx, edx ; 26/02/2016
0000BC78 8A5E03                 <1>       mov   bl, [esi+LD_FATType]
0000BC7B 83F802                 <1>       cmp   eax, 2
0000BC7E 0F82BE000000           <1>         jb      update_cluster_inv_data
0000BC84 80FB02                 <1>       cmp   bl, 2
0000BC87 0F877A010000           <1>         ja      update_fat32_cluster
                                <1>       ;cmp  bl, 1
                                <1>       ;jb   short update_cluster_inv_data
0000BC8D 8B4E78                 <1>       mov   ecx, [esi+LD_Clusters]
0000BC90 41                     <1>       inc   ecx
0000BC91 890D[EA5A0100]         <1>       mov   [LastCluster], ecx
0000BC97 39C8                   <1>       cmp   eax, ecx ; dword [LastCluster]
0000BC99 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:')
0000BC9F 80FB01                 <1>       cmp   bl, 1 ; correct comparison is this !
0000BCA2 0F86A2000000           <1>         jna     update_fat12_cluster
                                <1>
                                <1> update_fat16_cluster:
                                <1> pass_uc_fat16_errc:
                                <1>       ;sub   edx, edx
0000BCA8 BB00030000             <1>       mov   ebx, 300h ;768
0000BCAD F7F3                   <1>       div   ebx
                                <1>       ; EAX = Count of 3 FAT sectors
                                <1>       ; DX = Cluster offset in FAT buffer
0000BCAF 6689D3                 <1>       mov   bx, dx
0000BCB2 66D1E3                 <1>       shl   bx, 1 ; Multiply by 2
0000BCB5 66BA0300               <1>       mov   dx, 3
0000BCB9 F7E2                   <1>       mul   edx
                                <1>       ; EAX = FAT Sector
                                <1>       ; EDX = 0
                                <1>       ; EBX = Byte offset in FAT buffer
0000BCBB 8A0D[DE5A0100]         <1>       mov   cl, [FAT_BuffValidData]
0000BCC1 80F902                 <1>       cmp   cl, 2
0000BCC4 750A                   <1>       jne   short loc_uc_check_fat16_buff_sector_load
                                <1>
                                <1> loc_uc_check_fat16_buff_sector_save:
0000BCC6 3B05[E25A0100]         <1>       cmp   eax, [FAT_BuffSector]
0000BCCC 755D                   <1>       jne   short loc_uc_save_fat_buffer
0000BCCE EB15                   <1>       jmp   short loc_update_fat16_cell
                                <1>
                                <1> loc_uc_check_fat16_buff_sector_load:
0000BCD0 80F901                 <1>       cmp   cl, 1 ; byte [FAT_BuffValidData]
0000BCD3 0F85FB010000           <1>         jne     loc_uc_load_fat_sectors
0000BCD9 3B05[E25A0100]         <1>       cmp   eax, [FAT_BuffSector]
0000BCDF 0F85EF010000           <1>         jne     loc_uc_load_fat_sectors
                                <1>
                                <1> loc_update_fat16_cell:
                                <1> loc_update_fat16_buffer:
0000BCE5 81C3001C0900           <1>       add   ebx, FAT_Buffer ; 26/02/2016
                                <1>       ;movzx eax, word [ebx]
0000BCEB 668B03                 <1>       mov   ax, [ebx]
                                <1>       ; 01/03/2016
0000BCEE 89C2                   <1>       mov   edx, eax ; old value of the cluster
0000BCF0 A3[DA5A0100]           <1>       mov   [FAT_CurrentCluster], eax
0000BCF5 8B0D[7C5D0100]         <1>       mov   ecx, [ClusterValue] ; 32 bits
0000BCFB 66890B                 <1>       mov   [ebx], cx ; 16 bits !
                                <1>
```

```
0000BCFE C605[DE5A0100]02    <1>       mov    byte [FAT_BuffValidData], 2
                             <1>
0000BD05 6683F802            <1>       cmp    ax, 2
0000BD09 723A                <1>       jb     short return_uc_fat_stc
0000BD0B 3B05[EA5A0100]      <1>       cmp    eax, [LastCluster]
0000BD11 7732                <1>       ja     short return_uc_fat_stc
                             <1>
                             <1> loc_fat_buffer_updated:
                             <1>       ; 01/03/2016
0000BD13 F8                  <1>       clc
                             <1> loc_fat_buffer_stc_1:
0000BD14 9C                  <1>       pushf
0000BD15 21C9                <1>       and    ecx, ecx
0000BD17 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
0000BD19 FF05[E65A0100]      <1>       inc    dword [FAT_ClusterCounter]
                             <1>
                             <1> loc_fat_buffer_updated_1: ; new value of the cluster > 0
0000BD1F 09D2                <1>       or     edx, edx ; 02/03/2016
0000BD21 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
0000BD23 FF0D[E65A0100]      <1>       dec    dword [FAT_ClusterCounter] ; it may be negative number
                             <1>
                             <1> loc_fat_buffer_updated_2:
0000BD29 9D                  <1>       popf
0000BD2A C3                  <1>       retn
                             <1>
                             <1> loc_uc_save_fat_buffer:
                             <1>       ; byte [FAT_BuffValidData] = 2
0000BD2B E8D4010000          <1>       call   save_fat_buffer
0000BD30 0F8297010000        <1>         jc       loc_fat_sectors_rw_error2
                             <1>       ;mov   byte [FAT_BuffValidData], 1
0000BD36 A1[DA5A0100]        <1>       mov    eax, [FAT_CurrentCluster]
                             <1>       ;mov   ecx, [ClusterValue]
                             <1>       ;jmp   short loc_update_cluster_check_fat_buffer
0000BD3B 8A1E                <1>       mov    bl, [esi+LD_Name] ; 01/03/2016
0000BD3D E927FFFFFF          <1>         jmp      loc_uc_reset_fat_buffer_validation
                             <1>
                             <1> update_cluster_inv_data:
                             <1>       ;mov   eax, 0Dh
0000BD42 B00D                <1>       mov    al, 0Dh  ; Invalid Data
0000BD44 C3                  <1>       retn
                             <1>
                             <1> return_uc_fat_stc:
                             <1>       ; 01/03/2016
0000BD45 31C0                <1>       xor    eax, eax
0000BD47 F9                  <1>       stc
0000BD48 EBCA                <1>       jmp    short loc_fat_buffer_stc_1
                             <1>
                             <1> update_fat12_cluster:
                             <1> pass_uc_fat12_errc:
                             <1>       ;sub   edx, edx
0000BD4A BB00040000          <1>       mov    ebx, 400h ;1024
0000BD4F F7F3                <1>       div    ebx
                             <1>       ; EAX = Count of 3 FAT sectors
                             <1>       ; DX = Cluster offset in FAT buffer
0000BD51 66B90300            <1>       mov    cx, 3
0000BD55 6689C3              <1>       mov    bx, ax
0000BD58 6689C8              <1>       mov    ax, cx ; 3
0000BD5B 66F7E2              <1>       mul    dx     ; Multiply by 3
0000BD5E 66D1E8              <1>       shr    ax, 1  ; Divide by 2
0000BD61 6693                <1>       xchg   bx, ax
                             <1>       ; EAX = Count of 3 FAT sectors
                             <1>       ; EBX = Byte Offset in FAT buffer
0000BD63 66F7E1              <1>       mul    cx  ; 3 * AX
                             <1>       ; EAX = FAT Beginning Sector
                             <1>       ; EDX = 0
0000BD66 8A0D[DE5A0100]      <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 !)
0000BD6C 80F902              <1>       cmp    cl, 2 ; 2 = dirty buffer (must be written to disk)
0000BD6F 750A                <1>       jne    short loc_uc_check_fat12_buff_sector_load
                             <1>
                             <1> loc_uc_check_fat12_buff_sector_save:
0000BD71 3B05[E25A0100]      <1>       cmp    eax, [FAT_BuffSector]
0000BD77 75B2                <1>         jne      short loc_uc_save_fat_buffer
0000BD79 EB15                <1>       jmp    short loc_update_fat12_cell
                             <1>
                             <1> loc_uc_check_fat12_buff_sector_load:
0000BD7B 80F901              <1>       cmp    cl, 1 ; byte ptr [FAT_BuffValidData]
0000BD7E 0F8550010000        <1>         jne      loc_uc_load_fat_sectors
0000BD84 3B05[E25A0100]      <1>       cmp    eax, [FAT_BuffSector]
0000BD8A 0F8544010000        <1>         jne      loc_uc_load_fat_sectors
                             <1>
                             <1> loc_update_fat12_cell:
0000BD90 81C3001C0900        <1>       add    ebx, FAT_Buffer ; 26/02/2016
0000BD96 668B0D[DA5A0100]    <1>       mov    cx, [FAT_CurrentCluster]
0000BD9D 66D1E9              <1>       shr    cx, 1
0000BDA0 668B03              <1>       mov    ax, [ebx]
0000BDA3 6689C2              <1>       mov    dx, ax
0000BDA6 7344                <1>       jnc    short uc_fat12_nc_even
                             <1>
0000BDA8 6683E00F            <1>       and    ax, 0Fh
0000BDAC 8B0D[7C5D0100]      <1>       mov    ecx, [ClusterValue] ; 32 bits
0000BDB2 66C1E104            <1>       shl    cx, 4
0000BDB6 6609C1              <1>       or     cx, ax
0000BDB9 6689D0              <1>       mov    ax, dx
```

```
0000BDBC 66890B              <1>       mov    [ebx], cx  ; 16 bits !
0000BDBF 66C1E804            <1>       shr    ax, 4 ; al(bit4..7)+ah(bit0..7)
                             <1>
                             <1> update_fat12_buffer:
0000BDC3 A3[DA5A0100]        <1>       mov    [FAT_CurrentCluster], eax
0000BDC8 89C2                <1>       mov    edx, eax ; 01/03/2016
0000BDCA C605[DE5A0100]02    <1>       mov    byte [FAT_BuffValidData], 2
0000BDD1 6683F802            <1>       cmp    ax, 2
0000BDD5 0F826AFFFFFF        <1>         jb     return_uc_fat_stc
0000BDDB 3B05[EA5A0100]      <1>       cmp    eax, [LastCluster]
0000BDE1 0F875EFFFFFF        <1>         ja     return_uc_fat_stc
0000BDE7 E927FFFFFF          <1>         jmp    loc_fat_buffer_updated
                             <1>
                             <1> uc_fat12_nc_even:
0000BDEC 662500F0            <1>       and    ax, 0F000h
0000BDF0 8B0D[7C5D0100]      <1>       mov    ecx, [ClusterValue] ; 32 bits
0000BDF6 80E50F              <1>       and    ch, 0Fh
0000BDF9 6609C1              <1>       or     cx, ax
0000BDFC 6689D0              <1>       mov    ax, dx
0000BDFF 66890B              <1>       mov    [ebx], cx ; 16 bits !
0000BE02 80E40F              <1>       and    ah, 0Fh ; al(bit0..7)+ah(bit0..3)
0000BE05 EBBC                <1>       jmp    short update_fat12_buffer
                             <1>
                             <1> update_fat32_cluster:
0000BE07 8B4E78              <1>       mov    ecx, [esi+LD_Clusters]
0000BE0A 41                  <1>       inc    ecx
0000BE0B 890D[EA5A0100]      <1>       mov    [LastCluster], ecx
                             <1>
0000BE11 39C8                <1>       cmp    eax, ecx
0000BE13 0F872CFFFFFF        <1>         ja     return_uc_fat_stc
                             <1>
                             <1> pass_uc_fat32_errc:
                             <1>       ;sub   edx, edx
0000BE19 BB80010000          <1>       mov    ebx, 180h ;384
0000BE1E F7F3                <1>       div    ebx
                             <1>       ; EAX = Count of 3 FAT sectors
                             <1>       ; DX = Cluster offset in FAT buffer
0000BE20 89D3                <1>       mov    ebx, edx
0000BE22 C1E302              <1>       shl    ebx, 2 ; Multiply by 4
0000BE25 BA03000000          <1>       mov    edx, 3
0000BE2A F7E2                <1>       mul    edx
                             <1>       ; EBX = Cluster Offset in FAT buffer
                             <1>       ; EAX = FAT Sector
                             <1>       ; EDX = 0
0000BE2C 8A0D[DE5A0100]      <1>       mov    cl, [FAT_BuffValidData]
0000BE32 80F902              <1>       cmp    cl, 2
0000BE35 750E                <1>       jne    short loc_uc_check_fat32_buff_sector_load
                             <1>
                             <1> loc_uc_check_fat32_buff_sector_save:
0000BE37 3B05[E25A0100]      <1>       cmp    eax, [FAT_BuffSector]
0000BE3D 0F85E8FEFFFF        <1>         jne    loc_uc_save_fat_buffer
0000BE43 EB11                <1>       jmp    short loc_update_fat32_cell
                             <1>
                             <1> loc_uc_check_fat32_buff_sector_load:
0000BE45 80F901              <1>       cmp    cl, 1 ; byte [FAT_BuffValidData]
0000BE48 0F8586000000        <1>         jne    loc_uc_load_fat_sectors
0000BE4E 3B05[E25A0100]      <1>       cmp    eax, [FAT_BuffSector]
0000BE54 757E                <1>         jne    loc_uc_load_fat_sectors
                             <1>
                             <1> loc_update_fat32_cell:
                             <1> loc_update_fat32_buffer:
0000BE56 81C3001C0900        <1>       add    ebx, FAT_Buffer ; 26/02/2016
0000BE5C 8B03                <1>       mov    eax, [ebx]
0000BE5E 25FFFFFF0F          <1>       and    eax, 0FFFFFFFh ; 28 bit cluster value
                             <1>
0000BE63 8B15[DA5A0100]      <1>       mov    edx, [FAT_CurrentCluster] ; 01/03/2016
                             <1>
0000BE69 A3[DA5A0100]        <1>       mov    [FAT_CurrentCluster], eax
0000BE6E 8B0D[7C5D0100]      <1>       mov    ecx, [ClusterValue]
0000BE74 890B                <1>       mov    [ebx], ecx ; 29/02/2016
                             <1>
0000BE76 C605[DE5A0100]02    <1>       mov    byte [FAT_BuffValidData], 2
                             <1>
                             <1>       ; 01/03/2016
0000BE7D 21C0                <1>       and    eax, eax ; was it free cluster ?
0000BE7F 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>
0000BE81 3B563E              <1>       cmp    edx, [esi+LD_BPB+BPB_Reserved+4] ; First free cluster
0000BE84 7520                <1>       jne    short loc_upd_fat32_c3
                             <1>
0000BE86 3B15[EA5A0100]      <1>       cmp    edx, [LastCluster]
0000BE8C 7207                <1>       jb     short loc_upd_fat32_c0
                             <1>
0000BE8E BA02000000          <1>       mov    edx, 2 ; rewind !
0000BE93 EB0E                <1>       jmp    short loc_upd_fat32_c2
                             <1>
                             <1> loc_upd_fat32_c0:
0000BE95 FF463E              <1>       inc    dword [esi+LD_BPB+BPB_Reserved+4] ; set it to next cluster
0000BE98 EB0C                <1>       jmp    short loc_upd_fat32_c3
                             <1>
                             <1> loc_upd_fat32_c1:
0000BE9A 09C9                <1>       or     ecx, ecx ; will it be free cluster ?
0000BE9C 7508                <1>       jnz    short loc_upd_fat32_c3
                             <1>
0000BE9E 3B563E              <1>       cmp    edx, [esi+LD_BPB+BPB_Reserved+4] ; First free cluster
0000BEA1 7303                <1>       jnb    short loc_upd_fat32_c3
                             <1>
                             <1> loc_upd_fat32_c2:
0000BEA3 89563E              <1>       mov    [esi+LD_BPB+BPB_Reserved+4], edx
```

```
                                <1>
                                <1> loc_upd_fat32_c3:
0000BEA6 89C2                   <1>       mov    edx, eax
                                <1>
                                <1> loc_upd_fat32_c4:
0000BEA8 83F802                 <1>       cmp    eax, 2
0000BEAB 0F8294FEFFFF           <1>         jb      return_uc_fat_stc
                                <1>
                                <1> pass_uc_fat32_c_zero_check_2:
0000BEB1 3B05[EA5A0100]         <1>       cmp    eax, [LastCluster]
0000BEB7 0F8788FEFFFF           <1>         ja      return_uc_fat_stc
                                <1>
0000BEBD 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
0000BEC2 B811000000             <1>       mov    eax, 17 ; Drive not ready or read error
0000BEC7 8825[DE5A0100]         <1>       mov    [FAT_BuffValidData], ah ; 0
                                <1>
                                <1> loc_fat_sectors_rw_error2:
                                <1>       ;mov   eax, error code
                                <1>       ;mov   edx, 0
0000BECD 8B0D[7C5D0100]         <1>       mov    ecx, [ClusterValue]
0000BED3 C3                     <1>       retn
                                <1>
                                <1> loc_uc_load_fat_sectors:
0000BED4 A3[E25A0100]           <1>       mov    [FAT_BuffSector], eax
                                <1>
                                <1> load_uc_fat_sectors_zero:
0000BED9 034660                 <1>       add    eax, [esi+LD_FATBegin]
0000BEDC BB001C0900             <1>       mov    ebx, FAT_Buffer
0000BEE1 B903000000             <1>       mov    ecx, 3
0000BEE6 E8C7320000             <1>       call   disk_read
0000BEEB 72D5                   <1>       jc     short loc_fat_sectors_rw_error1
                                <1>
0000BEED C605[DE5A0100]01       <1>         mov     byte [FAT_BuffValidData], 1
0000BEF4 A1[DA5A0100]           <1>       mov    eax, [FAT_CurrentCluster]
0000BEF9 8B0D[7C5D0100]         <1>       mov    ecx, [ClusterValue]
0000BEFF E972FDFFFF             <1>         jmp     loc_update_cluster_check_fat_type
                                <1>
                                <1> save_fat_buffer:
                                <1>       ; 15/10/2016
                                <1>       ; 01/03/2016
                                <1>       ; 22/02/2016 (TRDOS 386 =  TRDOS v2.0)
                                <1>       ; 11/08/2011
                                <1>       ; 09/02/2005
                                <1>       ; INPUT ->
                                <1>       ;      None
                                <1>       ; OUTPUT ->
                                <1>       ;      cf = 0 -> OK.
                                <1>       ;      cf = 1 -> error code in AL (EAX)
                                <1>       ;
                                <1>       ;      EBX = FAT_Buffer address
                                <1>       ;
                                <1>       ; (EAX, EDX, ECX will be modified)
                                <1>
                                <1>       ;cmp   byte [FAT_BuffValidData], 2
                                <1>       ;je    short loc_save_fat_buff
                                <1>
                                <1> ;loc_save_fat_buffer_retn:
                                <1> ;     xor    eax, eax
                                <1> ;     retn
                                <1>
                                <1> loc_save_fat_buff:
0000BF04 31D2                   <1>       xor    edx, edx
0000BF06 8A35[DF5A0100]         <1>       mov    dh, [FAT_BuffDrvName]
0000BF0C 80FE41                 <1>       cmp    dh, 'A'
0000BF0F 722E                   <1>       jb     short loc_save_fat_buffer_inv_data_retn
0000BF11 80EE41                 <1>       sub    dh, 'A'
0000BF14 56                     <1>       push   esi ; *
0000BF15 BE00010900             <1>         mov     esi, Logical_DOSDisks
0000BF1A 01D6                   <1>       add    esi, edx
                                <1>
0000BF1C 8A5603                 <1>       mov    dl, [esi+LD_FATType]
0000BF1F 20D2                   <1>       and    dl, dl
0000BF21 741B                   <1>       jz     short loc_save_fat_buffer_inv_data_pop_retn
                                <1>
0000BF23 A1[E25A0100]           <1>       mov    eax, [FAT_BuffSector]
0000BF28 80FA02                 <1>       cmp    dl, 2
0000BF2B 770A                   <1>       ja     short loc_save_fat32_buff
                                <1>
                                <1> loc_save_fat_12_16_buff:
                                <1>       ; 01/03/2016
                                <1>       ; TRDOS v1 has a FATal bug here!
                                <1>       ; Correct code: mov dx, word ptr [FAT_BuffSector]+2
                                <1>       ; (DX:AX in TRDOS v1 -> EAX in TRDOS v2)
                                <1>       ;
0000BF2D 0FB74E1C               <1>       movzx  ecx, word [esi+LD_BPB+FATSecs]
0000BF31 29C1                   <1>       sub    ecx, eax
                                <1>       ; TRDOS v1 has a bug here... ('pop esi' was forgotten!)
                                <1>       ;jna   short loc_save_fat_buffer_inv_data_retn ; wrong addr!
0000BF33 7609                   <1>       jna    short loc_save_fat_buffer_inv_data_pop_retn ; correct addr.
0000BF35 EB15                   <1>       jmp    short loc_save_fat_buffer_check_rs3
                                <1>
                                <1> loc_save_fat32_buff:
0000BF37 8B4E2A                 <1>       mov    ecx, [esi+LD_BPB+FAT32_FAT_Size]
0000BF3A 29C1                   <1>       sub    ecx, eax
0000BF3C 770E                   <1>       ja     short loc_save_fat_buffer_check_rs3
                                <1>
                                <1> loc_save_fat_buffer_inv_data_pop_retn:
```

```
0000BF3E 5E                <1>       pop    esi ; *
                           <1> loc_save_fat_buffer_inv_data_retn:
0000BF3F B80D000000        <1>       mov    eax, 0Dh ; Invalid DATA
0000BF44 C3                <1>       retn
                           <1>
                           <1> loc_save_fat_buff_remain_sectors_3:
0000BF45 B903000000        <1>       mov    ecx, 3
0000BF4A EB05              <1>       jmp    short loc_save_fat_buff_continue
                           <1>
                           <1> loc_save_fat_buffer_check_rs3:
0000BF4C 83F903            <1>       cmp    ecx, 3
0000BF4F 77F4              <1>       ja     short loc_save_fat_buff_remain_sectors_3
                           <1>
                           <1> loc_save_fat_buff_continue:
0000BF51 BB001C0900        <1>       mov    ebx, FAT_Buffer
0000BF56 034660            <1>       add    eax, [esi+LD_FATBegin]
0000BF59 51                <1>       push   ecx
0000BF5A E844320000        <1>       call   disk_write
0000BF5F 59                <1>       pop    ecx
0000BF60 722B              <1>       jc     short loc_save_FAT_buff_write_err
                           <1>
0000BF62 807E0302          <1>       cmp    byte [esi+LD_FATType], 2
0000BF66 7605              <1>       jna    short loc_calc_2nd_fat12_16_addr
                           <1>
                           <1> loc_calc_2nd_fat32_addr:
0000BF68 8B462A            <1>       mov    eax, [esi+LD_BPB+FAT32_FAT_Size]
0000BF6B EB04              <1>       jmp    short loc_calc_2nd_fat_addr
                           <1>
                           <1> loc_calc_2nd_fat12_16_addr:
0000BF6D 0FB7461C          <1>       movzx  eax, word [esi+LD_BPB+FATSecs]
                           <1>
                           <1> loc_calc_2nd_fat_addr:
0000BF71 034660            <1>       add    eax, [esi+LD_FATBegin]
0000BF74 0305[E25A0100]    <1>       add    eax, [FAT_BuffSector]
0000BF7A BB001C0900        <1>       mov    ebx, FAT_Buffer
                           <1>       ; ecx = 1 to 3
0000BF7F E81F320000        <1>       call   disk_write
0000BF84 7207              <1>       jc     short loc_save_FAT_buff_write_err
                           <1>       ; Valid  buffer (1 = valid but do not save)
0000BF86 C605[DE5A0100]01  <1>       mov    byte [FAT_BuffValidData], 1
                           <1>
                           <1> loc_save_FAT_buff_write_err:
0000BF8D 5E                <1>       pop    esi ; *
0000BF8E BB001C0900        <1>       mov    ebx, FAT_Buffer
                           <1>       ; 15/10/2016 (1Dh -> 18)
                           <1>       ; 23/03/2016 (1Dh)
0000BF93 B812000000        <1>       mov    eax, 18 ; Drive not ready or write error
0000BF98 C3                <1>       retn
                           <1>
                           <1> calculate_fat_freespace:
                           <1>       ; 23/03/2016
                           <1>       ; 02/03/2016
                           <1>       ; 01/03/2016
                           <1>       ; 29/02/2016
                           <1>       ; 22/02/2016 (TRDOS 386 =  TRDOS v2.0)
                           <1>       ; 30/04/2011
                           <1>       ; 03/04/2010
                           <1>       ; 2005
                           <1>       ; INPUT ->
                           <1>       ;     EAX = Cluster count to be added or subtracted
                           <1>       ;     If BH = FFh, ESI = TR-DOS Logical Drive Description Table
                           <1>       ;     If BH < FFh, BH = TR-DOS Logical Drive Number
                           <1>       ;     BL:
                           <1>       ;     0 = Calculate, 1 = Add, 2 = Subtract, 3 = Get (Not Set/Calc)
                           <1>       ; OUTPUT ->
                           <1>       ;     EAX = Free Space in sectors
                           <1>       ;     ESI = Logical Dos Drive Description Table address
                           <1>       ;     BH = Logical Dos Drive Number (same with input value of BH)
                           <1>       ;     BL = Type of operation (same with input value of BL)
                           <1>       ;     ECX = 0 -> valid
                           <1>       ;     ECX > 0 -> error or invalid
                           <1>       ;     If EAX = FFFFFFFFh, it is 're-calculation needed'
                           <1>       ;                           sign due to r/w error
                           <1>
0000BF99 66891D[825D0100]  <1>       mov    [CFS_OPType], bx
0000BFA0 A3[845D0100]      <1>       mov    [CFS_CC], eax
                           <1>
0000BFA5 80FFFF            <1>       cmp    bh, 0FFh
0000BFA8 740B              <1>       je     short pass_calculate_freespace_get_drive_dt_offset
                           <1>
                           <1> loc_calculate_freespace_get_drive_dt_offset:
0000BFAA 31C0              <1>       xor    eax, eax
0000BFAC 88FC              <1>         mov     ah, bh
0000BFAE BE00010900        <1>       mov    esi, Logical_DOSDisks
0000BFB3 01C6              <1>         add     esi, eax
                           <1>
                           <1> pass_calculate_freespace_get_drive_dt_offset:
0000BFB5 08DB              <1>       or     bl, bl
0000BFB7 7435              <1>       jz     short loc_reset_fcc
                           <1>
                           <1> loc_get_free_sectors:
0000BFB9 8B4674            <1>       mov    eax, [esi+LD_FreeSectors]
                           <1>
                           <1>       ;xor   ecx, ecx
                           <1>       ;dec   ecx ; 0FFFFFFFFh
                           <1>       ;cmp   eax, ecx ; 29/02/2016
                           <1>       ;je    short loc_get_free_sectors_retn ; recalculation is needed!
                           <1>
                           <1>       ; 23/03/2016
0000BFBC 8B4E70            <1>       mov    ecx, [esi+LD_TotalSectors]
0000BFBF 39C1              <1>       cmp    ecx, eax ; Total sectors must be greater than Free sectors !
0000BFC1 7707              <1>       ja     short loc_get_free_sectors_check_optype
```

```
1149                                <1>
1150 0000BFC3 31C0                 <1>       xor    eax, eax
1151 0000BFC5 48                   <1>       dec    eax ; 0FFFFFFFFh  ; recalculation is needed!
1152 0000BFC6 894674               <1>       mov    [esi+LD_FreeSectors], eax ; reset (for recalculation)
1153                                <1>
1154                                <1> loc_get_free_sectors_retn:
1155 0000BFC9 C3                   <1>       retn
1156                                <1>
1157                                <1> loc_get_free_sectors_check_optype:
1158 0000BFCA 80FB03               <1>       cmp    bl, 3
1159 0000BFCD 7203                 <1>       jb     short loc_set_fcc
1160                                <1>
1161 0000BFCF 29C9                 <1>       sub    ecx, ecx ; 0
1162                                <1>
1163 0000BFD1 C3                   <1>       retn
1164                                <1>
1165                                <1> loc_set_fcc:
1166 0000BFD2 807E0302             <1>       cmp    byte [esi+LD_FATType], 2
1167 0000BFD6 0F87DF000000         <1>         ja       loc_update_FAT32_fs_info_fcc
1168                                <1>
1169                                <1>       ;mov   eax, [esi+LD_FreeSectors]
1170 0000BFDC 0FB64E13             <1>       movzx  ecx, byte [esi+LD_BPB+SecPerClust]
1171 0000BFE0 29D2                 <1>       sub    edx, edx
1172 0000BFE2 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 0000BFE4 A3[FB5A0100]         <1>       mov    [FreeClusterCount], eax
1180 0000BFE9 E988000000           <1>         jmp      loc_set_free_sectors_FAT12_FAT16
1181                                <1>
1182                                <1> loc_reset_fcc:
1183 0000BFEE 31C0                 <1>       xor    eax, eax
1184 0000BFF0 A3[FB5A0100]         <1>       mov    [FreeClusterCount], eax ; 0
1185 0000BFF5 8B5678               <1>       mov    edx, [esi+LD_Clusters]
1186 0000BFF8 42                   <1>       inc    edx
1187 0000BFF9 8915[EA5A0100]       <1>       mov    [LastCluster], edx
1188                                <1>
1189 0000BFFF 807E0302             <1>       cmp    byte [esi+LD_FATType], 2
1190 0000C003 7647                 <1>       jna    short loc_count_free_fat_clusters_0
1191                                <1>
1192 0000C005 48                   <1>       dec    eax ; FFFFFFFFh
1193 0000C006 A3[8C5D0100]         <1>       mov    [CFS_FAT32FC], eax
1194                                <1>
1195                                <1>       ; 29/02/2016
1196 0000C00B 89463A               <1>       mov    [esi+LD_BPB+BPB_Reserved], eax ; reset
1197 0000C00E 89463E               <1>       mov    [esi+LD_BPB+BPB_Reserved+4], eax ; reset
1198                                <1>
1199 0000C011 B802000000           <1>       mov    eax, 2
1200                                <1>
1201                                <1> loc_count_fc_next_cluster_0:
1202 0000C016 50                   <1>       push   eax
1203 0000C017 E801F9FFFF           <1>       call   get_next_cluster
1204 0000C01C 7310                 <1>       jnc    short loc_check_fat32_ff_cluster
1205 0000C01E 09C0                 <1>       or     eax, eax
1206 0000C020 741E                 <1>       jz     short pass_inc_cfs_fcc_0
1207                                <1>
1208                                <1> loc_put_fcc_unknown_sign:
1209 0000C022 58                   <1>       pop    eax
1210                                <1>       ; "Free count is Unknown" sign
1211                                <1>       ;mov   dword [FreeClusterCount], 0FFFFFFFFh
1212                                <1>
1213                                <1>       ; 29/02/2016
1214                                <1>       ; Save Free Cluster Count value in FAT32 'BPB_Reserved' area
1215                                <1>       ;mov   [esi+LD_BPB+BPB_Reserved], 0FFFFFFFFh ; unknown!
1216 0000C023 8B15[8C5D0100]       <1>       mov    edx, [CFS_FAT32FC] ; First Free Cluster
1217                                <1>       ; Save First Free Cluster value in FAT32 'BPB_Reserved+4' area
1218 0000C029 89563E               <1>       mov    [esi+LD_BPB+BPB_Reserved+4], edx
1219                                <1>
1220 0000C02C EB7D                 <1>         jmp      loc_put_fcc_invalid_sign
1221                                <1>
1222                                <1> loc_check_fat32_ff_cluster:
1223 0000C02E 09C0                 <1>       or     eax, eax
1224 0000C030 750E                 <1>       jnz    short pass_inc_cfs_fcc_0
1225 0000C032 58                   <1>       pop    eax
1226 0000C033 A3[8C5D0100]         <1>       mov    [CFS_FAT32FC], eax
1227                                <1>       ;mov   dword [FreeClusterCount], 1
1228 0000C038 FF05[FB5A0100]       <1>       inc    dword [FreeClusterCount]
1229 0000C03E EB27                 <1>       jmp    short pass_inc_cfs_fcc_1
1230                                <1>
1231                                <1> pass_inc_cfs_fcc_0:
1232 0000C040 58                   <1>       pop    eax
1233                                <1>
1234                                <1> pass_inc_cfs_fcc_0c:
1235 0000C041 40                   <1>       inc    eax ; add eax, 1
1236 0000C042 3B05[EA5A0100]       <1>       cmp    eax, [LastCluster]
1237 0000C048 76CC                 <1>       jna    short loc_count_fc_next_cluster_0
1238 0000C04A EB6F                 <1>       jmp    short loc_update_FAT32_fs_info_fcc
1239                                <1>
1240                                <1> loc_count_free_fat_clusters_0:
1241                                <1>       ;mov   eax, 2
1242 0000C04C B002                 <1>       mov    al, 2
1243                                <1>
1244                                <1> loc_count_fc_next_cluster:
1245 0000C04E 50                   <1>       push   eax
1246 0000C04F E8C9F8FFFF           <1>       call   get_next_cluster
1247 0000C054 720C                 <1>       jc     short loc_count_fcc_stc
1248                                <1>
1249                                <1> loc_count_free_clusters_1:
1250 0000C056 21C0                 <1>       and    eax, eax
```

```
1251 0000C058 750C                <1>       jnz    short pass_inc_cfs_fcc
1252                              <1>
1253 0000C05A FF05[FB5A0100]      <1>       inc    dword [FreeClusterCount]
1254 0000C060 EB04                <1>       jmp    short pass_inc_cfs_fcc
1255                              <1>
1256                              <1> loc_count_fcc_stc:
1257 0000C062 09C0                <1>       or     eax, eax
1258 0000C064 75BC                <1>       jnz    short loc_put_fcc_unknown_sign ; 29/02/2016
1259                              <1>
1260                              <1> pass_inc_cfs_fcc:
1261 0000C066 58                  <1>       pop    eax
1262                              <1>
1263                              <1> pass_inc_cfs_fcc_1:
1264 0000C067 40                  <1>       inc    eax ; add eax, 1
1265 0000C068 3B05[EA5A0100]      <1>       cmp    eax, [LastCluster]
1266 0000C06E 76DE                <1>       jna    short loc_count_fc_next_cluster
1267                              <1>
1268                              <1> loc_set_free_sectors:
1269 0000C070 807E0302            <1>       cmp    byte [esi+LD_FATType], 2
1270 0000C074 7745                <1>       ja     short loc_update_FAT32_fs_info_fcc
1271                              <1>
1272                              <1> loc_set_free_sectors_FAT12_FAT16:
1273 0000C076 803D[825D0100]00    <1>       cmp    byte [CFS_OPType], 0
1274 0000C07D 761C                <1>       jna    short pass_FAT_add_sub_fcc
1275 0000C07F A1[845D0100]        <1>       mov    eax, [CFS_CC]
1276 0000C084 803D[825D0100]01    <1>       cmp    byte [CFS_OPType], 1
1277 0000C08B 7708                <1>       ja     short pass_FAT_add_fcc
1278 0000C08D 0105[FB5A0100]      <1>       add    [FreeClusterCount], eax
1279 0000C093 EB06                <1>       jmp    short pass_FAT_add_sub_fcc
1280                              <1>
1281                              <1> pass_FAT_add_fcc:
1282 0000C095 2905[FB5A0100]      <1>       sub    [FreeClusterCount], eax
1283                              <1>
1284                              <1> pass_FAT_add_sub_fcc:
1285 0000C09B 0FB64613            <1>       movzx  eax, byte [esi+LD_BPB+SecPerClust]
1286 0000C09F 8B15[FB5A0100]      <1>       mov    edx, [FreeClusterCount]
1287 0000C0A5 F7E2                <1>       mul    edx
1288                              <1>
1289 0000C0A7 31C9                <1>       xor    ecx, ecx
1290 0000C0A9 EB05                <1>       jmp    short loc_cfs_retn_params
1291                              <1>
1292                              <1> loc_put_fcc_invalid_sign:
1293 0000C0AB 29C0                <1>             sub    eax, eax ; 0
1294 0000C0AD 48                  <1>       dec    eax ; FFFFFFFFh
1295                              <1> loc_fat32_ffc_recalc_needed:
1296 0000C0AE 89C1                <1>       mov    ecx, eax
1297                              <1>
1298                              <1> loc_cfs_retn_params:
1299 0000C0B0 894674              <1>       mov    [esi+LD_FreeSectors], eax
1300 0000C0B3 0FB71D[825D0100]    <1>       movzx  ebx, word [CFS_OPType]
1301 0000C0BA C3                  <1>       retn
1302                              <1>
1303                              <1> loc_update_FAT32_fs_info_fcc:
1304                              <1> loc_check_fcc_FSINFO_op:
1305                              <1>       ; 29/02/2016
1306                              <1>       ; EAX = Free cluster count (before this update) ; value from disk
1307                              <1>       ; EDX = First Free Cluster (before this update) ; value from disk
1308 0000C0BB 803D[825D0100]01    <1>       cmp    byte [CFS_OPType], 1
1309 0000C0C2 7221                <1>       jb     short loc_cfs_FAT32_get_rcalc_parms ; 0 = recalculated
1310 0000C0C4 7406                <1>       je     short loc_check_fcc_FSINFO_op1 ; 1 = add
1311                              <1> loc_check_fcc_FSINFO_op2: ; subtract
1312 0000C0C6 F71D[845D0100]      <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 0000C0CC 31D2                <1>       xor    edx, edx ; 0
1316 0000C0CE 4A                  <1>       dec    edx ; 0FFFFFFFFh
1317 0000C0CF 8B463A              <1>       mov    eax, [esi+LD_BPB+BPB_Reserved]
1318 0000C0D2 39D0                <1>       cmp    eax, edx
1319 0000C0D4 73D5                <1>       jnb    short loc_put_fcc_invalid_sign
1320 0000C0D6 0305[845D0100]      <1>         add     eax, [CFS_CC] ; free cluster count on disk + current count
1321 0000C0DC 72CD                <1>       jc     short loc_put_fcc_invalid_sign
1322                              <1>
1323 0000C0DE A3[FB5A0100]        <1>       mov    [FreeClusterCount], eax
1324 0000C0E3 EB0E                <1>       jmp    short loc_cfs_write_FSINFO_sector
1325                              <1>
1326                              <1> loc_cfs_FAT32_get_rcalc_parms:
1327 0000C0E5 8B15[8C5D0100]      <1>       mov    edx, [CFS_FAT32FC]
1328 0000C0EB A1[FB5A0100]        <1>       mov    eax, [FreeClusterCount]
1329 0000C0F0 89563E              <1>       mov    [esi+LD_BPB+BPB_Reserved+4], edx ; First Free Cluster
1330                              <1> loc_cfs_write_FSINFO_sector:
1331 0000C0F3 89463A              <1>       mov    [esi+LD_BPB+BPB_Reserved], eax ; Free cluster count
1332                              <1>       ; 01/03/2016
1333 0000C0F6 E8AA000000          <1>       call   set_fat32_fsinfo_sector_parms
1334 0000C0FB 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 0000C0FD 8B0D[FB5A0100]      <1>       mov    ecx, [FreeClusterCount]
1344 0000C103 0FB64613            <1>       movzx  eax, byte [esi+LD_BPB+SecPerClust]
1345 0000C107 F7E1                <1>       mul    ecx
1346                              <1>       ; 29/02/2016
1347 0000C109 31C9                <1>       xor    ecx, ecx ; 0
1348 0000C10B 09D2                <1>       or     edx, edx ; 0 ?
1349 0000C10D 759C                <1>         jnz     loc_put_fcc_invalid_sign
1350 0000C10F 394670              <1>       cmp    [esi+LD_TotalSectors], eax ; Volume size in sectors
1351 0000C112 7697                <1>         jna     short loc_put_fcc_invalid_sign
1352                              <1>       ;
```

```
1353                                    <1> loc_set_FAT32_free_sectors_ok:
1354 0000C114 31D2                     <1>       xor    edx, edx ; 0
1355 0000C116 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 0000C118 89C1                     <1>       mov    ecx, eax
1377                                    <1>
1378 0000C11A C705[945D0100]FFFF-      <1>       mov    dword [glc_index], 0FFFFFFFFh ; 22/10/2016
1378 0000C122 FFFF                     <1>
1379                                    <1>
1380                                    <1> loc_glc_get_next_cluster_1:
1381 0000C124 890D[905D0100]           <1>       mov    [glc_prevcluster], ecx
1382                                    <1>       ; 22/10/2016
1383 0000C12A FF05[945D0100]           <1>       inc    dword [glc_index]
1384                                    <1>
1385                                    <1> loc_glc_get_next_cluster_2:
1386 0000C130 E8E8F7FFFF               <1>       call   get_next_cluster
1387                                    <1>       ; ecx = current/previous cluster
1388                                    <1>       ; eax = next/last cluster
1389 0000C135 73ED                     <1>       jnc    short loc_glc_get_next_cluster_1
1390                                    <1>
1391 0000C137 09C0                     <1>       or     eax, eax
1392 0000C139 7509                     <1>       jnz    short loc_glc_stc_retn
1393                                    <1>
1394                                    <1>       ; ecx = previous cluster
1395 0000C13B 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 0000C13D 8B0D[905D0100]           <1>       mov    ecx, [glc_prevcluster]
1402 0000C143 C3                       <1>       retn
1403                                    <1>
1404                                    <1> loc_glc_stc_retn:
1405 0000C144 F5                       <1>       cmc    ;stc
1406 0000C145 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 0000C147 31C9                     <1>       xor    ecx, ecx ; mov ecx, 0
1426                                    <1>       ;mov   byte [FAT_BuffValidData], 0
1427 0000C149 890D[E65A0100]           <1>       mov    [FAT_ClusterCounter], ecx ; 0 ; reset
1428                                    <1>
1429                                    <1> loc_tcc_unlink_clusters:
1430 0000C14F 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 0000C154 73F9                     <1>       jnc short loc_tcc_unlink_clusters
1437                                    <1>
1438                                    <1> pass_tcc_unlink_clusters:
1439 0000C156 A2[9B5D0100]             <1>       mov    byte [TCC_FATErr], al
1440 0000C15B 803D[DE5A0100]02         <1>       cmp    byte [FAT_BuffValidData], 2
1441 0000C162 750E                     <1>       jne    short loc_tcc_calculate_FAT_freespace
1442 0000C164 E89BFDFFFF               <1>       call   save_fat_buffer
1443 0000C169 7307                     <1>       jnc    short loc_tcc_calculate_FAT_freespace
1444 0000C16B A2[9B5D0100]             <1>       mov    byte [TCC_FATErr], al ; Error
1445                                    <1>       ;mov   byte [FAT_BuffValidData], 0
1446                                    <1>
1447                                    <1>       ; 01/03/2016
1448 0000C170 EB12                     <1>       jmp    short loc_tcc_recalculate_FAT_freespace
1449                                    <1>
1450                                    <1> loc_tcc_calculate_FAT_freespace:
1451 0000C172 A1[E65A0100]             <1>       mov    eax, [FAT_ClusterCounter] ; signed (+-) number
1452 0000C177 66BB01FF                 <1>       mov    bx, 0FF01h ; BH = FFh -> ESI = Dos drv desc. table
1453                                    <1>                         ; BL = 1 -> add cluster
```

```
1454 0000C17B E819FEFFFF          <1>       call   calculate_fat_freespace
1455 0000C180 21C9                <1>       and    ecx, ecx ; cx = 0 -> valid free sector count
1456 0000C182 7409                <1>       jz     short pass_truncate_cc_recalc_FAT_freespace
1457                              <1>
1458                              <1> loc_tcc_recalculate_FAT_freespace:
1459 0000C184 66BB00FF            <1>       mov    bx, 0FF00h ; recalculate !
1460 0000C188 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 0000C18D 8B0D[E65A0100]      <1>       mov    ecx, [FAT_ClusterCounter]
1465                              <1>
1466 0000C193 803D[9B5D0100]00    <1>       cmp    byte [TCC_FATErr], 0
1467 0000C19A 7608                <1>       jna    short loc_tcc_unlink_clusters_retn
1468                              <1>
1469                              <1> loc_tcc_unlink_clusters_error:
1470 0000C19C 0FB605[9B5D0100]    <1>       movzx  eax, byte [TCC_FATErr]
1471 0000C1A3 F9                  <1>       stc
1472                              <1> loc_tcc_unlink_clusters_retn:
1473 0000C1A4 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 0000C1A5 E824000000          <1>       call   get_fat32_fsinfo_sector_parms
1491 0000C1AA 7221                <1>       jc     short update_fat32_fsinfo_sector_retn
1492                              <1>
1493 0000C1AC 8B463A              <1>       mov    eax, [esi+LD_BPB+BPB_Reserved] ; Free Cluster Count
1494 0000C1AF 8B563E              <1>       mov    edx, [esi+LD_BPB+BPB_Reserved+4] ; First free Cluster
1495                              <1>
1496                              <1>         ;mov ebx, DOSBootSectorBuff
1497 0000C1B2 8983E8010000        <1>       mov    [ebx+488], eax
1498 0000C1B8 8993EC010000        <1>       mov    [ebx+492], edx
1499                              <1>
1500 0000C1BE A1[885D0100]        <1>       mov    eax, [CFS_FAT32FSINFOSEC]
1501 0000C1C3 B901000000          <1>       mov    ecx, 1
1502 0000C1C8 E8D62F0000          <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 0000C1CD 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 0000C1CE 0FB74636            <1>       movzx  eax, word [esi+LD_BPB+FAT32_FSInfoSec]
1534 0000C1D2 03466C              <1>       add    eax, [esi+LD_StartSector]
1535 0000C1D5 A3[885D0100]        <1>       mov    [CFS_FAT32FSINFOSEC], eax
1536                              <1>
1537 0000C1DA BB[DA580100]        <1>         mov     ebx, DOSBootSectorBuff
1538 0000C1DF B901000000          <1>       mov    ecx, 1
1539 0000C1E4 E8C92F0000          <1>       call   disk_read
1540 0000C1E9 7232                <1>       jc     short loc_read_FAT32_fsinfo_sec_err
1541                              <1>
1542 0000C1EB BB[DA580100]        <1>       mov    ebx, DOSBootSectorBuff
1543                              <1>
1544 0000C1F0 813B52526141        <1>       cmp    dword [ebx], 41615252h
1545 0000C1F6 751E                <1>       jne    short loc_read_FAT32_fsinfo_sec_stc
1546                              <1>
1547 0000C1F8 81BBE4010000727241- <1>       cmp    dword [ebx+484], 61417272h
1547 0000C201 61                  <1>
1548 0000C202 7512                <1>       jne    short loc_read_FAT32_fsinfo_sec_stc
1549                              <1>
1550 0000C204 A1[885D0100]        <1>       mov    eax, [CFS_FAT32FSINFOSEC]
1551 0000C209 8B8BE8010000        <1>       mov    ecx, [ebx+488] ; free cluster count
1552 0000C20F 8B93EC010000        <1>       mov    edx, [ebx+492] ; first (next) free cluster
1553                              <1>
1554 0000C215 C3                  <1>       retn
```

```
1555                                    <1>
1556                                    <1> loc_read_FAT32_fsinfo_sec_stc:
1557                                    <1>       ; 15/10/2016 (0Bh -> 28)
1558 0000C216 B81C000000               <1>       mov    eax, 28 ; Invalid format!
1559 0000C21B 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 0000C21D B811000000               <1>       mov    eax, 17 ; Drive not ready or read error
1565                                    <1>
1566                                    <1> loc_read_FAT32_fsinfo_sec_stc_retn:
1567 0000C222 29DB                     <1>       sub    ebx, ebx ; 0
1568 0000C224 F9                       <1>       stc
1569 0000C225 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 0000C226 A3[B85E0100]             <1>       mov    [FAT_anc_LCluster], eax
1594                                    <1>
1595 0000C22B E844F9FFFF               <1>       call   get_first_free_cluster
1596 0000C230 720B                     <1>       jc     short loc_add_new_cluster_retn
1597                                    <1>       ; EAX >= 2 and EAX < FFFFFFFFh is valid
1598                                    <1>
1599 0000C232 89C2                     <1>       mov    edx, eax
1600                                    <1>
1601 0000C234 42                       <1>       inc    edx
1602                                    <1>       ;jnz   short loc_add_new_cluster_check_ffc_eax
1603 0000C235 7516                     <1>       jnz    short loc_add_new_cluster_save_fcc
1604                                    <1>
1605                                    <1> loc_add_new_cluster_no_disk_space_retn:
1606 0000C237 B827000000               <1>       mov    eax, 27h ; MSDOS err => insufficient disk space
1607                                    <1> loc_add_new_cluster_stc_retn:
1608 0000C23C F9                       <1>       stc
1609                                    <1> loc_add_new_cluster_retn:
1610 0000C23D 0FB65E13                 <1>       movzx  ebx, byte [esi+LD_BPB+SecPerClust]
1611 0000C241 8B4E74                   <1>       mov    ecx, [esi+LD_FreeSectors]
1612                                    <1>       ;xor   edx, edx
1613                                    <1>       ;stc
1614 0000C244 C3                       <1>       retn
1615                                    <1>
1616                                    <1> loc_anc_invalid_format_stc_retn:
1617 0000C245 F9                       <1>       stc
1618                                    <1> loc_add_new_cluster_invalid_format_retn:
1619                                    <1>       ; 15/10/2016 (0Bh -> 28)
1620 0000C246 B81C000000               <1>       mov    eax, 28 ; Invalid format
1621 0000C24B 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 0000C24D A3[BC5E0100]             <1>       mov    [FAT_anc_FFCluster], eax
1629                                    <1>
1630 0000C252 83E802                   <1>       sub    eax, 2
1631 0000C255 0FB65E13                 <1>         movzx   ebx, byte [esi+LD_BPB+SecPerClust]
1632 0000C259 F7E3                     <1>       mul    ebx
1633 0000C25B 09D2                     <1>       or     edx, edx
1634 0000C25D 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 0000C25F 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 0000C260 BF00000700               <1>       mov    edi, Cluster_Buffer ; 70000h (for current TRDOS 386 version)
1651 0000C265 89D9                     <1>       mov    ecx, ebx ; sector count
1652 0000C267 C1E107                   <1>       shl    ecx, 7 ; 1 sector = 512 bytes -> 128 double words
1653                                    <1>       ;xor   eax, eax ; 0
1654 0000C26A 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 0000C26C 89D0             <1>       mov    eax, edx
1660 0000C26E 034668           <1>         add     eax, [esi+LD_DATABegin]
1661 0000C271 72D3             <1>       jc     short loc_add_new_cluster_invalid_format_retn
1662                            <1>
1663 0000C273 89D9             <1>       mov    ecx, ebx ; ECX = sectors per cluster (<256)
1664 0000C275 BB00000700       <1>       mov    ebx, Cluster_Buffer
1665 0000C27A E8242F0000       <1>       call   disk_write
1666 0000C27F 7307             <1>       jnc    short loc_add_new_cluster_update_fat_nlc
1667                            <1>
1668                            <1>       ; 15/10/2016 (1Dh -> 18)
1669 0000C281 B812000000       <1>       mov    eax, 18 ; Write Error
1670 0000C286 EBB4             <1>       jmp    short loc_add_new_cluster_stc_retn
1671                            <1>
1672                            <1> loc_add_new_cluster_update_fat_nlc:
1673 0000C288 A1[BC5E0100]     <1>       mov    eax, [FAT_anc_FFCluster]
1674 0000C28D 31C9             <1>       xor    ecx, ecx
1675 0000C28F 890D[E65A0100]   <1>       mov    [FAT_ClusterCounter], ecx ; 0 ; reset
1676 0000C295 49               <1>       dec    ecx ; 0FFFFFFFFh
1677 0000C296 E8ACF9FFFF       <1>       call   update_cluster
1678 0000C29B 7304             <1>       jnc    short loc_add_new_cluster_update_fat_plc
1679 0000C29D 09C0             <1>       or     eax, eax ;EAX = 0 -> cluster value is 0 or eocc
1680 0000C29F 759B             <1>       jnz    short loc_add_new_cluster_stc_retn
1681                            <1>
1682                            <1> loc_add_new_cluster_update_fat_plc:
1683 0000C2A1 A1[B85E0100]     <1>       mov    eax, [FAT_anc_LCluster]
1684 0000C2A6 8B0D[BC5E0100]   <1>       mov    ecx, [FAT_anc_FFCluster]
1685 0000C2AC E896F9FFFF       <1>       call   update_cluster
1686 0000C2B1 7314             <1>       jnc    short loc_add_new_cluster_save_fat_buffer
1687 0000C2B3 09C0             <1>       or     eax, eax ; EAX = 0 -> cluster value is 0 or eocc
1688 0000C2B5 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 0000C2B7 66BB00FF         <1>       mov    bx, 0FF00h ; recalculate free space (BL = 0)
1695                            <1>                         ; (BH = FFh -> Use ESI as Drv Param. Tbl.)
1696 0000C2BB 50               <1>       push   eax
1697 0000C2BC E8D8FCFFFF       <1>       call   calculate_fat_freespace
1698 0000C2C1 58               <1>       pop    eax
1699 0000C2C2 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 0000C2C7 E838FCFFFF       <1>       call   save_fat_buffer
1706 0000C2CC 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 0000C2CE A1[E65A0100]     <1>       mov    eax, [FAT_ClusterCounter]
1711 0000C2D3 66BB01FF         <1>       mov    bx, 0FF01h ; BH = FFh -> ESI -> Dos drv desc. table
1712                            <1>              ; BL = 1 -> add cluster
1713 0000C2D7 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 0000C2D9 E8BBFCFFFF       <1>         call    calculate_fat_freespace
1717                            <1>       ;ECX = 0 -> no error, ECX > 0 -> error or invalid return
1718 0000C2DE 21C9             <1>       and    ecx, ecx ; ECX = 0 -> valid free sector count
1719 0000C2E0 7409             <1>       jz     short loc_add_new_cluster_return_cluster_number
1720                            <1>
1721                            <1> loc_add_new_cluster_recalc_FAT_freespace:
1722 0000C2E2 66BB00FF         <1>       mov    bx, 0FF00h  ; recalculate free space
1723 0000C2E6 E8AEFCFFFF       <1>         call    calculate_fat_freespace
1724                            <1>       ; cf = 0
1725                            <1> loc_add_new_cluster_return_cluster_number:
1726 0000C2EB 89C1             <1>       mov    ecx, eax ; Free sector count
1727 0000C2ED A1[BC5E0100]     <1>       mov    eax, [FAT_anc_FFCluster]
1728 0000C2F2 0FB65E13         <1>       movzx  ebx, byte [esi+LD_BPB+SecPerClust]
1729                            <1>       ;mov   edi, Cluster_Buffer
1730 0000C2F6 31D2             <1>       xor    edx, edx
1731 0000C2F8 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 0000C2F9 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 0000C2FD 807E0300         <1>       cmp    byte [esi+LD_FATType], 0
1755 0000C301 761C             <1>       jna    short write_fs_cluster
1756                            <1>
1757                            <1> write_fat_file_sectors:
1758 0000C303 83E802           <1>       sub    eax, 2 ; Beginning cluster number is always 2
```

```
1759 0000C306 0FB65613            <1>       movzx  edx, byte [esi+LD_BPB+BPB_SecPerClust] ; 18/03/2016
1760 0000C30A F7E2                <1>       mul    edx
1761 0000C30C 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 0000C30F E88F2E0000          <1>       call   disk_write
1770 0000C314 7306                <1>       jnc    short wclust_retn
1771                              <1>
1772                              <1>       ; 15/10/2016 (1Dh -> 18)
1773 0000C316 B812000000          <1>       mov    eax, 18 ; Drive not ready or write error !
1774 0000C31B C3                  <1>       retn
1775                              <1>
1776                              <1> wclust_retn:
1777 0000C31C 29C0                <1>       sub    eax, eax ; 0
1778 0000C31E 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 0000C31F B980000000          <1>       mov    ecx, 128 ; maximum count of sectors (before eof)
1793 0000C324 E801000000          <1>       call   write_fs_sectors
1794 0000C329 C3                  <1>       retn
1795                              <1>
1796                              <1> write_fs_sectors:
1797                              <1>       ; 21/03/2016 (TRDOS 386 =  TRDOS v2.0)
1798 0000C32A F9                  <1>       stc
1799 0000C32B 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 0000C32C 807E0301            <1>       cmp    byte [esi+LD_FATType], 1
1816 0000C330 721E                <1>         jb      short get_fs_section_by_index
1817                              <1>
1818 0000C332 3B4E78              <1>       cmp    ecx, [esi+LD_Clusters]
1819 0000C335 7207                <1>       jb     short gcbi_1
1820                              <1> gcbi_0:
1821 0000C337 F9                  <1>       stc
1822 0000C338 B823000000          <1>       mov    eax, 23h ; Cluster not available !
1823                              <1>                       ; MSDOS error code: FCB unavailable
1824 0000C33D C3                  <1>       retn
1825                              <1> gcbi_1:
1826 0000C33E 51                  <1>       push   ecx
1827 0000C33F E8D9F5FFFF          <1>       call   get_next_cluster
1828 0000C344 59                  <1>       pop    ecx
1829 0000C345 7203                <1>       jc     short gcbi_3
1830 0000C347 E2F5                <1>       loop   gcbi_1
1831                              <1> gcbi_2:
1832 0000C349 C3                  <1>       retn
1833                              <1> gcbi_3:
1834 0000C34A 09C0                <1>       or     eax, eax
1835 0000C34C 74E9                <1>       jz     short gcbi_0
1836 0000C34E F5                  <1>       cmc    ; stc
1837 0000C34F 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 0000C350 B8FFFFFFFF          <1>       mov    eax, 0FFFFFFFFh
1854 0000C355 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 0000C356 B800000000               <1>       mov    eax, 0
1872 0000C35B BA00000000               <1>       mov    edx, 0
1873 0000C360 C3                       <1>       retn
2309                                        %include 'trdosk6.s' ; 24/01/2016
   1                                    <1> ; ****************************************************************************
   2                                    <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - MAIN PROGRAM : trdosk6.s
   3                                    <1> ; ----------------------------------------------------------------------------
   4                                    <1> ; Last Update: 14/10/2017
   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> ; 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 0000C361 368925[5C030300]          <1>         mov     [ss:u.sp], esp ; Kernel stack points to return address
  54                                    <1>
  55                                    <1>       ; save user registers
  56 0000C368 1E                        <1>       push   ds
  57 0000C369 06                        <1>       push   es
  58 0000C36A 0FA0                      <1>       push   fs
  59 0000C36C 0FA8                      <1>       push   gs
  60 0000C36E 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>       ;
  66 0000C36F 50                        <1>       push   eax ; 01/07/2015
  67 0000C370 66B81000                  <1>       mov     ax, KDATA
  68 0000C374 8ED8                      <1>         mov     ds, ax
  69 0000C376 8EC0                      <1>         mov     es, ax
  70 0000C378 8EE0                      <1>         mov     fs, ax
  71 0000C37A 8EE8                      <1>         mov     gs, ax
  72 0000C37C A1[00520100]              <1>       mov    eax, [k_page_dir]
  73 0000C381 0F22D8                    <1>       mov    cr3, eax
  74 0000C384 58                        <1>       pop    eax ; 01/07/2015
  75                                    <1>       ; 19/10/2015
  76 0000C385 FC                        <1>       cld
  77                                    <1>       ;
  78 0000C386 FE05[5B030300]            <1>       inc    byte [sysflg]
  79                                    <1>              ; incb sysflg / indicate a system routine is in progress
  80 0000C38C FB                        <1>         sti  ; 18/01/2014
  81 0000C38D 0F851EA0FFFF              <1>       jnz     panic ; 24/05/2013
  82                                    <1>              ; beq 1f
  83                                    <1>              ; jmp panic ; / called if trap inside system
  84                                    <1> ;1:
  85                                    <1>       ; 17/03/2017
  86 0000C393 80642438FE                <1>       and    byte [esp+ESPACE+8], ~1 ; clear carry flag
  87                                    <1>
  88                                    <1>       ; 16/04/2015
```

```
 89 0000C398 A3[64030300]        <1>       mov    [u.r0], eax
 90 0000C39D 8925[60030300]      <1>       mov    [u.usp], esp ; kernel stack points to user's registers
 91                              <1>
 92                              <1>       ; 13/01/2017 (TRDOS 386 Feaure only !)
 93 0000C3A3 803D[D4030300]00    <1>       cmp    byte [u.t_lock], 0 ; timer interrupt lock ?
 94 0000C3AA 0F879D010000        <1>       ja     sysrele                ; yes, sys release only !!!
 95                              <1>
 96                              <1>              ; mov $s.syst+2,clockp
 97                              <1>              ; mov r0,-(sp) / save user registers
 98                              <1>              ; mov sp,u.r0 / pointer to bottom of users stack
 99                              <1>                       ; / in u.r0
100                              <1>              ; mov r1,-(sp)
101                              <1>              ; mov r2,-(sp)
102                              <1>              ; mov r3,-(sp)
103                              <1>              ; mov r4,-(sp)
104                              <1>              ; mov r5,-(sp)
105                              <1>              ; mov ac,-(sp) / "accumulator" register for extended
106                              <1>                       ; / arithmetic unit
107                              <1>              ; mov mq,-(sp) / "multiplier quotient" register for the
108                              <1>                       ; / extended arithmetic unit
109                              <1>              ; mov sc,-(sp) / "step count" register for the extended
110                              <1>                       ; / arithmetic unit
111                              <1>              ; mov sp,u.sp / u.sp points to top of users stack
112                              <1>              ; mov 18.(sp),r0 / store pc in r0
113                              <1>              ; mov -(r0),r0 / sys inst in r0      10400xxx
114                              <1>              ; sub $sys,r0 / get xxx code
115 0000C3B0 C1E002              <1>       shl    eax, 2
116                              <1>              ; asl r0 / multiply by 2 to jump indirect in bytes
117 0000C3B3 3DB8000000          <1>       cmp    eax, end_of_syscalls - syscalls
118                              <1>              ; cmp r0,$2f-1f / limit of table (35) exceeded
119                              <1>       ;jnb   short badsys
120                              <1>              ; bhis badsys / yes, bad system call
121 0000C3B8 F5                  <1>       cmc
122 0000C3B9 9C                  <1>       pushf
123 0000C3BA 50                  <1>       push   eax
124 0000C3BB 8B2D[5C030300]      <1>       mov    ebp, [u.sp] ; Kernel stack at the beginning of sys call
125 0000C3C1 B0FE                <1>       mov    al, 0FEh ; 11111110b
126 0000C3C3 1400                <1>       adc    al, 0 ; al = al + cf
127 0000C3C5 204508              <1>       and    [ebp+8], al ; flags (reset carry flag)
128                              <1>              ; bic $341,20.(sp) / set users processor priority to 0
129                              <1>                              ; / and clear carry bit
130 0000C3C8 5D                  <1>       pop    ebp ; eax
131 0000C3C9 9D                  <1>       popf
132 0000C3CA 0F8208020000        <1>         jc      badsys
133 0000C3D0 A1[64030300]        <1>       mov    eax, [u.r0]
134                              <1>       ; system call registers: EAX, EDX, ECX, EBX, ESI, EDI
135 0000C3D5 FFA5[DBC30000]      <1>       jmp    dword [ebp+syscalls]
136                              <1>              ; jmp *1f(r0) / jump indirect thru table of addresses
137                              <1>                            ; / to proper system routine.
138                              <1> syscalls: ; 1:
139                              <1>       ; 28/02/2017
140                              <1>       ; 20/02/2017
141                              <1>       ; 19/02/2017
142                              <1>       ; 15/10/2016
143                              <1>       ; 20/05/2016
144                              <1>       ; 19/05/2016
145                              <1>       ; 16/05/2016
146                              <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
147                              <1>       ; 21/09/2015
148                              <1>       ; 01/07/2015
149                              <1>       ; 16/04/2015 (32 bit address modification)
150                              <1>       ;dd sysrele  ; / 0
151 0000C3DB [BBE70000]          <1>       dd sysver    ; 0 ; Get TRDOS 386 version number (v2.0)
152 0000C3DF [3AC60000]          <1>       dd sysexit   ; / 1
153 0000C3E3 [10C80000]          <1>       dd sysfork   ; / 2
154 0000C3E7 [44CC0000]          <1>       dd sysread   ; / 3
155 0000C3EB [63CC0000]          <1>       dd syswrite  ; / 4
156 0000C3EF [FAC90000]          <1>       dd sysopen   ; / 5
157 0000C3F3 [1BCC0000]          <1>       dd sysclose  ; / 6
158 0000C3F7 [92C70000]          <1>       dd syswait   ; / 7
159 0000C3FB [28C90000]          <1>       dd syscreat  ; / 8
160 0000C3FF [12D80000]          <1>       dd syslink   ; / 9
161 0000C403 [5DD80000]          <1>       dd sysunlink ; / 10
162 0000C407 [EAD80000]          <1>       dd sysexec   ; / 11
163 0000C40B [E1DC0000]          <1>       dd syschdir  ; / 12
164 0000C40F [C0DD0000]          <1>       dd systime   ; / 13
165 0000C413 [DDCB0000]          <1>       dd sysmkdir  ; / 14
166 0000C417 [33DD0000]          <1>       dd syschmod  ; / 15
167 0000C41B [90DD0000]          <1>       dd syschown  ; / 16
168 0000C41F [F3DD0000]          <1>       dd sysbreak  ; / 17
169 0000C423 [91DA0000]          <1>       dd sysstat   ; / 18
170 0000C427 [38DE0000]          <1>       dd sysseek   ; / 19
171 0000C42B [4ADE0000]          <1>       dd systell   ; / 20
172 0000C42F [33DF0000]          <1>       dd sysmount  ; / 21
173 0000C433 [47DF0000]          <1>       dd sysumount ; / 22
174 0000C437 [BCDE0000]          <1>       dd syssetuid ; / 23
175 0000C43B [EDDE0000]          <1>       dd sysgetuid ; / 24
176 0000C43F [CFDD0000]          <1>       dd sysstime  ; / 25
177 0000C443 [B0DE0000]          <1>       dd sysquit   ; / 26
178 0000C447 [A4DE0000]          <1>       dd sysintr   ; / 27
179 0000C44B [A5DA0000]          <1>       dd sysfstat  ; / 28
180 0000C44F [FACC0000]          <1>       dd sysemt    ; / 29
181 0000C453 [ABCE0000]          <1>       dd sysmdate  ; / 30
182                              <1>       ;dd sysstty  ; / 31
183 0000C457 [BFCE0000]          <1>       dd sysvideo  ; 31 ; TRDOS 386 Video Functions (16/05/2016)
184                              <1>       ;dd sysgtty  ; / 32
185 0000C45B [C3FA0000]          <1>       dd sysaudio  ; 32 ; TRDOS 386 Audio Functions (16/05/2016)
186                              <1>       ;dd sysilgins; / 33
187 0000C45F [13CD0000]          <1>       dd systimer  ; 33 ; TRDOS 386 Timer Functions (18/05/2016)
188 0000C463 [5BDF0000]          <1>       dd syssleep  ; 34 ; Retro UNIX 8086 v1 feature only !
189                              <1>                            ; 11/06/2014
190 0000C467 [8ADF0000]          <1>       dd sysmsg    ; 35 ; Retro UNIX 386 v1 feature only !
```

```
                                <1>                         ; 01/07/2015
0000C46B [60E00000]             <1>       dd sysgeterr ; 36 ; Retro UNIX 386 v1 feature only !
                                <1>                         ; 21/09/2015 - get last error number
0000C46F [70F10000]             <1>       dd sysfpstat ; 37 ; TRDOS 386 FPU state option (28/02/2017)
0000C473 [D9E70000]             <1>       dd syspri    ; 38 ; change priority - TRDOS 386 (20/05/2016)
0000C477 [4DC50000]             <1>       dd sysrele   ; 39 ; TRDOS 386 (19/05/2016) (0 -> 39)
0000C47B [0CE90000]             <1>       dd sysfff    ; 40 ; Find First File - TRDOS 386 (15/10/2016)
0000C47F [EBE90000]             <1>       dd sysfnf    ; 41 ; Find Next File - TRDOS 386 (15/10/2016)
0000C483 [5BF00000]             <1>       dd sysalloc  ; 42 ; Allocate contiguous memory block/pages
                                <1>                         ; TRDOS 386 (19/02/2017) DMA buff fuctions
0000C487 [19F10000]             <1>       dd sysdalloc ; 43 ; Deallocate contiguous memory block/pages
                                <1>                         ; TRDOS 386 (19/02/2017) DMA buff fuctions
0000C48B [54F10000]             <1>       dd syscalbac ; 44 ; IRQ Callback and Signal Response Byte
                                <1>                         ; service setup - TRDOS 386 (20/02/2017)
                                <1>                         ; 28/08/2017 (20/08/2017)
0000C48F [47030100]             <1>       dd sysdma    ; 45 ; TRDOS 386 - (ISA) DMA service
                                <1>
                                <1> end_of_syscalls:
                                <1>
                                <1> error:
                                <1>       ; 18/05/2016
                                <1>       ; 13/05/2016
                                <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; 16/04/2015 - 17/09/2015 (Retro UNIX 386 v1)
                                <1>       ; 10/04/2013 - 07/08/2013 (Retro UNIX 8086 v1)
                                <1>       ;
                                <1>       ; 'error' merely sets the error bit off the processor status (c-bit)
                                <1>       ; then falls right into the 'sysret', 'sysrele' return sequence.
                                <1>       ;
                                <1>       ; INPUTS -> none
                                <1>       ; OUTPUTS ->
                                <1>       ;     processor status - carry (c) bit is set (means error)
                                <1>       ;
                                <1>       ; 26/05/2013 (Stack pointer must be reset here!
                                <1>       ;           Because, jumps to error procedure
                                <1>       ;           disrupts push-pop nesting balance)
                                <1>       ;
0000C493 8B2D[5C030300]         <1>       mov   ebp, [u.sp] ; interrupt (system call) return (iretd) address
0000C499 804D0801               <1>       or    byte [ebp+8], 1  ; set carry bit of flags register
                                <1>                            ; (system call will return with cf = 1)
                                <1>             ; bis $1,20.(r1) / set c bit in processor status word below
                                <1>                             ; / users stack
                                <1>       ; 17/09/2015
0000C49D 83ED30                 <1>       sub   ebp, ESPACE ; 48 ; total size of stack frame ('sysdefs.inc')
                                <1>                            ; for saving/restoring user registers
                                <1>       ;cmp   ebp, [u.usp]
                                <1>       ;je    short err0
0000C4A0 892D[60030300]         <1>       mov   [u.usp], ebp
                                <1> ;err0:
                                <1>       ; 01/09/2015
0000C4A6 8B25[60030300]         <1>       mov   esp, [u.usp]     ; Retro Unix 8086 v1 modification!
                                <1>                              ; 10/04/2013
                                <1>                              ; (If an I/O error occurs during disk I/O,
                                <1>                              ; related procedures will jump to 'error'
                                <1>                              ; procedure directly without returning to
                                <1>                              ; the caller procedure. So, stack pointer
                                <1>                                   ; must be restored here.)
                                <1>       ; 13/05/2016
                                <1>       ; NOTE: (The last) error code is in 'u.error', it can be retrieved by
                                <1>       ;     'get last error' system call later.
                                <1>
                                <1>       ; 03/09/2015 - 09/06/2015 - 07/08/2013
0000C4AC C605[C6030300]00       <1>       mov   byte [u.kcall], 0 ; namei_r, mkdir_w reset
                                <1>
                                <1> sysret: ; < return from system call>
                                <1>       ; 01/03/2017
                                <1>       ; 28/02/2017
                                <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; 16/04/2015 - 10/09/2015 (Retro UNIX 386 v1)
                                <1>       ; 10/04/2013 - 23/02/2014 (Retro UNIX 8086 v1)
                                <1>       ;
                                <1>       ; 'sysret' first checks to see if process is about to be
                                <1>       ; terminated (u.bsys). If it is, 'sysexit' is called.
                                <1>       ; If not, following happens:
                                <1>       ;     1) The user's stack pointer is restored.
                                <1>       ;     2) r1=0 and 'iget' is called to see if last mentioned
                                <1>       ;        i-node has been modified. If it has, it is written out
                                <1>       ;        via 'ppoke'.
                                <1>       ;     3) If the super block has been modified, it is written out
                                <1>       ;        via 'ppoke'.
                                <1>       ;     4) If the dismountable file system's super block has been
                                <1>       ;        modified, it is written out to the specified device
                                <1>       ;        via 'ppoke'.
                                <1>       ;     5) A check is made if user's time quantum (uquant) ran out
                                <1>       ;        during his execution. If so, 'tswap' is called to give
                                <1>       ;        another user a chance to run.
                                <1>       ;     6) 'sysret' now goes into 'sysrele'.
                                <1>       ;         (See 'sysrele' for conclusion.)
                                <1>       ;
                                <1>       ; Calling sequence:
                                <1>       ;     jump table or 'br sysret'
                                <1>       ; Arguments:
                                <1>       ;     -
                                <1>       ; ............................................................
                                <1>       ;
                                <1>       ; ((AX=r1 for 'iget' input))
                                <1>       ;
0000C4B3 31C0                   <1>       xor   eax, eax ; 28/02/2017
                                <1> sysret0: ; 29/07/2015 (eax = 0, jump from sysexec)
0000C4B5 FEC0                   <1>       inc   al ; 04/05/2013
0000C4B7 3805[B2030300]         <1>       cmp   [u.bsys], al ; 1
                                <1>             ; tstb u.bsys / is a process about to be terminated because
```

```
                                <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
0000C4C3 FEC8                   <1>       dec   al ; mov ax, 0
                                <1>             ; clr r1 / zero r1 to check last mentioned i-node
0000C4C5 E8CF2C0000             <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:
0000C4CA 803D[A8030300]00       <1>       cmp   byte [u.quant], 0 ; 16/05/2013
                                <1>             ; tstb uquant / is the time quantum 0?
0000C4D1 7705                   <1>         ja      short swapret
                                <1>             ; bne 1f / no, don't swap it out
                                <1> sysrelease: ; 07/12/2013 (jump from 'clock')
0000C4D3 E8AB210000             <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
0000C4D8 E8BD2C0000             <1>       call  isintr
                                <1>       ; 20/10/2013
0000C4DD 7405                   <1>       jz    short sysrel1
0000C4DF 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:
0000C4E4 FA                     <1>       cli   ; 14/10/2015
                                <1> sysrel2:
                                <1>       ; 28/02/2017
                                <1>       ; Check if there is a (delayed) callback for current user/process
0000C4E5 A0[D7030300]           <1>       mov   al, [u.irqwait]
0000C4EA 240F                   <1>       and   al, 0Fh ; is there a waiting IRQ callback service ?
0000C4EC 7444                   <1>       jz    short sysrel8 ; no
                                <1>
                                <1>       ; Set return to IRQ callback service and return from the service
0000C4EE 0FB6D8                 <1>       movzx  ebx, al
0000C4F1 883D[D7030300]         <1>       mov   [u.irqwait], bh ; 0 ; reset
0000C4F7 8A9B[E20F0100]         <1>       mov   bl, [ebx+IRQenum] ; (available) IRQ index +1 (1 to 9)
                                <1>       ; 01/03/2017
0000C4FD FECB                   <1>       dec   bl ; IRQ index number, 0 to 8
0000C4FF 7831                   <1>       js    short sysrel8 ; 0 -> FFh (not in use!?)
                                <1>       ;
0000C501 A0[B3030300]           <1>       mov   al, [u.uno] ; current process (user) number
0000C506 3883[36650100]         <1>       cmp   [ebx+IRQ.owner], al
0000C50C 7524                   <1>       jne   short sysrel8 ; it is not the current user/process !?
0000C50E F683[48650100]01       <1>       test  byte [ebx+IRQ.method], 1 ; callback ?
0000C515 741B                   <1>       jz    short sysrel8 ; not a callback method !?
                                <1>
0000C517 8B93[5A650100]         <1>       mov   edx, [ebx+IRQ.addr] ; IRQ callback service address (virtual)
0000C51D C605[D8030300]01       <1>       mov   byte [u.r_lock], 1 ; IRQ callback service in progress flag
                                <1>
0000C524 E802220000             <1>       call  wswap ; save user's registers & status
                                <1>                   ;      (for return from IRQ callback service)
                                <1>
0000C529 8B2D[5C030300]         <1>       mov   ebp, [u.sp]; kernel's stack, points to EIP (user)
0000C52F 895500                 <1>       mov   [ebp], edx ; IRQ call back service address
                                <1> sysrel8:
0000C532 FE0D[5B030300]         <1>       dec   byte [sysflg]
                                <1>             ; decb sysflg / turn system flag off
                                <1>
0000C538 A1[B8030300]           <1>       mov   eax, [u.pgdir]
0000C53D 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
0000C540 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>         ;
0000C541 A1[64030300]           <1>         mov    eax, [u.r0]  ; ((return value in EAX))
0000C546 0FA9                   <1>         pop    gs
0000C548 0FA1                   <1>         pop    fs
0000C54A 07                     <1>         pop    es
0000C54B 1F                     <1>         pop    ds
                                <1>         ;or    word [esp+8], 200h ; 22/01/2017 ; force enabling interrupts
0000C54C 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
0000C54D 803D[D4030300]00       <1>         cmp    byte [u.t_lock], 0 ; TIMER INT LOCK
                                <1>         ;je    short sysrel0 ; classic (Retro UNIX 386 type) sysrele
0000C554 7734                   <1>         ja     short sysrel3
                                <1>         ; 27/02/2017
0000C556 803D[D8030300]00       <1>         cmp    byte [u.r_lock], 0 ; IRQ callback lock
0000C55D 0F8667FFFFFF           <1>         jna    sysrel0 ; classic sysrele ; 24/03/2017
0000C563 E859000000             <1>         call   sysrel7
0000C568 803D[D8030300]00       <1>         cmp    byte [u.r_lock], 0 ; IRQ callback service lock
0000C56F 7628                   <1>         jna    short sysrel4
0000C571 C605[D8030300]00       <1>         mov    byte [u.r_lock], 0 ; reset
                                <1>         ;mov   byte [u.irqwait], 0 ; reset ; 28/02/2017
0000C578 A0[D9030300]           <1>         mov    al, [u.r_mode]
0000C57D 08C0                   <1>         or     al, al
0000C57F 7518                   <1>         jnz    short sysrel4
0000C581 FEC8                   <1>         dec    al
0000C583 A2[D9030300]           <1>         mov    [u.r_mode], al ; 0FFh ; not necessary !?
0000C588 EB32                   <1>         jmp    short sysrel6
                                <1> sysrel3:
                                <1>         ; 27/02/2017
0000C58A E832000000             <1>         call   sysrel7
                                <1>         ; 14/01/2017
0000C58F 28C0                   <1>         sub    al, al
0000C591 3805[D4030300]         <1>         cmp    [u.t_lock], al ; 0 ; TIMER INT LOCK
0000C597 770E                   <1>         ja     short sysrel5 ; yes
                                <1> sysrel4:
                                <1>         ; 29/01/2017
0000C599 8B44241C               <1>         mov    eax, [esp+28] ; eax
0000C59D A3[64030300]           <1>         mov    [u.r0], eax
0000C5A2 E93EFFFFFF             <1>         jmp    sysrel2
                                <1> sysrel5:
0000C5A7 A2[D4030300]           <1>         mov    [u.t_lock], al ; 0 ; reset
0000C5AC A0[D5030300]           <1>         mov    al, [u.t_mode]
0000C5B1 20C0                   <1>         and    al, al
                                <1>         ;jnz   short sysrel2 ; 0FFh ; user mode
0000C5B3 75E4                   <1>         jnz    short sysrel4 ; 29/01/2017
0000C5B5 FEC8                   <1>         dec    al
0000C5B7 A2[D5030300]           <1>         mov    [u.t_mode], al ; 0FFh ; not necessary !?
                                <1> sysrel6:
                                <1>         ; cpu will continue from the interrupted sytem call addr
0000C5BC 61                     <1>         popad          ; edi, esi, ebp, esp, ebx, edx, ecx, eax
0000C5BD 83C410                 <1>         add    esp, 16       ; pass segment segisters: ds, es, fs, gs
0000C5C0 CF                     <1>         iretd          ; eip, cs, eflags
                                <1>
                                <1> sysrel7:
0000C5C1 0FB61D[B3030300]       <1>         movzx  ebx, byte [u.uno] ; current process number
0000C5C8 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
0000C5CC 8B83[BC000300]         <1>         mov     eax, [ebx+p.upage-4] ; UPAGE address
0000C5D2 FA                     <1>         cli    ; disable interrupts till 'iretd'
0000C5D3 E98B210000             <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>         ;
0000C5D8 FE05[B2030300]         <1>         inc    byte [u.bsys]
                                <1>         ;
0000C5DE 8B1D[5C030300]         <1>         mov    ebx, [u.sp] ; esp at the beginning of 'sysent'
0000C5E4 8B03                   <1>         mov    eax, [ebx] ; EIP (return address, not 'INT 30h' address)
```

```
497 0000C5E6 83E802              <1>       sub    eax, 2 ; CDh, ##h
498 0000C5E9 E81B6DFFFF          <1>       call   dwordtohex
499 0000C5EE 8915[BB0D0100]      <1>       mov    [eip_str], edx
500 0000C5F4 A3[BF0D0100]        <1>       mov    [eip_str+4], eax
501 0000C5F9 A1[64030300]        <1>       mov    eax, [u.r0]
502 0000C5FE E8066DFFFF          <1>       call   dwordtohex
503 0000C603 8915[AA0D0100]      <1>       mov    [eax_str], edx
504 0000C609 A3[AE0D0100]        <1>       mov    [eax_str+4], eax
505                              <1>
506 0000C60E 66C705[9F0D0100]34- <1>       mov    word [int_num_str], SYSCALL_INT_NUM ; 25/12/2016
506 0000C616 30                  <1>
507                              <1>
508 0000C617 BE[710D0100]        <1>       mov    esi, ifc_msg ; "invalid funtion call !" msg (trdosk9.s)
509 0000C61C E83C9DFFFF          <1>       call   print_msg
510                              <1>
511 0000C621 EB17                <1>       jmp    sysexit
512                              <1>
513                              <1> intract: ; / interrupt action
514                              <1>       ; 14/10/2015
515                              <1>       ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
516                              <1>       ; 09/05/2013 - 07/12/2013 (Retro UNIX 8086 v1)
517                              <1>       ;
518                              <1>       ; Retro UNIX 8086 v1 modification !
519                              <1>       ; (Process/task switching and quit routine by using
520                              <1>       ; Retro UNIX 8086 v1 keyboard interrupt output.))
521                              <1>       ;
522                              <1>       ; input -> 'u.quit'  (also value of 'u.intr' > 0)
523                              <1>       ; output -> If value of 'u.quit' = FFFFh ('ctrl+brk' sign)
524                              <1>       ;            'intract' will jump to 'sysexit'.
525                              <1>       ;          Intract will return to the caller
526                              <1>       ;            if value of 'u.quit' <> FFFFh.
527                              <1>       ; 14/10/2015
528 0000C623 FB                  <1>       sti
529                              <1>       ; 07/12/2013
530 0000C624 66FF05[AC030300]    <1>       inc    word [u.quit]
531 0000C62B 7408                <1>       jz     short intrct0 ; FFFFh -> 0
532 0000C62D 66FF0D[AC030300]    <1>       dec    word [u.quit]
533                              <1>       ; 16/04/2015
534 0000C634 C3                  <1>       retn
535                              <1> intrct0:
536 0000C635 58                  <1>       pop    eax ; call intract -> retn
537                              <1>       ;
538 0000C636 31C0                <1>       xor    eax, eax
539 0000C638 FEC0                <1>       inc    al  ; mov ax, 1
540                              <1> ;;;
541                              <1>       ; UNIX v1 original 'intract' routine...
542                              <1>       ; / interrupt action
543                              <1>             ;cmp *(sp),$rti / are you in a clock interrupt?
544                              <1>             ; bne 1f / no, 1f
545                              <1>             ; cmp (sp)+,(sp)+ / pop clock pointer
546                              <1>       ; 1: / now in user area
547                              <1>             ; mov r1,-(sp) / save r1
548                              <1>             ; mov u.ttyp,r1
549                              <1>                   ; / pointer to tty buffer in control-to r1
550                              <1>             ; cmpb 6(r1),$177
551                              <1>                   ; / is the interrupt char equal to "del"
552                              <1>             ; beq 1f / yes, 1f
553                              <1>             ; clrb 6(r1)
554                              <1>                    ; / no, clear the byte
555                              <1>                   ; / (must be a quit character)
556                              <1>             ; mov (sp)+,r1 / restore r1
557                              <1>             ; clr u.quit / clear quit flag
558                              <1>             ; bis $20,2(sp)
559                              <1>                   ; / set trace for quit (sets t bit of
560                              <1>                   ; / ps-trace trap)
561                              <1>             ; rti   ;  / return from interrupt
562                              <1>       ; 1: / interrupt char = del
563                              <1>             ; clrb 6(r1) / clear the interrupt byte
564                              <1>                      ; / in the buffer
565                              <1>             ; mov (sp)+,r1 / restore r1
566                              <1>             ; cmp u.intr,$core / should control be
567                              <1>                         ; / transferred to loc core?
568                              <1>             ; blo 1f
569                              <1>             ; jmp *u.intr / user to do rti yes,
570                              <1>                         ; / transfer to loc core
571                              <1>       ; 1:
572                              <1>             ; sys 1 / exit
573                              <1>
574                              <1> sysexit: ; <terminate process>
575                              <1>       ; 27/05/2017
576                              <1>       ; 10/04/2017
577                              <1>       ; 28/02/2017
578                              <1>       ; 26/02/2017
579                              <1>       ; 02/01/2017, 23/01/2017
580                              <1>       ; 06/06/2016, 10/06/2016
581                              <1>       ; 19/05/2016, 23/05/2016
582                              <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
583                              <1>       ; 16/04/2015 - 01/09/2015 (Retro UNIX 386 v1)
584                              <1>       ; 19/04/2013 - 14/02/2014 (Retro UNIX 8086 v1)
585                              <1>       ;
586                              <1>       ; 'sysexit' terminates a process. First each file that
587                              <1>       ; the process has opened is closed by 'flose'. The process
588                              <1>       ; status is then set to unused. The 'p.pid' table is then
589                              <1>       ; searched to find children of the dying process. If any of
590                              <1>       ; children are zombies (died by not waited for), they are
591                              <1>       ; set free. The 'p.pid' table is then searched to find the
592                              <1>       ; dying process's parent. When the parent is found, it is
593                              <1>       ; checked to see if it is free or it is a zombie. If it is
594                              <1>       ; one of these, the dying process just dies. If it is waiting
595                              <1>       ; for a child process to die, it notified that it doesn't
596                              <1>       ; have to wait anymore by setting it's status from 2 to 1
597                              <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
0000C63A 6648                   <1>       dec    ax ; 0
0000C63C 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
0000C642 FA                     <1>       cli    ; disable interrupts
                                <1>       ; 23/01/2017 - reset timer frequency (to 18.2Hz)
0000C643 B036                   <1>       mov    al, 00110110b ; 36h
0000C645 E643                   <1>       out    43h, al
0000C647 28C0                   <1>       sub    al, al ; 0
0000C649 E640                   <1>       out    40h, al ; LB
0000C64B E640                   <1>       out    40h, al ; HB
                                <1>       ;
0000C64D 0FB61D[B3030300]       <1>       movzx  ebx, byte [u.uno]
                                <1>       ;mov   bl, [u.uno] ; process number of dying process
0000C654 3883[FF000300]         <1>       cmp    byte [ebx+p.timer-1], al ; 0
0000C65A 763B                   <1>       jna    short sysexit_12 ; no timer events for this process
0000C65C 8883[FF000300]         <1>       mov    byte [ebx+p.timer-1], al ; 0 ; reset
0000C662 A0[975F0100]           <1>       mov    al, [timer_events]
                                <1>       ;or    al, al
                                <1>       ;jz    short sysexit_12 ; no timer events
0000C667 88C1                   <1>       mov    cl, al
                                <1>       ;cli   ; disable interrupts
0000C669 B410                   <1>       mov    ah, 16 ; number of available timer events
0000C66B BE[60040300]           <1>       mov    esi, timer_set ; beginning address of timer events
                                <1> sysexit_7:
0000C670 8A06                   <1>       mov    al, [esi] ; process number (of timer event)
0000C672 38D8                   <1>       cmp    al, bl ; process number comparison
0000C674 7411                   <1>       je     short sysexit_10
0000C676 20C0                   <1>       and    al, al
0000C678 7404                   <1>       jz     short sysexit_9
                                <1> sysexit_8:
0000C67A FEC9                   <1>       dec    cl
0000C67C 7416                   <1>       jz     short sysexit_11
                                <1> sysexit_9:
0000C67E FECC                   <1>       dec    ah
0000C680 7415                   <1>       jz     short sysexit_12
0000C682 83C610                 <1>       add    esi, 16
0000C685 EBE9                   <1>       jmp    short sysexit_7
                                <1>
                                <1> sysexit_10:
                                <1>       ;mov   byte [esi], 0
0000C687 66C7060000             <1>       mov    word [esi], 0
                                <1>       ;mov   dword [esi+12], 0
                                <1>       ;
0000C68C FE0D[975F0100]         <1>       dec    byte [timer_events] ; 02/01/2017
                                <1>       ;
0000C692 EBE6                   <1>       jmp    short sysexit_8
                                <1>
                                <1> sysexit_11:
0000C694 6629C0                 <1>       sub    ax, ax ; 0 ; 26/02/2017
                                <1> sysexit_12:
                                <1>       ; 26/02/2017 (Unlink IRQ callbacks belong to the user)
0000C697 803D[D6030300]00       <1>       cmp    byte [u.irqc], 0 ; Count of IRQ callbacks
0000C69E 7E2E                   <1>       jng    short sysexit_16 ; zero or invalid
                                <1>       ; 28/02/2017
                                <1>       ; clear IRQ callback flags (for 'sysrele' and 'sysret')
0000C6A0 A2[D7030300]           <1>       mov    [u.irqwait], al ; 0 ; force to clear waiting flag
0000C6A5 A2[D8030300]           <1>       mov    [u.r_lock], al ; 0 ; force to clear busy flag
0000C6AA BE[36650100]           <1>       mov    esi, IRQ.owner
                                <1> sysexit_13:
0000C6AF AC                     <1>       lodsb
0000C6B0 3A05[B3030300]         <1>       cmp    al, [u.uno] ; owner  = current user ?
0000C6B6 750C                   <1>       jne    short sysexit_14
0000C6B8 C646FF00               <1>       mov    byte [esi-1], 0 ; owner = 0 : Free
0000C6BC FE0D[D6030300]         <1>       dec    byte [u.irqc]
0000C6C2 7408                   <1>       jz     short sysexit_15
```

```
                                <1> sysexit_14:
0000C6C4 81FE[3E650100]         <1>         cmp    esi, IRQ.owner + 8 ; the last IRQ index number ?
0000C6CA 76E3                   <1>         jna    short sysexit_13 ; no
                                <1> sysexit_15:
0000C6CC 30C0                   <1>         xor    al, al ; 0
                                <1> sysexit_16: ; 2:
0000C6CE 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
0000C6CF E8E5130000             <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
0000C6D4 FEC0                   <1>         inc    al
                                <1>                ; inc r1 / increment file descriptor
                                <1>         ;cmp   ax, 10
0000C6D6 3C0A                   <1>         cmp    al, 10
                                <1>                ; cmp r1,$10. / end of u.fp list?
0000C6D8 72F5                   <1>         jb     short sysexit_1
                                <1>                ; blt 1b / no, go back
                                <1>         ;movzx ebx, byte [u.uno]
0000C6DA 8A1D[B3030300]         <1>         mov    bl, [u.uno] ; 02/01/2017
                                <1>                ; movb u.uno,r1 / yes, move dying process's number to r1
0000C6E0 88A3[AF000300]         <1>         mov    [ebx+p.stat-1], ah ; 0, SFREE
                                <1>                ; clrb p.stat-1(r1) / free the process
                                <1>         ; 10/04/2017
0000C6E6 381D[AD650100]         <1>         cmp    [audio_user], bl
0000C6EC 7518                   <1>         jne    short sysexit_17
                                <1>         ; reset audio device (current) owner and 'initializated' flag
0000C6EE 883D[AD650100]         <1>         mov    [audio_user], bh ; 0
                                <1>         ; 27/05/2017
0000C6F4 8B0D[98650100]         <1>         mov    ecx,  [audio_buffer]
0000C6FA 09C9                   <1>         or     ecx, ecx
0000C6FC 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)
0000C6FE 29C9                   <1>         sub    ecx, ecx
0000C700 890D[98650100]         <1>         mov    [audio_buffer], ecx ; 0
                                <1>         ;pop   ebx
                                <1> sysexit_17:
                                <1>         ;shl   bx, 1
0000C706 D0E3                   <1>         shl    bl, 1
                                <1>                ; asl r1 / use r1 for index into the below tables
0000C708 668B8B[1E000300]       <1>         mov    cx, [ebx+p.pid-2]
                                <1>                ; mov p.pid-2(r1),r3 / move dying process's name to r3
0000C70F 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
0000C716 30DB                   <1>         xor    bl, bl ; 0
                                <1>                ; clr r2
0000C718 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
0000C71A 80C302                 <1>         add    bl, 2
                                <1>                ; add $2,r2 / search parent process table
                                <1>                            ; / for dying process's name
0000C71D 66398B[3E000300]       <1>         cmp    [ebx+p.ppid-2], cx
                                <1>                ; cmp p.ppid-2(r2),r3 / found it?
0000C724 7513                   <1>         jne    short sysexit_4
                                <1>                ; bne 3f / no
                                <1>         ;shr   bx, 1
0000C726 D0EB                   <1>         shr    bl, 1
                                <1>                ; asr r2 / yes, it is a parent
0000C728 80BB[AF000300]03       <1>         cmp    byte [ebx+p.stat-1], 3 ; SZOMB
                                <1>                ; cmpb p.stat-1(r2),$3 / is the child of this
                                <1>                                       ; / dying process a zombie
0000C72F 7506                   <1>         jne    short sysexit_3
                                <1>                ; bne 2f / no
0000C731 88A3[AF000300]         <1>         mov    [ebx+p.stat-1], ah ; 0, SFREE
                                <1>                ; clrb p.stat-1(r2) / yes, free the child process
                                <1> sysexit_3: ; 2:
                                <1>         ;shr   bx, 1
0000C737 D0E3                   <1>         shl    bl, 1
                                <1>                ; asl r2
                                <1> sysexit_4: ; 3:
                                <1>                ; / search the process name table
                                <1>                ; / for the dying process's parent
0000C739 663993[1E000300]       <1>         cmp    [ebx+p.pid-2], dx
                                <1>                ; cmp p.pid-2(r2),r4 / found it?
0000C740 7502                   <1>         jne    short sysexit_5
                                <1>                ; bne 3f / no
0000C742 89DE                   <1>         mov    esi, ebx
                                <1>                ; mov r2,r5 / yes, put index to p.pid table (parents
                                <1>                          ; / process # x2) in r5
                                <1> sysexit_5: ; 3:
                                <1>         ;cmp   bx, nproc + nproc
0000C744 80FB20                 <1>         cmp    bl, nproc + nproc
                                <1>                ; cmp r2,$nproc+nproc / has whole table been searched?
0000C747 72D1                   <1>         jb     short sysexit_2
                                <1>                ; blt 1b / no, go back
```

```
                                <1>             ; mov r5,r1 / yes, r1 now has parents process # x2
0000C749 21F6                   <1>       and   esi, esi ; r5=r1
0000C74B 7436                   <1>       jz    short sysexit_6
                                <1>             ; beq 2f / no parent has been found.
                                <1>                   ; / The process just dies
0000C74D 66D1EE                 <1>       shr   si, 1
                                <1>             ; asr r1 / set up index to p.stat
0000C750 8A86[AF000300]         <1>       mov   al, [esi+p.stat-1]
                                <1>             ; movb p.stat-1(r1),r2 / move status of parent to r2
0000C756 20C0                   <1>       and   al, al
0000C758 7429                   <1>       jz    short sysexit_6
                                <1>             ; beq 2f / if its been freed, 2f
0000C75A 3C03                   <1>       cmp   al, 3
                                <1>             ; cmp r2,$3 / is parent a zombie?
0000C75C 7425                   <1>       je    short sysexit_6
                                <1>             ; beq 2f / yes, 2f
                                <1>       ; BH = 0
0000C75E 8A1D[B3030300]         <1>       mov   bl, [u.uno]
                                <1>             ; movb u.uno,r3 / move dying process's number to r3
0000C764 C683[AF000300]03       <1>       mov   byte [ebx+p.stat-1], 3  ; SZOMB
                                <1>             ; movb $3,p.stat-1(r3) / make the process a zombie
0000C76B 3C01                   <1>       cmp   al, 1 ; SRUN
0000C76D 7414                   <1>       je    short sysexit_6
                                <1>       ;cmp  al, 2
                                <1>             ; cmp r2,$2 / is the parent waiting for
                                <1>                     ; / this child to die
                                <1>       ;jne  short sysexit_6
                                <1>             ; bne 2f / yes, notify parent not to wait any more
                                <1>       ; p.stat = 2 --> waiting
                                <1>       ; p.stat = 4 --> sleeping
0000C76F C686[AF000300]01       <1>       mov   byte [esi+p.stat-1], 1 ; SRUN
                                <1>       ;dec  byte [esi+p.stat-1]
                                <1>             ; decb p.stat-1(r1) / awaken it by putting it (parent)
0000C776 6689F0                 <1>       mov   ax, si ; r1  (process number in AL)
                                <1>       ;
                                <1>       ;mov  ebx, runq + 4
                                <1>             ; mov $runq+4,r2 / on the runq
0000C779 BB[54030300]           <1>       mov   ebx, runq+2 ; normal run queue ; 02/01/2017
0000C77E E818200000             <1>       call  putlu
                                <1>             ; jsr r0, putlu
                                <1> sysexit_6:
                                <1>             ; / the process dies
0000C783 C605[B3030300]00       <1>       mov   byte [u.uno], 0
                                <1>             ; clrb u.uno / put zero as the process number,
                                <1>                 ; / so "swap" will
0000C78A E80E1F0000             <1>       call  swap
                                <1>             ; jsr r0,swap / overwrite process with another process
                                <1>
                                <1> hlt_sys:
                                <1>       ;sti
                                <1> hlts0:
0000C78F F4                     <1>       hlt
0000C790 EBFD                   <1>       jmp   short hlts0
                                <1>             ; 0 / and thereby kill it; halt?
                                <1>
                                <1> syswait: ; < wait for a processs to die >
                                <1>       ; 17/09/2015
                                <1>       ; 02/09/2015
                                <1>       ; 01/09/2015
                                <1>       ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
                                <1>       ; 24/05/2013 - 05/02/2014 (Retro UNIX 8086 v1)
                                <1>       ;
                                <1>       ; 'syswait' waits for a process die.
                                <1>       ; It works in following way:
                                <1>       ;    1) From the parent process number, the parent's
                                <1>       ;      process name is found. The p.ppid table of parent
                                <1>       ;      names is then searched for this process name.
                                <1>       ;      If a match occurs, r2 contains child's process
                                <1>       ;      number. The child status is checked to see if it is
                                <1>       ;      a zombie, i.e; dead but not waited for (p.stat=3)
                                <1>       ;      If it is, the child process is freed and it's name
                                <1>       ;      is put in (u.r0). A return is then made via 'sysret'.
                                <1>       ;      If the child is not a zombie, nothing happens and
                                <1>       ;      the search goes on through the p.ppid table until
                                <1>       ;      all processes are checked or a zombie is found.
                                <1>       ;    2) If no zombies are found, a check is made to see if
                                <1>       ;      there are any children at all. If there are none,
                                <1>       ;      an error return is made. If there are, the parent's
                                <1>       ;      status is set to 2 (waiting for child to die),
                                <1>       ;      the parent is swapped out, and a branch to 'syswait'
                                <1>       ;      is made to wait on the next process.
                                <1>       ;
                                <1>       ; Calling sequence:
                                <1>       ;      ?
                                <1>       ; Arguments:
                                <1>       ;      -
                                <1>       ; Inputs: -
                                <1>       ; Outputs: if zombie found, it's name put in u.r0.
                                <1>       ; ............................................................
                                <1>       ;
                                <1>
                                <1> ; / wait for a process to die
                                <1>
                                <1> syswait_0:
0000C792 0FB61D[B3030300]       <1>       movzx  ebx, byte [u.uno] ; 01/09/2015
                                <1>             ; movb u.uno,r1 / put parents process number in r1
0000C799 D0E3                   <1>       shl   bl, 1
                                <1>       ;shl  bx, 1
                                <1>             ; asl r1 / x2 to get index into p.pid table
0000C79B 668B83[1E000300]       <1>       mov   ax, [ebx+p.pid-2]
                                <1>             ; mov p.pid-2(r1),r1 / get the name of this process
0000C7A2 31F6                   <1>       xor   esi, esi
```

```
                                <1>             ; clr r2
0000C7A4 31C9                   <1>       xor   ecx, ecx ; 30/10/2013
                                <1>       ;xor  cl, cl
                                <1>             ; clr r3 / initialize reg 3
                                <1> syswait_1: ; 1:
0000C7A6 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
0000C7AA 663B86[3E000300]       <1>       cmp   ax, [esi+p.ppid-2]
                                <1>             ; cmp p.ppid-2(r2),r1 / r2 will contain the childs
                                <1>                                   ; / process number
0000C7B1 7535                   <1>       jne   short syswait_3
                                <1>             ;bne 3f / branch if no match of parent process name
                                <1>       ;inc  cx
0000C7B3 FEC1                   <1>       inc   cl
                                <1>             ;inc r3 / yes, a match, r3 indicates number of children
0000C7B5 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).
0000C7B8 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?
0000C7BF 7524                   <1>       jne   short syswait_2
                                <1>             ; bne 2f / no, skip it
0000C7C1 88BE[AF000300]         <1>       mov   [esi+p.stat-1], bh ; 0
                                <1>             ; clrb p.stat-1(r2) / yes, free it
0000C7C7 66D1E6                 <1>       shl   si, 1
                                <1>             ; asl r2 / r2x2 to get index into p.pid table
0000C7CA 0FB786[1E000300]       <1>       movzx  eax, word [esi+p.pid-2]
0000C7D1 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>       ;
0000C7D6 6629C0                 <1>       sub   ax, ax
0000C7D9 668986[3E000300]       <1>       mov   [esi+p.ppid-2], ax ; 0 ; 17/09/2015
0000C7E0 E9D0FCFFFF             <1>       jmp   sysret0 ; ax = 0
                                <1>       ;
                                <1>       ;jmp  sysret
                                <1>             ; br sysret1 / return cause child is dead
                                <1> syswait_2: ; 2:
0000C7E5 66D1E6                 <1>       shl   si, 1
                                <1>             ; asl r2 / r2x2 to get index into p.ppid table
                                <1> syswait_3: ; 3:
0000C7E8 6683FE20               <1>       cmp   si, nproc+nproc
                                <1>             ; cmp r2,$nproc+nproc / have all processes been checked?
0000C7EC 72B8                   <1>       jb    short syswait_1
                                <1>             ; blt 1b / no, continue search
                                <1>       ;and  cx, cx
0000C7EE 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
0000C7F0 750B                   <1>       jnz   short syswait_4
                                <1>       ;jz   error
                                <1>             ; beq error1 / there are no children, error
0000C7F2 890D[64030300]         <1>       mov   [u.r0], ecx ; 0
0000C7F8 E996FCFFFF             <1>       jmp   error
                                <1> syswait_4:
0000C7FD 8A1D[B3030300]         <1>       mov   bl, [u.uno]
                                <1>             ; movb u.uno,r1 / there are children so put
                                <1>                           ; / parent process number in r1
0000C803 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
0000C809 E88F1E0000             <1>       call  swap
                                <1>             ; jsr r0,swap / swap it out, because it's waiting
0000C80E 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
```

```
1006                    <1>       ;     an error occurs.
1007                    <1>       ;    2) when one is found, it becomes the child process number
1008                    <1>       ;     and it's status (p.stat) is set to active.
1009                    <1>       ;    3) If the parent had a control tty, the interrupt
1010                    <1>       ;     character in that tty buffer is cleared.
1011                    <1>       ;    4) The child process is put on the lowest priority run
1012                    <1>       ;     queue via 'putlu'.
1013                    <1>       ;    5) A new process name is gotten from 'mpid' (actually
1014                    <1>       ;     it is a unique number) and is put in the child's unique
1015                    <1>       ;     identifier; process id (p.pid).
1016                    <1>       ;    6) The process name of the parent is then obtained and
1017                    <1>       ;     placed in the unique identifier of the parent process
1018                    <1>       ;     name is then put in 'u.r0'.
1019                    <1>       ;    7) The child process is then written out on disk by
1020                    <1>       ;     'wswap',i.e., the parent process is copied onto disk
1021                    <1>       ;     and the child is born. (The child process is written
1022                    <1>       ;     out on disk/drum with 'u.uno' being the child process
1023                    <1>       ;     number.)
1024                    <1>       ;    8) The parent process number is then restored to 'u.uno'.
1025                    <1>       ;    9) The child process name is put in 'u.r0'.
1026                    <1>       ;   10) The pc on the stack sp + 18 is incremented by 2 to
1027                    <1>       ;     create the return address for the parent process.
1028                    <1>       ;   11) The 'u.fp' list as then searched to see what files
1029                    <1>       ;     the parent has opened. For each file the parent has
1030                    <1>       ;     opened, the corresponding 'fsp' entry must be updated
1031                    <1>       ;     to indicate that the child process also has opened
1032                    <1>       ;     the file. A branch to 'sysret' is then made.

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

```
1107                                <1>       ; EBX = return address for child process
1108                                <1>           ; (Retro UNIX 8086 v1 modification !)
1109 0000C810 31F6                 <1>       xor    esi, esi
1110                                <1>              ; clr r1
1111                                <1> sysfork_1: ; 1: / search p.stat table for unused process number
1112 0000C812 46                   <1>       inc    esi
1113                                <1>              ; inc r1
1114 0000C813 80BE[AF000300]00     <1>       cmp    byte [esi+p.stat-1], 0 ; SFREE, 05/02/2014
1115                                <1>              ; tstb p.stat-1(r1) / is process active, unused, dead
1116 0000C81A 760B                 <1>       jna    short sysfork_2
1117                                <1>              ; beq 1f / it's unused so branch
1118 0000C81C 6683FE10             <1>       cmp    si, nproc
1119                                <1>              ; cmp r1,$nproc / all processes checked
1120 0000C820 72F0                 <1>       jb     short sysfork_1
1121                                <1>              ; blt 1b / no, branch back
1122                                <1>       ;
1123                                <1>       ; Retro UNIX 8086 v1. modification:
1124                                <1>       ;      Parent process returns from 'sysfork' to address
1125                                <1>       ;      which is just after 'sysfork' system call in parent
1126                                <1>       ;      process. Child process returns to address which is put
1127                                <1>       ;      in BX register by parent process for 'sysfork'.
1128                                <1>       ;
1129                                <1>              ;add $2,18.(sp) / add 2 to pc when trap occured, points
1130                                <1>                           ; / to old process return
1131                                <1>              ; br error1 / no room for a new process
1132 0000C822 E96CFCFFFF           <1>       jmp    error
1133                                <1> sysfork_2: ; 1:
1134 0000C827 E84D83FFFF           <1>       call   allocate_page
1135 0000C82C 0F8261FCFFFF         <1>       jc     error
1136 0000C832 50                   <1>       push   eax   ; UPAGE (user structure page) address
1137                                <1>       ; Retro UNIX 386 v1 modification!
1138 0000C833 E85085FFFF           <1>       call   duplicate_page_dir
1139                                <1>              ; EAX = New page directory
1140 0000C838 730B                 <1>       jnc    short sysfork_3
1141 0000C83A 58                   <1>       pop    eax   ; UPAGE (user structure page) address
1142 0000C83B E81785FFFF           <1>       call   deallocate_page
1143 0000C840 E94EFCFFFF           <1>       jmp    error
1144                                <1> sysfork_3:
1145                                <1>       ; Retro UNIX 386 v1 modification !
1146 0000C845 56                   <1>       push   esi
1147 0000C846 E8E01E0000           <1>       call   wswap ; save current user (u) structure, user registers
1148                                <1>                    ; and interrupt return components (for IRET)
1149 0000C84B 8705[B8030300]       <1>       xchg   eax, [u.pgdir] ; page directory of the child process
1150 0000C851 A3[BC030300]         <1>       mov    [u.ppgdir], eax ; page directory of the parent process
1151 0000C856 5E                   <1>       pop    esi
1152 0000C857 58                   <1>       pop    eax   ; UPAGE (user structure page) address
1153                                <1>              ; [u.usp] = esp
1154 0000C858 89F7                 <1>       mov    edi, esi
1155 0000C85A 66C1E702             <1>       shl    di, 2
1156 0000C85E 8987[BC000300]       <1>       mov    [edi+p.upage-4], eax ; memory page for 'user' struct
1157 0000C864 A3[B4030300]         <1>       mov    [u.upage], eax ; memory page for 'user' struct (child)
1158                                <1>       ; 28/08/2015
1159 0000C869 0FB605[B3030300]     <1>       movzx  eax, byte [u.uno] ; parent process number
1160                                <1>              ; movb u.uno,-(sp) / save parent process number
1161 0000C870 89C7                 <1>       mov    edi, eax
1162 0000C872 50                   <1>         push eax ; **
1163 0000C873 8A87[7F000300]       <1>       mov     al, [edi+p.ttyc-1] ; console tty (parent)
1164                                <1>       ; 18/09/2015
1165                                <1>       ;mov     [esi+p.ttyc-1], al ; set child's console tty
1166                                <1>       ;mov     [esi+p.waitc-1], ah ; 0 ; reset child's wait channel
1167 0000C879 668986[7F000300]     <1>       mov     [esi+p.ttyc-1], ax ; al - set child's console tty
1168                                <1>                                ; ah - reset child's wait channel
1169 0000C880 89F0                 <1>       mov    eax, esi
1170 0000C882 A2[B3030300]         <1>       mov    [u.uno], al ; child process number
1171                                <1>              ;movb r1,u.uno / set child process number to r1
1172 0000C887 FE86[AF000300]       <1>         inc     byte [esi+p.stat-1] ; 1, SRUN, 05/02/2014
1173                                <1>              ; incb p.stat-1(r1) / set p.stat entry for child
1174                                <1>                           ; / process to active status
1175                                <1>              ; mov u.ttyp,r2 / put pointer to parent process'
1176                                <1>                           ; / control tty buffer in r2
1177                                <1>                 ; beq 2f / branch, if no such tty assigned
1178                                <1>              ; clrb 6(r2) / clear interrupt character in tty buffer
1179                                <1>       ; 2:
1180 0000C88D 53                   <1>       push   ebx  ; * return address for the child process
1181                                <1>                    ; * Retro UNIX 8086 v1 feature only !
1182                                <1>       ; (Retro UNIX 8086 v1 modification!)
1183                                <1>              ; mov $runq+4,r2
1184 0000C88E BB[54030300]         <1>       mov    ebx, runq+2 ; normal run queue ; 02/01/2017
1185 0000C893 E8031F0000           <1>       call   putlu
1186                                <1>              ; jsr r0,putlu / put child process on lowest priority
1187                                <1>                       ; / run queue
1188 0000C898 66D1E6               <1>       shl    si, 1
1189                                <1>              ; asl r1 / multiply r1 by 2 to get index
1190                                <1>                     ; / into p.pid table
1191 0000C89B 66FF05[4E030300]     <1>       inc    word [mpid]
1192                                <1>              ; inc mpid / increment m.pid; get a new process name
1193 0000C8A2 66A1[4E030300]       <1>       mov    ax, [mpid]
1194 0000C8A8 668986[1E000300]     <1>       mov    [esi+p.pid-2], ax
1195                                <1>              ;mov mpid,p.pid-2(r1) / put new process name
1196                                <1>                                ; / in child process' name slot
1197 0000C8AF 5A                   <1>       pop    edx  ; * return address for the child process
1198                                <1>                    ; * Retro UNIX 8086 v1 feature only !
1199 0000C8B0 5B                   <1>       pop    ebx  ; **
1200                                <1>       ;mov   ebx, [esp] ; ** parent process number
1201                                <1>              ; movb (sp),r2 / put parent process number in r2
1202 0000C8B1 66D1E3               <1>       shl    bx, 1
1203                                <1>              ;asl r2 / multiply by 2 to get index into below tables
1204                                <1>       ;movzx eax, word [ebx+p.pid-2]
1205 0000C8B4 668B83[1E000300]     <1>       mov    ax, [ebx+p.pid-2]
1206                                <1>              ; mov p.pid-2(r2),r2 / get process name of parent
1207                                <1>                                ; / process
1208 0000C8BB 668986[3E000300]     <1>       mov    [esi+p.ppid-2], ax
```

```
1209                                    <1>             ; mov r2,p.ppid-2(r1) / put parent process name
1210                                    <1>                     ; / in parent process slot for child
1211 0000C8C2 A3[64030300]              <1>       mov   [u.r0], eax
1212                                    <1>             ; mov r2,*u.r0 / put parent process name on stack
1213                                    <1>                       ; / at location where r0 was saved
1214 0000C8C7 8B2D[5C030300]            <1>       mov   ebp, [u.sp] ; points to return address (EIP for IRET)
1215 0000C8CD 895500                    <1>       mov   [ebp], edx ; *, CS:EIP -> EIP
1216                                    <1>                       ; * return address for the child process
1217                                    <1>             ; mov $sysret1,-(sp) /
1218                                    <1>             ; mov sp,u.usp / contents of sp at the time when
1219                                    <1>                         ; / user is swapped out
1220                                    <1>             ; mov $sstack,sp / point sp to swapping stack space
1221                                    <1>       ; 04/09/2015 - 01/09/2015
1222                                    <1>       ; [u.usp] = esp
1223 0000C8D0 68[B3C40000]              <1>       push  sysret ; ***
1224 0000C8D5 8925[60030300]            <1>       mov   [u.usp], esp ; points to 'sysret' address (***)
1225                                    <1>                         ; (for child process)
1226 0000C8DB 31C0                      <1>       xor   eax, eax
1227 0000C8DD 66A3[94030300]            <1>       mov   [u.ttyp], ax ; 0
1228                                    <1>       ;
1229 0000C8E3 E8431E0000                <1>       call  wswap ; Retro UNIX 8086 v1 modification !
1230                                    <1>             ;jsr r0,wswap / put child process out on drum
1231                                    <1>             ;jsr r0,unpack / unpack user stack
1232                                    <1>             ;mov u.usp,sp / restore user stack pointer
1233                                    <1>             ; tst (sp)+ / bump stack pointer
1234                                    <1>       ; Retro UNIX 386 v1 modification !
1235 0000C8E8 58                        <1>       pop   eax ; ***
1236 0000C8E9 66D1E3                    <1>       shl   bx, 1
1237 0000C8EC 8B83[BC000300]            <1>       mov    eax, [ebx+p.upage-4] ; UPAGE address ; 14/05/2015
1238 0000C8F2 E86C1E0000                <1>       call  rswap ; restore parent process 'u' structure,
1239                                    <1>                   ; registers and return address (for IRET)
1240                                    <1>             ;movb (sp)+,u.uno / put parent process number in u.uno
1241 0000C8F7 0FB705[4E030300]          <1>         movzx   eax, word [mpid]
1242 0000C8FE A3[64030300]              <1>       mov   [u.r0], eax
1243                                    <1>             ; mov mpid,*u.r0 / put child process name on stack
1244                                    <1>                             ; / where r0 was saved
1245                                    <1>             ; add $2,18.(sp) / add 2 to pc on stack; gives parent
1246                                    <1>                               ; / process return
1247                                    <1>       ;xor   ebx, ebx
1248 0000C903 31F6                      <1>       xor    esi, esi
1249                                    <1>             ;clr r1
1250                                    <1> sysfork_4: ; 1: / search u.fp list to find the files
1251                                    <1>            ; / opened by the parent process
1252                                    <1>       ; 01/09/2015
1253                                    <1>       ;xor   bh, bh
1254                                    <1>       ;mov   bl, [esi+u.fp]
1255 0000C905 8A86[6A030300]            <1>       mov   al, [esi+u.fp]
1256                                    <1>             ; movb u.fp(r1),r2 / get an open file for this process
1257                                    <1>         ;or      bl, bl
1258 0000C90B 08C0                      <1>       or    al, al
1259 0000C90D 740D                      <1>       jz    short sysfork_5
1260                                    <1>             ; beq 2f / file has not been opened by parent,
1261                                    <1>                    ; / so branch
1262 0000C90F B40A                      <1>       mov   ah, 10 ; Retro UNIX 386 v1 fsp structure size = 10 bytes
1263 0000C911 F6E4                      <1>       mul   ah
1264                                    <1>       ;movzx ebx, ax
1265 0000C913 6689C3                    <1>       mov   bx, ax
1266                                    <1>       ;shl     bx, 3
1267                                    <1>             ; asl r2 / multiply by 8
1268                                    <1>                   ; asl r2 / to get index into fsp table
1269                                    <1>                   ; asl r2
1270 0000C916 FE83[4E010300]            <1>       inc    byte [ebx+fsp-2]
1271                                    <1>             ; incb fsp-2(r2) / increment number of processes
1272                                    <1>                         ; / using file, because child will now be
1273                                    <1>                         ; / using this file
1274                                    <1> sysfork_5: ; 2:
1275 0000C91C 46                        <1>         inc    esi
1276                                    <1>             ; inc r1 / get next open file
1277 0000C91D 6683FE0A                  <1>         cmp    si, 10
1278                                    <1>             ; cmp r1,$10. / 10. files is the maximum number which
1279                                    <1>                     ; / can be opened
1280 0000C921 72E2                      <1>       jb    short sysfork_4
1281                                    <1>             ; blt 1b / check next entry
1282 0000C923 E98BFBFFFF                <1>       jmp   sysret
1283                                    <1>             ; br sysret1
1284                                    <1>
1285                                    <1> syscreat: ; < create file >
1286                                    <1>       ; 27/10/2016
1287                                    <1>       ; 25/10/2016, 26/10/2016
1288                                    <1>       ; 15/10/2016, 16/10/2016, 17/10/2016
1289                                    <1>       ; 10/10/2016 (TRDOS 386 = TRDOS v2.0)
1290                                    <1>       ;          -derived from INT_21H.ASM-
1291                                    <1>       ;           ("loc_INT21h_create_file")
1292                                    <1>         ;   10/07/2011 (12/03/2011)
1293                                    <1>         ;   INT 21h Function AH = 3Ch
1294                                    <1>         ;   Create File
1295                                    <1>         ;   INPUT
1296                                    <1>         ;      CX = Attributes
1297                                    <1>         ;          DS:DX= Address of zero terminaned path name
1298                                    <1>         ;
1299                                    <1>       ; 27/12/2015 (Retro UNIX 386 v1.1)
1300                                    <1>       ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
1301                                    <1>       ; 27/05/2013 (Retro UNIX 8086 v1)
1302                                    <1>       ;
1303                                    <1>       ; 'syscreat' called with two arguments; name and mode.
1304                                    <1>       ; u.namep points to name of the file and mode is put
1305                                    <1>       ; on the stack. 'namei' is called to get i-number of the file.
1306                                    <1>       ; If the file aready exists, it's mode and owner remain
1307                                    <1>       ; unchanged, but it is truncated to zero length. If the file
1308                                    <1>       ; did not exist, an i-node is created with the new mode via
1309                                    <1>       ; 'maknod' whether or not the file already existed, it is
1310                                    <1>       ; open for writing. The fsp table is then searched for a free
```

```
1311                                    <1>       ; entry. When a free entry is found, proper data is placed
1312                                    <1>       ; in it and the number of this entry is put in the u.fp list.
1313                                    <1>       ; The index to the u.fp (also know as the file descriptor)
1314                                    <1>       ; is put in the user's r0.
1315                                    <1>       ;
1316                                    <1>       ; Calling sequence:
1317                                    <1>       ;     syscreate; name; mode
1318                                    <1>       ; Arguments:
1319                                    <1>       ;     name - name of the file to be created
1320                                    <1>       ;     mode - mode of the file to be created
1321                                    <1>       ; Inputs: (arguments)
1322                                    <1>       ; Outputs: *u.r0 - index to u.fp list
1323                                    <1>       ;              (the file descriptor of new file)
1324                                    <1>       ; ..........................................................
1325                                    <1>       ;
1326                                    <1>       ; Retro UNIX 8086 v1 modification:
1327                                    <1>       ;       'syscreate' system call has two arguments; so,
1328                                    <1>       ;     * 1st argument, name is pointed to by BX register
1329                                    <1>       ;     * 2nd argument, mode is in CX register
1330                                    <1>       ;
1331                                    <1>       ;     AX register (will be restored via 'u.r0') will return
1332                                    <1>       ;     to the user with the file descriptor/number
1333                                    <1>       ;     (index to u.fp list).
1334                                    <1>       ;
1335                                    <1>       ;call  arg2
1336                                    <1>       ; * name - 'u.namep' points to address of file/path name
1337                                    <1>       ;          in the user's program segment ('u.segmnt')
1338                                    <1>       ;          with offset in BX register (as sysopen argument 1).
1339                                    <1>       ; * mode - sysopen argument 2 is in CX register
1340                                    <1>       ;          which is on top of stack.
1341                                    <1>       ;
1342                                    <1>       ; TRDOS 386 (10/10/2016)
1343                                    <1>       ;
1344                                    <1>         ; INPUT ->
1345                                    <1>         ;      CL = File Attributes
1346                                    <1>       ;           bit 0 (1) - Read only file (R)
1347                                    <1>       ;            bit 1 (1) - Hidden file (H)
1348                                    <1>         ;            bit 2 (1) - System file (R)
1349                                    <1>       ;            bit 3 (1) - Volume label/name (V)
1350                                    <1>         ;            bit 4 (1) - Subdirectory (D)
1351                                    <1>       ;           bit 5 (1) - File has been archived (A)
1352                                    <1>         ;          EBX = Pointer to filename (ASCIIZ) -path-
1353                                    <1>       ;
1354                                    <1>       ; OUTPUT ->
1355                                    <1>       ;          eax = File/Device Handle/Number (index) (AL)
1356                                    <1>       ;          cf = 1 -> Error code in AL
1357                                    <1>       ;
1358                                    <1>       ; Modified Registers: EAX (at the return of system call)
1359                                    <1>       ;
1360                                    <1>       ; Note: If the file is existing and it has not any one
1361                                    <1>       ;     of S,H,R,V,D attributes, it will be truncated
1362                                    <1>       ;     to zero length; otherwise, access error will be
1363                                    <1>       ;     returned.
1364                                    <1>
1365                                    <1> sysmkdir_0:
1366 0000C928 F6C108                   <1>       test   cl, 08h ; Volume name
1367 0000C92B 740A                     <1>       jz     short syscreat_0
1368                                    <1>
1369                                    <1>       ; Volume name or long name creation
1370                                    <1>       ; is not permitted (in TRDOS 386)!
1371 0000C92D B80B000000               <1>       mov    eax, ERR_FILE_ACCESS  ; 11 ; 'permission denied !'
1372 0000C932 E927020000               <1>         jmp  sysopen_dev_err
1373                                    <1>
1374                                    <1> syscreat_0:
1375                                    <1>         ;mov [u.namep], ebx
1376 0000C937 51                       <1>       push   ecx
1377 0000C938 89DE                     <1>       mov    esi, ebx
1378                                    <1>       ; file name is forced, change directory as temporary
1379                                    <1>       ;mov   ax, 1
1380                                    <1>       ;mov   [FFF_Valid], ah ; 0 ; reset ; 17/10/2016
1381                                    <1>       ;call  set_working_path
1382 0000C93A E8962C0000               <1>       call   set_working_path_x ; 17/10/2016
1383 0000C93F 0F82D8000000             <1>       jc     syscreat_err
1384                                    <1>
1385                                    <1>       ; 16/10/2016
1386 0000C945 803D[BB5F0100]00         <1>       cmp    byte [SWP_inv_fname], 0
1387 0000C94C 776D                     <1>       ja     short  syscreat_inv_fname ; invalid file name !
1388                                    <1>
1389                                    <1>       ; Here, we have a valid path and also a valid file name
1390                                    <1>       ; (Working dir has been changed if the path
1391                                    <1>       ;  -file name string- had contained a dir name.)
1392                                    <1>
1393 0000C94E 6631C0                   <1>       xor    ax, ax
1394                                    <1>       ;mov   esi, FindFile_Name
1395 0000C951 E899B6FFFF               <1>       call   find_first_file
1396 0000C956 59                       <1>       pop    ecx
1397                                    <1>              ; ESI = Directory Entry (FindFile_DirEntry) Location
1398                                    <1>              ; EDI = Directory Buffer Directory Entry Location
1399                                    <1>              ; EAX = File Size
1400                                    <1>              ;  BL = Attributes of The File/Directory
1401                                    <1>              ;  BH = Long Name Yes/No Status (>0 is YES)
1402                                    <1>              ;  DX > 0 : Ambiguous filename chars are used
1403 0000C957 726A                     <1>       jc     short syscreat_1 ; file not found (the good!)
1404                                    <1>                             ; or another error (the bad')
1405                                    <1>
1406                                    <1>       ; (& the uggly!) truncate file to zero length before open
1407                                    <1>
1408                                    <1>       ;'*' and '?' already checked at 'set_working_path' stage
1409                                    <1>       ;and   dx, dx
1410                                    <1>       ;jnz   short sysmkdir_err ; permission denied
1411                                    <1>                               ; invalid filename chars
1412                                    <1>
```

```
1413                                    <1>      ;test  cl, 10h ; subdirectory ?
1414                                    <1>      ;jnz   short sysmkdir_err
1415                                    <1>
1416                                    <1>      ; BL = File Attributes:
1417                                    <1>      ;           bit 0 (1) - Read only file (R)
1418                                    <1>      ;             bit 1 (1) - Hidden file (H)
1419                                    <1>        ;             bit 2 (1) - System file (R)
1420                                    <1>      ;             bit 3 (1) - Volume label/name (V)
1421                                    <1>        ;             bit 4 (1) - Subdirectory (D)
1422                                    <1>      ;           bit 5 (1) - File has been archived
1423                                    <1>
1424                                    <1>      ; * existing directory must not be truncated
1425                                    <1>      ;   (we don't know it is empty or not, at this stage)
1426                                    <1>      ; * existing volume name (or a long name) can not be
1427                                    <1>      ;   re-created or truncated by 'syscreat'
1428                                    <1>      ; * A file with S, H, R attributes must not be truncated
1429                                    <1>      ;   (change attributes to normal, if you need truncate it)
1430                                    <1>
1431 0000C959 F6C31F                   <1>      test   bl, 00011111b  ; check attributes of existing file
1432 0000C95C 754F                     <1>      jnz    short sysmkdir_err
1433                                    <1>
1434                                    <1>      ;; normal file, OK to continue...
1435                                    <1>
1436                                    <1>      ; ESI = FindFile_DirEntry
1437 0000C95E 668B4614                 <1>      mov    ax, [esi+DirEntry_FstClusHI] ; 20
1438 0000C962 66C1E010                 <1>      shl    ax, 16
1439 0000C966 668B461A                 <1>      mov    ax, [esi+DirEntry_FstClusLO] ; 26
1440                                    <1>      ; EAX = First cluster to be truncated/unlinked
1441 0000C96A 57                       <1>      push   edi
1442 0000C96B 51                       <1>      push   ecx
1443 0000C96C BE00010900               <1>      mov    esi, Logical_DOSDisks
1444 0000C971 29C9                     <1>      sub    ecx, ecx
1445 0000C973 8A2D[C6520100]           <1>      mov    ch, [Current_Drv]
1446 0000C979 01CE                     <1>      add    esi, ecx
1447                                    <1>      ; ESI = Logical dos drive description table address
1448 0000C97B E8C7F7FFFF               <1>      call   truncate_cluster_chain
1449 0000C980 59                       <1>      pop    ecx
1450 0000C981 5F                       <1>      pop    edi
1451 0000C982 7230                     <1>      jc     short syscreate_truncate_err
1452                                    <1>
1453                                    <1>      ; 26/10/2016
1454                                    <1>      ; EDI = Directory entry address in directory buffer
1455                                    <1>      ; Update directory entry
1456 0000C984 E843DCFFFF               <1>      call   convert_current_date_time
1457                                    <1>      ; OUTPUT -> DX = Date in dos dir entry format
1458                                    <1>        ;       AX = Time in dos dir entry format
1459 0000C989 66894716                 <1>      mov    [edi+DirEntry_WrtTime], ax
1460 0000C98D 66895718                 <1>      mov    [edi+DirEntry_WrtDate], dx
1461 0000C991 66895712                 <1>      mov    [edi+DirEntry_LastAccDate], dx
1462 0000C995 31C0                     <1>      xor    eax, eax ; file size = 0
1463 0000C997 89471C                   <1>      mov    [edi+DirEntry_FileSize], eax ; 0
1464 0000C99A C605[F05A0100]02         <1>      mov    byte [DirBuff_ValidData], 2 ; data changed sign
1465 0000C9A1 BE[BC5C0100]             <1>      mov    esi, FindFile_DirEntry
1466 0000C9A6 B201                     <1>      mov    dl, 1 ; open file for writing
1467 0000C9A8 E9AA000000               <1>      jmp    sysopen_2
1468                                    <1>
1469                                    <1> sysmkdir_err:
1470                                    <1>      ; 1 = write, 2 = read & write, >2 = invalid
1471 0000C9AD B80B000000               <1>        mov  eax, ERR_FILE_ACCESS  ; 11 ; 'permission denied !'
1472 0000C9B2 EB73                     <1>        jmp  short sysopen_err
1473                                    <1>
1474                                    <1> syscreate_truncate_err:
1475 0000C9B4 B812000000               <1>      mov    eax, ERR_DRV_WRITE ; 18 ; 'disk write error !'
1476 0000C9B9 EB6C                     <1>        jmp  short sysopen_err
1477                                    <1>
1478                                    <1> syscreat_inv_fname:  ; invalid file name chars
1479                                    <1>      ; 16/10/2016
1480 0000C9BB B81A000000               <1>      mov    eax, ERR_INV_FILE_NAME  ; 26 ; invalid file name chars
1481 0000C9C0 59                       <1>      pop    ecx
1482 0000C9C1 EB64                     <1>      jmp    sysopen_err
1483                                    <1>
1484                                    <1> syscreat_1:
1485                                    <1>      ; Error code in EAX
1486 0000C9C3 3C02                     <1>        cmp  al, 02h ; 'File not found' error
1487 0000C9C5 7560                     <1>        jne  sysopen_err
1488                                    <1>
1489 0000C9C7 F6C110                   <1>      test   cl, 10h ; Directory
1490 0000C9CA 0F852C020000             <1>      jnz    sysmkdir_2
1491                                    <1>
1492                                    <1> syscreat_2:
1493 0000C9D0 BE[AC5C0100]             <1>      mov    esi, FindFile_Name
1494                                    <1>        ;xor edx, edx
1495 0000C9D5 31C0                     <1>        xor  eax, eax ; File Size  = 0
1496 0000C9D7 31DB                     <1>      xor    ebx, ebx
1497 0000C9D9 4B                       <1>      dec    ebx ; FFFFFFFFh -> create empty file
1498                                    <1>                  ;             (only for FAT fs)
1499                                    <1>      ; CL = File Attributes
1500 0000C9DA E8F6EBFFFF               <1>      call   create_file
1501 0000C9DF 7246                     <1>      jc     sysopen_err
1502                                    <1>             ; EAX = New file's first cluster
1503                                    <1>             ; ESI = Logical Dos Drv Descr. Table Addr.
1504                                    <1>             ; EBX = offset CreateFile_Size
1505                                    <1>             ; ECX = Sectors per cluster (<256)
1506                                    <1>             ; EDX = Directory entry index/number (<65536)
1507                                    <1>      ; 26/10/2016
1508                                    <1>      ;mov   esi, Directory_Buffer
1509                                    <1>      ;shl   dx, 5 ; *32
1510                                    <1>      ;add   esi, edx
1511                                    <1>      ;; esi = directory entry address in directory buffer
1512                                    <1>
1513                                    <1>      ; Here, directory entry has been created but last
1514                                    <1>      ; modification date & time of the parent dir has not
```

```
                                <1>       ; been updated, yet!
                                <1>       ; (Note: Directory and FAT buffers have been updated...)
                                <1>
0000C9E1 E81FDDFFFF             <1>       call   update_parent_dir_lmdt ; now, it is OK too!
                                <1>
                                <1>       ; 25/10/2016
0000C9E6 66B80018               <1>       mov    ax, 1800h
0000C9EA BE[AC5C0100]           <1>       mov    esi, FindFile_Name
0000C9EF E8FBB5FFFF             <1>       call   find_first_file
0000C9F4 7231                   <1>       jc     short sysopen_err
                                <1>
                                <1>       ; Only possible error after here is
                                <1>       ; "too many open files !" error.
                                <1>       ;
                                <1>       ; If "syscreat" will return with that error,
                                <1>       ; (the file has been created but it could not be opened)
                                <1>       ; the user must retry to open this file again
                                <1>       ; or must close another file before using
                                <1>       ; "sysopen" system call.
                                <1>
0000C9F6 B201                   <1>       mov    dl, 1 ; open file for writing
                                <1>       ; ESI = Directory Entry (FindFile_DirEntry) Location
                                <1>       ; EAX = File Size (= 0)
0000C9F8 EB5D                   <1>       jmp    short sysopen_2
                                <1>
                                <1> sysopen: ;<open file>
                                <1>       ; 26/10/2016
                                <1>       ; 24/10/2016
                                <1>       ; 17/10/2016
                                <1>       ; 15/10/2016
                                <1>       ; 06/10/2016, 07/10/2016, 08/10/2016
                                <1>       ; 05/10/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>       ;          -derived from INT_21H.ASM-
                                <1>       ;           ("loc_INT21h_open_file")
                                <1>         ;   26/02/2011
                                <1>         ;   INT 21h Function AH = 3Dh
                                <1>         ;   Open File
                                <1>         ;   INPUT
                                <1>         ;      AL= File Access Value
                                <1>       ;          0- Open for reading
                                <1>       ;           1- Open for writing
                                <1>         ;          2- Open for reading and writing
                                <1>         ;          DS:DX= Pointer to filename (ASCIIZ)
                                <1>         ;
                                <1>       ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
                                <1>       ; 22/05/2013 - 27/05/2013 (Retro UNIX 8086 v1)
                                <1>       ;
                                <1>       ; 'sysopen' opens a file in following manner:
                                <1>       ;   1) The second argument in a sysopen says whether to
                                <1>       ;     open the file ro read (0) or write (>0).
                                <1>       ;   2) I-node of the particular file is obtained via 'namei'.
                                <1>       ;   3) The file is opened by 'iopen'.
                                <1>       ;   4) Next housekeeping is performed on the fsp table
                                <1>       ;     and the user's open file list - u.fp.
                                <1>       ;     a) u.fp and fsp are scanned for the next available slot.
                                <1>       ;     b) An entry for the file is created in the fsp table.
                                <1>       ;     c) The number of this entry is put on u.fp list.
                                <1>       ;     d) The file descriptor index to u.fp list is pointed
                                <1>       ;        to by u.r0.
                                <1>       ;
                                <1>       ; Calling sequence:
                                <1>       ;     sysopen; name; mode
                                <1>       ; Arguments:
                                <1>       ;     name - file name or path name
                                <1>       ;     mode - 0 to open for reading
                                <1>       ;            1 to open for writing
                                <1>       ; Inputs: (arguments)
                                <1>       ; Outputs: *u.r0 - index to u.fp list (the file descriptor)
                                <1>       ;             is put into r0's location on the stack.
                                <1>       ; ..........................................................
                                <1>       ;
                                <1>       ; Retro UNIX 8086 v1 modification:
                                <1>       ;       'sysopen' system call has two arguments; so,
                                <1>       ;     * 1st argument, name is pointed to by BX register
                                <1>       ;     * 2nd argument, mode is in CX register
                                <1>       ;
                                <1>       ;     AX register (will be restored via 'u.r0') will return
                                <1>       ;     to the user with the file descriptor/number
                                <1>       ;     (index to u.fp list).
                                <1>       ;
                                <1>       ;call  arg2
                                <1>       ; * name - 'u.namep' points to address of file/path name
                                <1>       ;          in the user's program segment ('u.segmnt')
                                <1>       ;          with offset in BX register (as sysopen argument 1).
                                <1>       ; * mode - sysopen argument 2 is in CX register
                                <1>       ;          which is on top of stack.
                                <1>       ;
                                <1>       ; jsr r0,arg2 / get sys args into u.namep and on stack
                                <1>       ;
                                <1>           ; system call registers: ebx, ecx (through 'sysenter')
                                <1>       ;
                                <1>       ; TRDOS 386 (05/10/2016)
                                <1>       ;
                                <1>         ; INPUT ->
                                <1>         ;      CL = File Access Value (Open Mode)
                                <1>       ;           0 - Open file for reading
                                <1>       ;            1 - Open file for writing
                                <1>         ;            2 - Open device for reading
                                <1>       ;           3 - Open device for writing
                                <1>         ;          EBX = Pointer to filename/devicename (ASCIIZ)
                                <1>       ; OUTPUT ->
                                <1>       ;          eax = File/Device Handle/Number (index) (AL)
```

```
1617                                    <1>       ;          cf = 1 -> Error code in AL
1618                                    <1>       ;
1619                                    <1>       ; Modified Registers: EAX (at the return of system call)
1620                                    <1>       ;
1621                                    <1>
1622 0000C9FA 80F901                  <1>       cmp    cl, 1 ; read file (0), write file (1)
1623 0000C9FD 7614                    <1>       jna    short sysopen_0
1624                                    <1>
1625 0000C9FF 80F903                  <1>       cmp    cl, 3
1626 0000CA02 0F8640010000            <1>       jna    sysopen_device
1627                                    <1>
1628                                    <1>       ; Invalid access code
1629 0000CA08 B817000000              <1>       mov    eax, ERR_INV_PARAMETER
1630 0000CA0D 0F874B010000            <1>       ja     sysopen_dev_err
1631                                    <1>
1632                                    <1> sysopen_0:
1633                                    <1>       ;mov   [u.namep], ebx
1634 0000CA13 51                      <1>       push   ecx
1635 0000CA14 89DE                    <1>       mov    esi, ebx
1636                                    <1>       ; file name is forced, change directory as temporary
1637                                    <1>       ;mov   ax, 1
1638                                    <1>       ;mov   [FFF_Valid], ah ; 0 ; reset ; 17/10/2016
1639                                    <1>       ;call  set_working_path
1640 0000CA16 E8BA2B0000              <1>       call   set_working_path_x ; 17/10/2016
1641 0000CA1B 731E                    <1>       jnc    short sysopen_1
1642                                    <1>
1643                                    <1> syscreat_err: ; ecx = file attributes (for 'syscreat')
1644 0000CA1D 59                      <1>       pop    ecx ; open mode
1645 0000CA1E 21C0                    <1>       and    eax, eax  ; 0 -> Bad Path!
1646 0000CA20 7505                    <1>       jnz    short sysopen_err
1647                                    <1>       ; eax = 0
1648 0000CA22 B80C000000              <1>       mov    eax, ERR_DIR_NOT_FOUND ; Directory not found !
1649                                    <1> sysopen_err:
1650 0000CA27 A3[64030300]            <1>       mov    [u.r0], eax
1651 0000CA2C A3[C8030300]            <1>       mov    [u.error], eax
1652 0000CA31 E8742C0000              <1>       call   reset_working_path
1653 0000CA36 E958FAFFFF              <1>       jmp    error
1654                                    <1>
1655                                    <1> sysopen_1:
1656                                    <1>       ;mov   esi, FindFile_Name
1657 0000CA3B 66B80018                <1>         mov  ax, 1800h ; Only files
1658 0000CA3F E8ABB5FFFF              <1>       call   find_first_file
1659 0000CA44 5A                      <1>       pop    edx
1660 0000CA45 72E0                    <1>       jc     short sysopen_err ; eax = 2 (File not found !)
1661                                    <1>
1662                                    <1>       ; check_open_file_attr_access_code
1663                                    <1>
1664 0000CA47 F6C307                  <1>         test bl, 7  ; system, hidden, readonly
1665 0000CA4A 740B                    <1>         jz   short sysopen_2
1666                                    <1>
1667 0000CA4C 20D2                    <1>       and    dl, dl ; 0 = read mode
1668 0000CA4E 7407                    <1>       jz     short sysopen_2
1669                                    <1>
1670                                    <1>       ; 1 = write, 2 = read & write, >2 = invalid
1671 0000CA50 B80B000000              <1>         mov  eax, ERR_FILE_ACCESS  ; 11  = 'permission denied !'
1672 0000CA55 EBD0                    <1>         jmp  short sysopen_err
1673                                    <1>
1674                                    <1> sysopen_2:
1675                                    <1>       ; esi = Directory Entry (FindFile_DirEntry) Location
1676 0000CA57 89F3                    <1>       mov    ebx, esi
1677 0000CA59 31F6                    <1>         xor    esi, esi ; 0
1678 0000CA5B 31FF                    <1>         xor    edi, edi ; 0
1679                                    <1> sysopen_3: ; scan the list of entries in fsp table
1680 0000CA5D 80BE[6A030300]00        <1>         cmp    byte [esi+u.fp], 0
1681 0000CA64 760F                    <1>         jna    short sysopen_4 ; empty slot
1682 0000CA66 6646                    <1>         inc    si
1683 0000CA68 6683FE0A                <1>         cmp    si, 10
1684 0000CA6C 72EF                    <1>       jb     short sysopen_3
1685                                    <1> toomanyf:
1686 0000CA6E B80D000000              <1>       mov    eax, ERR_TOO_MANY_FILES ; too many open files !
1687 0000CA73 EBB2                    <1>       jmp    short sysopen_err
1688                                    <1>
1689                                    <1> sysopen_4:
1690 0000CA75 80BF[2A630100]00        <1>         cmp    byte [edi+OF_MODE], 0 ; Scan open files table
1691 0000CA7C 760A                    <1>       jna    short sysopen_5
1692 0000CA7E 6647                    <1>       inc    di
1693 0000CA80 6683FF0A                <1>       cmp    di, OPENFILES ; max. number of open files (=10)
1694 0000CA84 72EF                    <1>       jb     short sysopen_4
1695 0000CA86 EBE6                    <1>       jmp    short toomanyf
1696                                    <1>
1697                                    <1> sysopen_5:
1698 0000CA88 FEC2                    <1>       inc    dl
1699 0000CA8A 8897[2A630100]          <1>         mov    [edi+OF_MODE], dl
1700 0000CA90 8A15[6A5C0100]          <1>       mov    dl, [FindFile_Drv]
1701 0000CA96 8897[20630100]          <1>         mov    [edi+OF_DRIVE], dl ; Logical DOS drive number
1702 0000CA9C 66C1E702                <1>       shl    di, 2 ; *4 (dword offset)
1703                                    <1>
1704 0000CAA0 8987[70630100]          <1>       mov    [edi+OF_SIZE], eax ; File size in bytes
1705                                    <1>
1706 0000CAA6 668B4314                <1>         mov  ax, [ebx+DirEntry_FstClusHI]
1707 0000CAAA C1E010                  <1>       shl    eax, 16
1708 0000CAAD 668B431A                <1>       mov    ax, [ebx+DirEntry_FstClusLO]
1709 0000CAB1 8987[F8620100]          <1>       mov    [edi+OF_FCLUSTER], eax ; First cluster
1710 0000CAB7 8987[10640100]          <1>       mov    [edi+OF_CCLUSTER], eax ; Current cluster
1711                                    <1>
1712 0000CABD 31DB                    <1>         xor  ebx, ebx
1713 0000CABF 899F[48630100]          <1>         mov    [edi+OF_POINTER], ebx ; offset pointer (0)
1714 0000CAC5 899F[38640100]          <1>         mov    [edi+OF_CCINDEX], ebx ; cluster index (0)
1715                                    <1>
1716 0000CACB A1[DC5C0100]            <1>       mov    eax, [FindFile_DirFirstCluster]
1717 0000CAD0 8987[98630100]          <1>       mov    [edi+OF_DIRFCLUSTER], eax
1718                                    <1>
```

```
1719 0000CAD6 A1[E05C0100]        <1>       mov    eax, [FindFile_DirCluster]
1720 0000CADB 8987[C0630100]      <1>       mov    [edi+OF_DIRCLUSTER], eax
1721                              <1>
1722                              <1>       ; Get (& Save) Volume ID
1723                              <1>       ; Important for files of removable drives
1724                              <1>       ; (In order to check the drive has same volume/disk)
1725 0000CAE1 88D7                <1>       mov    bh, dl
1726 0000CAE3 81C300010900        <1>         add  ebx, Logical_DOSDisks
1727 0000CAE9 8A4303              <1>         mov  al, [ebx+LD_FATType]
1728 0000CAEC 3C01                <1>         cmp  al, 1
1729 0000CAEE 7209                <1>         jb   short sysopen_6_fs
1730 0000CAF0 3C02                <1>         cmp  al, 2
1731 0000CAF2 770A                <1>         ja   short sysopen_6_fat32
1732                              <1> sysopen_6_fat:
1733 0000CAF4 8B432D              <1>         mov  eax, [ebx+LD_BPB+VolumeID]
1734 0000CAF7 EB08                <1>         jmp  short sysopen_7
1735                              <1> sysopen_6_fs:
1736 0000CAF9 8B4328              <1>         mov  eax, [ebx+LD_FS_VolumeSerial]
1737 0000CAFC EB03                <1>         jmp  short sysopen_7
1738                              <1> sysopen_6_fat32:
1739 0000CAFE 8B4349              <1>         mov  eax, [ebx+LD_BPB+FAT32_VolID]
1740                              <1> sysopen_7:
1741 0000CB01 A3[BC520100]        <1>         mov  [Current_VolSerial], eax
1742                              <1>
1743 0000CB06 8987[E8630100]      <1>       mov    [edi+OF_VOLUMEID], eax
1744                              <1>
1745                              <1>       ; 24/10/2016
1746 0000CB0C 66D1EF              <1>       shr    di, 1 ; 4/2, word offset
1747 0000CB0F 668B1D[E45C0100]    <1>       mov    bx, [FindFile_DirEntryNumber]
1748 0000CB16 66899F[60640100]    <1>       mov    [edi+OF_DIRENTRY], bx
1749                              <1>
1750 0000CB1D 31D2                <1>       xor    edx, edx
1751                              <1>       ;shr   di, 2 ; /4 (byte offset)
1752 0000CB1F 66D1EF              <1>       shr    di, 1 ; 2/2, byte offset
1753 0000CB22 8897[3E630100]      <1>       mov    byte [edi+OF_OPENCOUNT], dl ; 0
1754 0000CB28 8897[34630100]      <1>       mov    byte [edi+OF_STATUS], dl ; 0
1755                              <1>
1756 0000CB2E 89FB                <1>       mov    ebx, edi
1757 0000CB30 FEC3                <1>       inc    bl
1758                              <1>
1759 0000CB32 889E[6A030300]      <1>         mov     [esi+u.fp], bl ; Open File Entry Number
1760 0000CB38 8935[64030300]      <1>       mov     [u.r0], esi ; move index to u.fp list
1761                              <1>                       ; into eax on stack
1762                              <1>
1763 0000CB3E E8672B0000          <1>         call       reset_working_path
1764                              <1>
1765 0000CB43 E96BF9FFFF          <1>       jmp    sysret
1766                              <1>
1767                              <1>       ; (Retro UNIX 386 v1.0)
1768                              <1>       ; 'fsp' table (10 bytes/entry)
1769                              <1>       ; bit 15                              bit 0
1770                              <1>       ; ---|----------------------------------------
1771                              <1>       ; r/w|       i-number of open file
1772                              <1>       ; ---|----------------------------------------
1773                              <1>       ;               device number
1774                              <1>       ; ----------------------------------------------
1775                              <1>       ; offset pointer, r/w pointer to file (bit 0-15)
1776                              <1>       ; ----------------------------------------------
1777                              <1>       ; offset pointer, r/w pointer to file (bit 16-31)
1778                              <1>       ; ----------------------|-----------------------
1779                              <1>       ;  flag that says file    | number of processes
1780                              <1>       ;   has been deleted      | that have file open
1781                              <1>       ; ----------------------|-----------------------
1782                              <1>
1783                              <1> sysopen_device:
1784                              <1>       ; 15/10/2016
1785                              <1>       ; 08/10/2016
1786                              <1>       ; 07/10/2016 (TRDOS 386 = TRDOS v2.0)
1787 0000CB48 51                  <1>       push   ecx ; open mode
1788 0000CB49 89E5                <1>       mov    ebp, esp
1789 0000CB4B B910000000          <1>       mov    ecx, 16 ; transfer length = 16 bytes
1790 0000CB50 29CC                <1>       sub    esp, ecx
1791 0000CB52 89E7                <1>       mov    edi, esp ; destination address
1792 0000CB54 89DE                <1>       mov    esi, ebx ; dev name in user's memory space
1793 0000CB56 E8751D0000          <1>       call   transfer_from_user_buffer
1794 0000CB5B 7310                <1>       jnc    short sysopen_dev_0
1795                              <1>       ; eax = ERR_OUT_OF_MEMORY = 4 = ERR_MINOR_IM
1796 0000CB5D 59                  <1>       pop    ecx
1797                              <1> sysopen_dev_err:
1798 0000CB5E A3[64030300]        <1>       mov    [u.r0], eax
1799 0000CB63 A3[C8030300]        <1>       mov    [u.error], eax
1800 0000CB68 E926F9FFFF          <1>       jmp    error
1801                              <1> sysopen_dev_0:
1802 0000CB6D 89FE                <1>       mov    esi, edi ; Device name addr (max. 16 bytes, ASCIIZ)
1803                              <1>                       ; for example: "tty, TTY, /dev/tty"
1804 0000CB6F E8DE2D0000          <1>       call   get_device_number
1805 0000CB74 89EC                <1>       mov    esp, ebp
1806 0000CB76 59                  <1>       pop    ecx
1807 0000CB77 7307                <1>       jnc    short sysopen_dev_1
1808 0000CB79 B818000000          <1>       mov    eax, ERR_INV_DEV_NAME ; 24 ; 'invalid device name !'
1809 0000CB7E EBDE                <1>       jmp    short sysopen_dev_err
1810                              <1> sysopen_dev_1:
1811                              <1>       ; eax = Device Number (AL)
1812                              <1>       ;  cl = Open mode (2 = device read, 3 = device write)
1813 0000CB80 31DB                <1>         xor     ebx, ebx ; 0
1814                              <1> sysopen_dev_2: ; scan the list of entries
1815 0000CB82 389B[6A030300]      <1>         cmp     [ebx+u.fp], bl ; 0
1816 0000CB88 760E                <1>         jna     short sysopen_dev_3 ; empty slot
1817 0000CB8A FEC3                <1>         inc     bl
1818 0000CB8C 80FB0A              <1>         cmp     bl, 10
1819 0000CB8F 72F1                <1>       jb     short sysopen_dev_2
1820                              <1>       ;
```

```
1821 0000CB91 B80D000000          <1>       mov    eax, ERR_TOO_MANY_FILES ; too many open files !
1822 0000CB96 EBC6                <1>       jmp    short sysopen_dev_err
1823                              <1> sysopen_dev_3:
1824 0000CB98 891D[64030300]      <1>       mov    [u.r0], ebx ; File/Device index/handle/descriptor
1825                              <1>       ; eax = device number (entry offset)
1826 0000CB9E 8AA8[BC600100]      <1>       mov    ch, [eax+DEV_ACCESS] ; bit 0 = accessable by users
1827                              <1>                                   ; bit 1 = read access perm
1828                              <1>                                   ; bit 2 = write access perm
1829                              <1>                                   ; bit 3 = IOCTL permit to users
1830                              <1>                                   ; bit 4 = block device if set
1831                              <1>                                   ; bit 5 = 16 bit or 1024 byte
1832                              <1>                                   ; bit 6 = 32 bit or 2048 byte
1833                              <1>                                   ; bit 7 = installable device drv
1834 0000CBA4 F6C501              <1>       test   ch, 1 ; accessable by normal users (except root)
1835 0000CBA7 7510                <1>       jnz    short sysopen_dev_4 ; yes, permission has been given
1836 0000CBA9 803D[B0030300]00    <1>       cmp    byte [u.uid], 0 ; root?
1837 0000CBB0 7607                <1>       jna    short sysopen_dev_4 ; superuser can open all devices
1838                              <1> sysopen_dev_perm_err:
1839 0000CBB2 B80B000000          <1>       mov    eax, ERR_DEV_ACCESS  ; 11  = 'permission denied !'
1840 0000CBB7 EBA5                <1>       jmp    short sysopen_dev_err
1841                              <1> sysopen_dev_4:
1842 0000CBB9 D0ED                <1>       shr    ch, 1 ; result: 1 = read, 2 = write, 3 = r & w
1843 0000CBBB FEC9                <1>       dec    cl  ; result: 1 = read, 2 = write
1844 0000CBBD 84E9                <1>       test   cl, ch
1845 0000CBBF 74F1                <1>       jz     short sysopen_dev_perm_err
1846                              <1>
1847 0000CBC1 D0E5                <1>       shl    ch, 1 ; bit 0 = 0
1848                              <1>       ; eax = device number (entry offset)
1849 0000CBC3 E8A62E0000          <1>       call   device_open
1850 0000CBC8 72E8                <1>       jc     short sysopen_dev_perm_err
1851                              <1>
1852                              <1>       ; eax = device number (entry offset)
1853 0000CBCA 0C80                <1>       or     al, 80h ; set device bit (set bit 7 to 1)
1854 0000CBCC 8B1D[64030300]      <1>       mov    ebx, [u.r0]
1855 0000CBD2 8883[6A030300]      <1>       mov    [ebx+u.fp], al      ; bit 7 (=1) points to device
1856                              <1>
1857 0000CBD8 E9D6F8FFFF          <1>       jmp    sysret
1858                              <1>
1859                              <1> sysmkdir: ; < make directory >
1860                              <1>       ; 15/10/2016
1861                              <1>       ; 10/10/2016 (TRDOS 386 = TRDOS v2.0)
1862                              <1>       ;          -derived from INT_21H.ASM-
1863                              <1>       ;            ("loc_INT21h_create_file")
1864                              <1>         ;    10/07/2011 (12/03/2011)
1865                              <1>         ;    INT 21h Function AH = 3Ch
1866                              <1>         ;    Create File
1867                              <1>         ;    INPUT
1868                              <1>         ;       CX = Attributes
1869                              <1>         ;           DS:DX= Address of zero terminaned path name
1870                              <1>         ;
1871                              <1>         ;
1872                              <1>       ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
1873                              <1>       ; 27/05/2013 - 02/08/2013 (Retro UNIX 8086 v1)
1874                              <1>       ;
1875                              <1>       ; 'sysmkdir' creates an empty directory whose name is
1876                              <1>       ; pointed to by arg 1. The mode of the directory is arg 2.
1877                              <1>       ; The special entries '.' and '..' are not present.
1878                              <1>       ; Errors are indicated if the directory already exists or
1879                              <1>       ; user is not the super user.
1880                              <1>       ;
1881                              <1>       ; Calling sequence:
1882                              <1>       ;     sysmkdir; name; mode
1883                              <1>       ; Arguments:
1884                              <1>       ;     name - points to the name of the directory
1885                              <1>       ;     mode - mode of the directory
1886                              <1>       ; Inputs: (arguments)
1887                              <1>       ; Outputs: -
1888                              <1>       ;    (sets 'directory' flag to 1;
1889                              <1>       ;    'set user id on execution' and 'executable' flags to 0)
1890                              <1>       ; ..........................................................
1891                              <1>       ;
1892                              <1>       ; Retro UNIX 8086 v1 modification:
1893                              <1>       ;       'sysmkdir' system call has two arguments; so,
1894                              <1>       ;     * 1st argument, name is pointed to by BX register
1895                              <1>       ;     * 2nd argument, mode is in CX register
1896                              <1>       ;
1897                              <1>       ; TRDOS 386 (10/10/2016)
1898                              <1>       ;
1899                              <1>         ; INPUT ->
1900                              <1>         ;      CL = Directory Attributes
1901                              <1>       ;           bit 0 (1) - Read only file/dir (R)
1902                              <1>       ;            bit 1 (1) - Hidden file/dir (H)
1903                              <1>         ;           bit 2 (1) - System file/dir (R)
1904                              <1>       ;            bit 3 (1) - Volume label/name (V)
1905                              <1>         ;           bit 4 (1) - Subdirectory (D)
1906                              <1>       ;           bit 5 (1) - File/Dir has been archived (A)
1907                              <1>       ;        CX = 0 -> create normal directory
1908                              <1>         ;         EBX = Pointer to directory name (ASCIIZ) -path-
1909                              <1>       ;
1910                              <1>       ; OUTPUT ->
1911                              <1>       ;         eax = First cluster of the new directory
1912                              <1>       ;         cf = 1 -> Error code in AL
1913                              <1>       ;
1914                              <1>       ; Modified Registers: EAX (at the return of system call)
1915                              <1>       ;
1916                              <1>       ; Note: If the file or directory is existing
1917                              <1>       ;     an access error will be returned.
1918                              <1>
1919 0000CBDD 6621C9              <1>       and    cx, cx ; if cx = 0 -> create a normal subdir
1920 0000CBE0 7413                <1>       jz     short sysmkdir_1
1921                              <1>
1922 0000CBE2 F6C110              <1>       test   cl, 10h ; if dir flags set, also use other flags
```

```
1923 0000CBE5 0F853DFDFFFF        <1>       jnz    sysmkdir_0 ; jump to head of 'syscreat'
1924                              <1>
1925                              <1>       ; CX has wrong flags
1926 0000CBEB B817000000          <1>       mov    eax, ERR_INV_FLAGS
1927 0000CBF0 E969FFFFFF          <1>       jmp    sysopen_dev_err
1928                              <1>
1929                              <1> sysmkdir_1:
1930 0000CBF5 B110                <1>       mov    cl, 10h ; set subdir flag and reset other flags
1931 0000CBF7 E92CFDFFFF          <1>       jmp    sysmkdir_0 ; jump to head of 'syscreat'
1932                              <1> sysmkdir_2:
1933                              <1>       ; jump from 'syscreat' ; from 'syscreat_1'
1934                              <1>       ;  CL = Directory attributes/flags
1935 0000CBFC BE[AC5C0100]        <1>       mov    esi, FindFile_Name
1936 0000CC01 E8FFD6FFFF          <1>       call   make_sub_directory
1937 0000CC06 0F821BFEFFFF        <1>       jc     sysopen_err       ; NOTE: Old type (TRDOS 8086)
1938                              <1>                                ; error codes must be modified
1939                              <1>                                ; for next TRDOS 386 versions
1940                              <1>                                ; (10/10/2016)
1941                              <1>                                ; Old (MSDOS type)
1942                              <1>                                ; error codes (2011):
1943                              <1>                                ;  2 = file not found
1944                              <1>                                ;  3 = directory not found
1945                              <1>                                ;  5 = access denied
1946                              <1>                                ; 12 = no more files
1947                              <1>                                  ; 19 = disk write protected
1948                              <1>                                ; 39 = insufficient disk space
1949                              <1>                                ; 'sysdefs.s' ; 10/10/2016
1950                              <1>
1951 0000CC0C A3[64030300]        <1>       mov    [u.r0], eax ; New sub dir's first cluster
1952                              <1>
1953 0000CC11 E8942A0000          <1>         call       reset_working_path
1954                              <1>
1955 0000CC16 E998F8FFFF          <1>       jmp    sysret
1956                              <1>
1957                              <1> sysclose: ;<close file>
1958                              <1>       ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
1959                              <1>       ;
1960                              <1>       ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
1961                              <1>       ; 22/05/2013 - 26/05/2013 (Retro UNIX 8086 v1)
1962                              <1>       ;
1963                              <1>       ; 'sysclose', given a file descriptor in 'u.r0', closes the
1964                              <1>       ; associated file. The file descriptor (index to 'u.fp' list)
1965                              <1>       ; is put in r1 and 'fclose' is called.
1966                              <1>       ;
1967                              <1>       ; Calling sequence:
1968                              <1>       ;     sysclose
1969                              <1>       ; Arguments:
1970                              <1>       ;     -
1971                              <1>       ; Inputs: *u.r0 - file descriptor
1972                              <1>       ; Outputs: -
1973                              <1>       ; ..........................................................
1974                              <1>       ;
1975                              <1>       ; Retro UNIX 8086 v1 modification:
1976                              <1>       ;      The user/application program puts file descriptor
1977                              <1>       ;       in BX register as 'sysclose' system call argument.
1978                              <1>       ;      (argument transfer method 1)
1979                              <1>
1980                              <1>       ; TRDOS 386 (06/10/2016)
1981                              <1>       ;
1982                              <1>         ; INPUT ->
1983                              <1>         ;      EBX = File Handle/Number (file index) (AL)
1984                              <1>       ; OUTPUT ->
1985                              <1>       ;         cf = 0 -> EAX = 0
1986                              <1>       ;         cf = 1 -> Error code in EAX (ERR_FILE_NOT_OPEN)
1987                              <1>       ;
1988                              <1>       ; Modified Registers: EAX (at the return of system call)
1989                              <1>       ;
1990                              <1>
1991 0000CC1B 89D8                <1>       mov    eax, ebx
1992 0000CC1D 31DB                <1>       xor    ebx, ebx
1993 0000CC1F 891D[64030300]      <1>       mov    [u.r0], ebx ; 0  ; return value of EAX
1994 0000CC25 E88F0E0000          <1>       call   fclose
1995 0000CC2A 0F8383F8FFFF        <1>       jnc    sysret
1996 0000CC30 B80A000000          <1>       mov    eax, ERR_FILE_NOT_OPEN ; file not open !
1997 0000CC35 A3[C8030300]        <1>       mov    [u.error], eax ;
1998 0000CC3A A3[64030300]        <1>       mov    [u.r0], eax ; ! invalid handle !
1999 0000CC3F E94FF8FFFF          <1>       jmp    error
2000                              <1>
2001                              <1> sysread: ; < read from file >
2002                              <1>       ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
2003                              <1>       ;          -derived from INT_21H.ASM-
2004                              <1>       ;           ("loc_INT21h_read_file")
2005                              <1>         ;   13/03/2011 (05/03/2011)
2006                              <1>         ;   INT 21h Function AH = 3Fh
2007                              <1>         ;   Read from a File
2008                              <1>         ;   INPUT
2009                              <1>       ;        BX = File Handle
2010                              <1>         ;      CX = Number of bytes to read
2011                              <1>         ;          DS:DX= Buffer address
2012                              <1>         ;
2013                              <1>       ; Note: TRDOS 386 'sysread' has been derived from
2014                              <1>       ;     Retro UNIX 386 v1 'sysread', except a few
2015                              <1>       ;     code modifications.
2016                              <1>       ;
2017                              <1>       ; 13/05/2015 (Retro UNIX 386 v1)
2018                              <1>       ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
2019                              <1>       ; 23/05/2013 (Retro UNIX 8086 v1)
2020                              <1>       ;
2021                              <1>       ; 'sysread' is given a buffer to read into and the number of
2022                              <1>       ; characters to be read. If finds the file from the file
2023                              <1>       ; descriptor located in *u.r0 (r0). This file descriptor
2024                              <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
0000CC44 E8BE0E0000             <1>       call  getf1
0000CC49 7277                   <1>       jc    short device_read ; read data from device
                                <1>       ; EAX = First cluster of the file
                                <1>
0000CC4B E83F000000             <1>       call  rw1
0000CC50 730A                   <1>       jnc   short sysread_0
                                <1>
0000CC52 A3[64030300]           <1>       mov   [u.r0], eax ; error code
0000CC57 E937F8FFFF             <1>       jmp   error
                                <1>
                                <1> sysread_0:
0000CC5C E84C170000             <1>       call  readi
0000CC61 EB1D                   <1>       jmp   short rw0
                                <1>
                                <1> syswrite: ; < write to file >
                                <1>       ; 23/10/2016
                                <1>       ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>       ;          -derived from INT_21H.ASM-
                                <1>       ;           ("loc_INT21h_write_file")
                                <1>         ;   13/03/2011 (05/03/2011)
                                <1>         ;   INT 21h Function AH = 40h
                                <1>         ;   Write to a File
                                <1>         ;   INPUT
                                <1>       ;        BX = File Handle
                                <1>         ;      CX = Number of bytes to write
                                <1>         ;          DS:DX= Buffer address
                                <1>         ;
                                <1>       ; Note: TRDOS 386 'sysrwrite' has been derived from
                                <1>       ;     Retro UNIX 386 v1 'syswrite', except a few
                                <1>       ;     code modifications.
                                <1>       ;
                                <1>
                                <1>       ; 13/05/2015 (Retro UNIX 386 v1)
                                <1>       ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
                                <1>       ; 23/05/2013 (Retro UNIX 8086 v1)
                                <1>       ;
                                <1>       ; 'syswrite' is given a buffer to write onto an output file
                                <1>       ; and the number of characters to write. If finds the file
                                <1>       ; from the file descriptor located in *u.r0 (r0). This file
                                <1>       ; descriptor is returned from a successful open or create call
                                <1>       ; (sysopen or syscreat). The i-number of file is obtained via
                                <1>       ; 'rw1' and buffer is written on the output file via 'write'.
                                <1>       ;
                                <1>       ; Calling sequence:
                                <1>       ;     syswrite; buffer; nchars
                                <1>       ; Arguments:
                                <1>       ;     buffer - location of contiguous bytes to be writtten.
                                <1>       ;     nchars - number of characters to be written.
                                <1>       ; Inputs: *u.r0 - file descriptor (& arguments)
                                <1>       ; Outputs: *u.r0 - number of bytes written.
                                <1>       ; ..........................................................
                                <1>       ;
                                <1>       ; Retro UNIX 8086 v1 modification:
                                <1>       ;       'syswrite' system call has three arguments; so,
                                <1>       ;     * 1st argument, file descriptor is in BX register
                                <1>       ;     * 2nd argument, buffer address/offset in CX register
                                <1>       ;     * 3rd argument, number of bytes is in DX register
                                <1>       ;
                                <1>       ;     AX register (will be restored via 'u.r0') will return
                                <1>       ;     to the user with number of bytes written.
                                <1>       ;
                                <1>       ; INPUT ->
                                <1>         ;       EBX = File handle (descriptor/index)
                                <1>       ;         ECX = Buffer address
                                <1>         ;           EDX = Number of bytes
                                <1>       ; OUTPUT ->
```

```
2127                                    <1>       ;          EAX = Number of bytes have been written
2128                                    <1>       ;          cf = 1 -> Error code in AL
2129                                    <1>       ;
2130                                    <1>       ; Modified Registers: EAX (at the return of system call)
2131                                    <1>       ;
2132                                    <1>
2133                                    <1>       ; EBX = File descriptor
2134 0000CC63 E89F0E0000                <1>       call   getf1
2135 0000CC68 7274                      <1>       jc     short device_write ; write data to device
2136                                    <1>       ; EAX = First cluster of the file
2137                                    <1>       ; EBX = File number  (Open file number) ; 23/10/2016
2138                                    <1>
2139 0000CC6A E820000000                <1>       call   rw1
2140 0000CC6F 730A                      <1>       jnc    short syswrite_0
2141 0000CC71 A3[64030300]              <1>       mov    [u.r0], eax ; error code
2142 0000CC76 E918F8FFFF                <1>       jmp    error
2143                                    <1>
2144                                    <1> syswrite_0:
2145 0000CC7B E8681E0000                <1>       call   writei
2146                                    <1> rw0: ; 1:
2147 0000CC80 A1[8C030300]              <1>         mov  eax, [u.nread]
2148 0000CC85 A3[64030300]              <1>       mov    [u.r0], eax
2149 0000CC8A E924F8FFFF                <1>       jmp    sysret
2150                                    <1>
2151                                    <1> rw1:
2152                                    <1>       ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
2153                                    <1>       ; 14/05/2015 (Retro UNIX 386 v1)
2154                                    <1>       ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
2155                                    <1>       ; 23/05/2013 - 24/05/2013 (Retro UNIX 8086 v1)
2156                                    <1>       ; System call registers: ebx, ecx, edx (through 'sysenter')
2157                                    <1>       ;
2158                                    <1>       ; EBX = File descriptor
2159                                    <1>       ;call  getf1 ; calling point in 'getf' from 'rw1'
2160                                    <1>       ;jc    short device_rw ; read/write data from/to device
2161                                    <1>       ; EAX = First cluster of the file
2162                                    <1>
2163 0000CC8F 83F802                    <1>       cmp    eax, 2
2164 0000CC92 7217                      <1>       jb     short rw2
2165                                    <1>       ;
2166 0000CC94 890D[84030300]            <1>       mov    [u.base], ecx       ; buffer address/offset
2167                                    <1>                                  ;(in the user's virtual memory space)
2168 0000CC9A 8915[88030300]            <1>       mov    [u.count], edx
2169                                    <1>
2170 0000CCA0 C705[C8030300]0000-       <1>         mov     dword [u.error], 0 ; reset the last error code
2170 0000CCA8 0000                      <1>
2171 0000CCAA C3                        <1>       retn
2172                                    <1>
2173                                    <1> rw2:
2174 0000CCAB B80A000000                <1>       mov    eax, ERR_FILE_NOT_OPEN ; file not open !
2175 0000CCB0 A3[C8030300]              <1>       mov    dword [u.error], eax
2176 0000CCB5 C3                        <1>       retn
2177                                    <1> rw3:
2178 0000CCB6 B80B000000                <1>       mov    eax, ERR_FILE_ACCESS ; permission denied !
2179 0000CCBB A3[C8030300]              <1>       mov    dword [u.error], eax
2180 0000CCC0 F9                        <1>       stc
2181 0000CCC1 C3                        <1>       retn
2182                                    <1>
2183                                    <1> device_read:
2184                                    <1>       ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
2185                                    <1>       ; cl = DEV_OPENMODE ; open mode
2186                                    <1>       ; ch = DEV_ACCESS   ; access flags
2187                                    <1>       ; al = DEV_DRIVER   ; device number (eax)
2188                                    <1>
2189 0000CCC2 F6C101                    <1>       test   cl, 1 ; 1 = read, 2 = write, 3 = read&write
2190 0000CCC5 74EF                      <1>       jz     short rw3
2191                                    <1>
2192 0000CCC7 89C3                      <1>       mov    ebx, eax
2193 0000CCC9 66C1E302                  <1>       shl    bx, 2 ; *4
2194                                    <1>
2195 0000CCCD F6C580                    <1>       test   ch, 80h ; bit 7, installable device driver flag
2196 0000CCD0 7406                      <1>       jz     short d_read_2 ; Kernel device
2197                                    <1>       ; installable device
2198                                    <1> d_read_1:
2199 0000CCD2 FFA3[78600100]            <1>         jmp  dword [ebx+IDEV_RADDR-4]
2200                                    <1> d_read_2:
2201 0000CCD8 FFA3[2A0F0100]            <1>       jmp    dword [ebx+KDEV_RADDR-4]
2202                                    <1>
2203                                    <1> device_write:
2204                                    <1>       ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
2205                                    <1>       ; cl = DEV_OPENMODE ; open mode
2206                                    <1>       ; ch = DEV_ACCESS   ; access flags
2207                                    <1>       ; al = DEV_DRIVER   ; device number (eax)
2208                                    <1>
2209 0000CCDE F6C102                    <1>       test   cl, 2 ; 1 = read, 2 = write, 3 = read&write
2210 0000CCE1 74D3                      <1>       jz     short rw3
2211                                    <1>
2212 0000CCE3 89C3                      <1>       mov    ebx, eax
2213 0000CCE5 66C1E302                  <1>       shl    bx, 2 ; *4
2214                                    <1>
2215 0000CCE9 F6C580                    <1>       test   ch, 80h ; bit 7, installable device driver flag
2216 0000CCEC 7406                      <1>       jz     short d_write_2 ; Kernel device
2217                                    <1>       ; installable device
2218                                    <1> d_write_1:
2219 0000CCEE FFA3[98600100]            <1>         jmp  dword [ebx+IDEV_WADDR-4]
2220                                    <1> d_write_2:
2221 0000CCF4 FFA3[7A0F0100]            <1>       jmp    dword [ebx+KDEV_WADDR-4]
2222                                    <1>
2223                                    <1>
2224                                    <1> sysemt: ; enable (or disable) multi tasking -time sharing-
2225                                    <1>       ;
2226                                    <1>       ; 23/05/2016 - TRDOS 386 (TRDOS v2.0)
2227                                    <1>       ; 14/05/2015 (Retro UNIX 386 v1)
```

```
2228                                    <1>       ; 10/12/2013 - 20/04/2014 (Retro UNIX 8086 v1)
2229                                    <1>       ;
2230                                    <1>       ; Retro UNIX 8086 v1 modification:
2231                                    <1>       ;     'Enable Multi Tasking'  system call instead
2232                                    <1>       ;     of 'Emulator Trap' in original UNIX v1 for PDP-11.
2233                                    <1>       ;
2234                                    <1>       ; Retro UNIX 8086 v1 feature only!
2235                                    <1>       ;     Using purpose: Kernel will start without time-out
2236                                    <1>       ;     (internal clock/timer) functionality.
2237                                    <1>       ;     Then etc/init will enable clock/timer for
2238                                    <1>       ;     multi tasking.
2239                                    <1>       ;
2240                                    <1>       ; INPUT ->
2241                                    <1>       ;     BL = 0 -> disable multi tasking
2242                                    <1>       ;     BL > 1 -> enable multi tasking (time sharing)
2243                                    <1>       ; OUTPUT ->
2244                                    <1>       ;     none
2245                                    <1>       ;
2246                                    <1>       ;  Note: Multi tasking is disabled during system
2247                                    <1>       ;       initialization, it must be enabled by using
2248                                    <1>       ;       this system call. (Otherwise, running proces
2249                                    <1>       ;       will not be changed by another process within
2250                                    <1>       ;       run time sequence/schedule, if running process
2251                                    <1>       ;       will not 'release' itself. Only 'wakeup' procedure
2252                                    <1>       ;       for waiting processes and programmed timer events
2253                                    <1>       ;       for other processes can change running process
2254                                    <1>       ;       while multi tasking is disabled.) ** 23/05/2016 **
2255                                    <1>
2256 0000CCFA 803D[B0030300]00          <1>       cmp    byte [u.uid], 0 ; root ?
2257                                    <1>       ;ja    error
2258 0000CD01 0F87D1F8FFFF             <1>       ja     badsys ; 14/05/2015
2259                                    <1>       ;
2260 0000CD07 FA                       <1>       cli
2261 0000CD08 881D[965F0100]           <1>       mov    [multi_tasking], bl ; 0 to disable, >0 to enable
2262 0000CD0E E9A0F7FFFF               <1>       jmp    sysret
2263                                    <1>
2264                                    <1> systimer:
2265                                    <1>       ; 02/01/2017
2266                                    <1>       ; 21/12/2016
2267                                    <1>       ; 19/12/2016
2268                                    <1>       ; 10/12/2016 (callback)
2269                                    <1>       ; 10/06/2016
2270                                    <1>       ; 07/06/2016
2271                                    <1>       ; 06/06/2016
2272                                    <1>       ; 21/05/2016
2273                                    <1>       ; 19/05/2016
2274                                    <1>       ; 18/05/2016 - TRDOS 386 (TRDOS v2.0)
2275                                    <1>       ; (TRDOS 386 feature only!)
2276                                    <1>       ;
2277                                    <1>       ; (start or stop timer event(s))
2278                                    <1>       ;
2279                                    <1>       ; INPUT ->
2280                                    <1>       ;     BL = Signal return byte (response byte)
2281                                    <1>       ;          (Any requested value between 0 and 255)
2282                                    <1>       ;          (Kernel will put it at the requested address)
2283                                    <1>       ;     BH = Time count unit
2284                                    <1>       ;          0 = Stop timer event
2285                                    <1>       ;          1 = 18.2 ticks per second
2286                                    <1>       ;          2 = 10 milliseconds
2287                                    <1>       ;          3 = 1 second (for real time clock interrupt)
2288                                    <1>       ;          4 = time/tick count in current time count unit
2289                                    <1>       ;         // 10/12/2016
2290                                    <1>       ;         80h = Stop timer event (callback method)
2291                                    <1>       ;         81h = 18.2 ticks per second, callback method
2292                                    <1>       ;         82h = 10 milliseconds, callback method
2293                                    <1>       ;         83h = 1 second (for RTC int), callback method
2294                                    <1>       ;         84h = current time count unit, callback method
2295                                    <1>       ;
2296                                    <1>       ;         Note: Only 03h or 83h will set real time clock
2297                                    <1>       ;             (RTC) events (Others are for PIT events)!
2298                                    <1>       ;
2299                                    <1>       ;     NOTE: If callback (user service) method is used,
2300                                    <1>       ;         EDX will point to the return address (of service
2301                                    <1>       ;         procedure) in user's space instead of signal
2302                                    <1>       ;         response byte address. (TRDOS 386 kernel will
2303                                    <1>       ;         direct the cpu to that address -in user's space-
2304                                    <1>       ;         at the return of system call or interrupt
2305                                    <1>       ;         just after the adjusted count/time is elapsed.)
2306                                    <1>       ;         User's sevice routine must be ended with a
2307                                    <1>       ;         'iret'. Normal return addresses from system
2308                                    <1>       ;         calls or and interrupts will be kept same except
2309                                    <1>       ;         the timer returns.
2310                                    <1>       ;
2311                                    <1>       ;     BH = 0 -> Stop timer event
2312                                    <1>       ;     BL = Timer event number (1 to 255) if BH = 0
2313                                    <1>       ;          If BL = 0, all timer events (which are belongs
2314                                    <1>       ;           to running process) will be stopped
2315                                    <1>       ;     ECX = Time/Tick count (depending on time count unit)
2316                                    <1>       ;     EDX = Signal return (Response) byte address
2317                                    <1>       ;           (virtual address in user's memory space)
2318                                    <1>       ; OUTPUT ->
2319                                    <1>       ;     AL = Timer event number   (1 to 255) (max. value = 16)
2320                                    <1>       ;     IF BH Input = 0 & CF = 0 & AL = 0 ->
2321                                    <1>       ;          timer event(s) has/have been stopped/finished
2322                                    <1>       ;     CF = 1 & AL = 0 -> no timer setting space to set
2323                                    <1>       ;     CF = 1 & AL > 0 -> timer count unit is not usable
2324                                    <1>       ;
2325                                    <1>       ;     NOTE: To modify a time count for a user function,
2326                                    <1>       ;           at first, current timer event must be stopped
2327                                    <1>       ;           then a new timer event (which is related with
2328                                    <1>       ;           same user function) must be started.
2329                                    <1>       ;
```

```
2330                                    <1>       ;           Signal return (response) byte may be used for
2331                                    <1>       ;           several purposes. Kernel will put this value
2332                                    <1>       ;           to requested address during timer interrupt,
2333                                    <1>       ;           program/user can check this value to understand
2334                                    <1>       ;           which event has been occurred and what is changed.
2335                                    <1>       ;           (Multi timer events can share same signal address)
2336                                    <1>       ;
2337                                    <1>       ;     NOTE: If the process is running while the time count
2338                                    <1>       ;           is reached, kernel will put signal return (response)
2339                                    <1>       ;           byte value at requested address during timer
2340                                    <1>       ;           interrupt and the process will continue to run.
2341                                    <1>       ;           Program/process must call (jump to) it's timer event
2342                                    <1>       ;           function as required, for checking the timer event
2343                                    <1>       ;           status via signal return (response) byte address.
2344                                    <1>       ;
2345                                    <1>       ;           If the process is not running (waiting or sleeping
2346                                    <1>       ;           or released) while the time count is reached,
2347                                    <1>       ;           it is restarted from where it left, to ensure
2348                                    <1>       ;           proper multi media (video, audio, clock, timer)
2349                                    <1>       ;           functionality.
2350                                    <1>       ;
2351                                    <1>       ;           (It is better to use 'syswait' or 'syssleep',
2352                                    <1>       ;           or 'sysrele' system call just after the timer
2353                                    <1>       ;           function. Otherwise, timer events may block other
2354                                    <1>       ;           processes which are not using timer events.)

2355                                    <1>       ;
2356                                    <1>       ; Timer Event Structure: (max. 16 timer events, 16*16 bytes)
2357                                    <1>       ;      Owner:              resb 1 ; 0 = free
2358                                    <1>       ;                         ;>0 = process number (u.uno)
2359                                    <1>       ;     Calback:     resb 1 ; 1 = callback, 0 = response byte
2360                                    <1>       ;     Interrupt:      resb 1 ; 0 = Timer interrupt (or none)
2361                                    <1>       ;                          ; 1 = Real Time Clock interrupt
2362                                    <1>       ;     Response:       resb 1 ; 0 to 255, signal return value
2363                                    <1>       ;     Count Limit: resd 1 ; count of ticks (total/set)
2364                                    <1>       ;     Current Count:      resd 1 ; count of ticks (current)
2365                                    <1>       ;     Response Addr:  resd 1 ; response byte (pointer) address
2366                                    <1>       ;
2367                                    <1>
2368                                    <1>       ; 19/12/2016 (timer callback)
2369 0000CD13 C605[D4640100]00          <1>       mov   byte [tcallback], 0
2370 0000CD1A C605[D5640100]00          <1>       mov   byte [trtc], 0
2371 0000CD21 C705[D0030300]0000-       <1>       mov   dword [u.tcb], 0 ; this is not necessary...
2371 0000CD29 0000                      <1>
2372                                    <1>
2373 0000CD2B 80FF80                    <1>       cmp   bh, 80h
2374 0000CD2E 7225                      <1>       jb    short systimer_cb2
2375 0000CD30 7704                      <1>       ja    short systimer_cb0
2376                                    <1>
2377 0000CD32 31D2                      <1>       xor   edx, edx ; 0, reset callback address
2378 0000CD34 EB0B                      <1>       jmp   short systimer_cb1
2379                                    <1>
2380                                    <1> systimer_cb0:
2381 0000CD36 80FF84                    <1>       cmp   bh, 84h
2382 0000CD39 7764                      <1>       ja    short systimer_5 ;  undefined, error
2383                                    <1>
2384                                    <1>       ;mov  byte [tcallback], 1 ; 19/12/2016
2385 0000CD3B FE05[D4640100]            <1>       inc   byte [tcallback]
2386                                    <1>
2387                                    <1> systimer_cb1:
2388 0000CD41 0FB635[B3030300]          <1>       movzx esi, byte [u.uno] ; process number
2389 0000CD48 66C1E602                  <1>       shl   si, 2
2390 0000CD4C 8996[0C010300]            <1>       mov   [esi+p.tcb-4], edx ; set process timer callback address
2391                                    <1>                                ; (overwrite prev value if it is set!)
2392 0000CD52 80E77F                    <1>       and   bh, 7Fh
2393                                    <1>
2394                                    <1> systimer_cb2:
2395 0000CD55 80FF02                    <1>       cmp   bh, 2
2396 0000CD58 7445                      <1>         je      short systimer_5  ; only 18.2 ticks per second is usable
2397                                    <1>                                 ; 10 milliseconds (100 Hertz) timer
2398                                    <1>                                 ; will be set later (18/05/2016)
2399 0000CD5A 774B                      <1>         ja      short systimer_6
2400                                    <1>
2401 0000CD5C 20FF                      <1>       and   bh, bh
2402 0000CD5E 0F84BA000000              <1>         jz      systimer_9        ; stop timer event(s)
2403                                    <1>
2404                                    <1>       ; bh = 1 (timer interrupt, 18.2 Hz, IBM PC/AT ROMBIOS default)
2405                                    <1>
2406                                    <1> systimer_19:
2407 0000CD64 B00A                      <1>       mov   al, 10 ; (*)
2408                                    <1>
2409                                    <1> systimer_0:
2410 0000CD66 B710                      <1>       mov   bh, 16
2411                                    <1>       ;
2412 0000CD68 383D[975F0100]            <1>       cmp   [timer_events], bh ; 16 ; 07/06/2016
2413 0000CD6E 7319                      <1>       jnb   short systimer_3  ; max. 16 timer events
2414                                    <1>       ;
2415 0000CD70 50                        <1>       push  eax ; (*)
2416                                    <1>
2417 0000CD71 BF[60040300]              <1>       mov   edi, timer_set  ; beginning address of timer events
2418                                    <1>                             ; setting space
2419 0000CD76 30C0                      <1>       xor   al, al ; 0
2420                                    <1> systimer_1:
2421 0000CD78 FEC0                      <1>       inc   al
2422 0000CD7A 803F00                    <1>       cmp   byte [edi], 0       ; is it free space ?
2423 0000CD7D 7639                      <1>       jna   short systimer_7 ; yes
2424 0000CD7F FECF                      <1>       dec   bh
2425 0000CD81 7405                      <1>       jz    short systimer_2
2426 0000CD83 83C710                    <1>       add   edi, 16
2427 0000CD86 EBF0                      <1>       jmp   short  systimer_1 ; next event space
2428                                    <1>
2429                                    <1> systimer_2:
```

```
2430 0000CD88 58                  <1>       pop    eax ; (*) discard
2431                              <1> systimer_3:
2432 0000CD89 C605[64030300]00    <1>       mov    byte [u.r0], 0
2433                              <1> systimer_4:
2434 0000CD90 C705[C8030300]1B00- <1>         mov     dword [u.error], ERR_MISC
2434 0000CD98 0000                <1>
2435                              <1>                                     ; one of miscellaneous/other errors
2436 0000CD9A E9F4F6FFFF          <1>       jmp    error ; cf -> 1
2437                              <1>
2438                              <1> systimer_5:
2439 0000CD9F 883D[64030300]      <1>       mov    [u.r0], bh ; Time count unit (=2 or >3)
2440 0000CDA5 EBE9                <1>       jmp    short systimer_4 ; 07/06/2016
2441                              <1>
2442                              <1> systimer_6:
2443 0000CDA7 80FF04              <1>       cmp    bh, 4
2444 0000CDAA 77F3                <1>         ja     short systimer_5  ; undefined time count unit
2445                              <1>       ;jb    short systimer_16
2446                              <1>
2447                              <1>       ;mov   al, 1  ; default (use current timer unit)
2448                              <1>                     ; countdown value is in ECX !
2449                              <1>                     ; max. value of ecx = 4294967296/10
2450                              <1>         ;jmp    short systimer_0
2451                              <1>       ;jmp   short systimer_19
2452 0000CDAC 74B6                <1>       je     short systimer_19
2453                              <1>
2454                              <1> systimer_16:
2455                              <1>       ; bh = 3
2456                              <1>       ; timer event via real time clock interrupt
2457                              <1>       ; interrupt/update frequency: 1 Hz (1 tick per second)
2458                              <1>
2459 0000CDAE B0B6                <1>       mov    al, 182 ; (*) ; 18.2 * 10
2460 0000CDB0 FE05[D5640100]      <1>       inc    byte [trtc] ; timer event via real time clock
2461 0000CDB6 EBAE                <1>         jmp     short systimer_0
2462                              <1>
2463                              <1> systimer_7:
2464 0000CDB8 A2[64030300]        <1>       mov    [u.r0], al ; timer event number
2465                              <1>       ;
2466                              <1>       ; edi = address of empty timer event area
2467 0000CDBD A0[B3030300]        <1>       mov    al, [u.uno]
2468 0000CDC2 FA                  <1>       cli    ; disable interrupts
2469 0000CDC3 AA                  <1>       stosb  ; process number
2470 0000CDC4 A0[D4640100]        <1>       mov    al, [tcallback] ; timer callback flag
2471 0000CDC9 AA                  <1>       stosb  ; 1= callback method, 0= signal response byte method
2472 0000CDCA A0[D5640100]        <1>       mov    al, [trtc] ; timer interrupt type
2473 0000CDCF AA                  <1>       stosb  ; 1= real time clock, 0= programmable interval timer
2474 0000CDD0 88D8                <1>       mov    al, bl ; Signal return (Response) value
2475 0000CDD2 AA                  <1>       stosb   ; response byte
2476 0000CDD3 58                  <1>       pop    eax ; (*) ; 10 or 182
2477 0000CDD4 89D3                <1>       mov    ebx, edx ; virtual address for response/signal byte
2478 0000CDD6 F7E1                <1>       mul    ecx
2479                              <1>       ; (eax = 10 * count of 18.2 Hz timer ticks)
2480                              <1>       ; (count down step = 10)
2481 0000CDD8 AB                  <1>       stosd  ; count limit (reset value)
2482 0000CDD9 AB                  <1>       stosd  ; current count value
2483                              <1>
2484                              <1>       ; 19/12/2016
2485 0000CDDA 803D[D4640100]00    <1>       cmp    byte [tcallback], 0 ; timer callback method ?
2486 0000CDE1 7604                <1>       jna    short systimer_17 ; no
2487 0000CDE3 89D8                <1>       mov    eax, ebx ; virtual address for callback routine
2488 0000CDE5 EB0D                <1>       jmp    short systimer_18
2489                              <1>
2490                              <1> systimer_17: ; signal response byte method
2491                              <1>       ; ebx = virtual address
2492                              <1>       ; [u.pgdir] = page directory's physical address
2493                              <1>       ; 20/02/2017
2494 0000CDE7 FE05[D6640100]      <1>       inc     byte [no_page_swap] ; 1
2495                              <1>                     ; Do not add this page to swap queue
2496                              <1>                     ; and remove it from swap queue if it is
2497                              <1>                     ; on the queue.
2498 0000CDED E89C84FFFF          <1>       call   get_physical_addr
2499 0000CDF2 721A                <1>       jc     short systimer_8 ; 07/06/2016
2500                              <1>       ; eax = physical address of the virtual address in user's space
2501                              <1> systimer_18:
2502 0000CDF4 AB                  <1>       stosd  ; response addr (physical) or callback addr (virtual)
2503 0000CDF5 FE05[975F0100]      <1>       inc    byte [timer_events] ; 07/06/201
2504                              <1>       ; 02/01/2017
2505 0000CDFB 0FB605[B3030300]    <1>       movzx  eax, byte [u.uno]
2506 0000CE02 FE80[FF000300]      <1>       inc    byte [eax+p.timer-1]
2507                              <1>       ;
2508 0000CE08 FB                  <1>       sti    ; enable interrupts
2509 0000CE09 E9A5F6FFFF          <1>       jmp    sysret
2510                              <1>
2511                              <1> systimer_8:
2512                              <1>       ; 10/06/2016
2513                              <1>       ; 07/06/2016
2514 0000CE0E 28C0                <1>       sub    al, al ; 0
2515 0000CE10 8847F4              <1>       mov    [edi-12], al ; clear process number (free timer event)
2516                              <1>       ;mov   dword [edi], eax ; 0
2517 0000CE13 FB                  <1>       sti
2518 0000CE14 A2[64030300]        <1>       mov    [u.r0], al ; 0
2519 0000CE19 E975F6FFFF          <1>       jmp    error
2520                              <1>
2521                              <1> systimer_9:
2522                              <1>       ; 10/06/2016
2523                              <1>       ; 07/06/2016
2524 0000CE1E 28C0                <1>       sub    al, al
2525 0000CE20 A2[64030300]        <1>       mov    byte [u.r0], al ; 0
2526 0000CE25 3805[975F0100]      <1>       cmp     byte [timer_events], al ;  0
2527 0000CE2B 7631                <1>       jna    short systimer_12
2528                              <1>
2529                              <1>       ; Note: ecx and edx are undefined here
2530                              <1>       ;     (for stop timer function)
```

```
2531                                    <1>
2532 0000CE2D BE[60040300]              <1>       mov   esi, timer_set  ; beginning address of timer events
2533                                    <1>                             ; setting space
2534 0000CE32 A0[B3030300]              <1>       mov   al, [u.uno]
2535                                    <1>
2536 0000CE37 B710                      <1>       mov   bh, 16
2537                                    <1>
2538 0000CE39 08DB                      <1>       or    bl, bl
2539 0000CE3B 7544                      <1>       jnz   short systimer_15
2540                                    <1>
2541                                    <1>       ; clear timer event areas belong to current process
2542                                    <1>       ; (for stopping all timer events belong to current process)
2543 0000CE3D FA                        <1>       cli   ; disable interrupts
2544                                    <1> systimer_10:
2545                                    <1>       ; 10/06/2016
2546                                    <1>       ; 07/06/2016
2547 0000CE3E 8A26                      <1>       mov   ah, [esi]
2548 0000CE40 08E4                      <1>       or    ah, ah ; 0 ?
2549 0000CE42 7411                      <1>       jz    short systimer_11
2550 0000CE44 38C4                      <1>       cmp   ah, al ; is the process number (owner) same ?
2551 0000CE46 750D                      <1>         jne     short systimer_11 ; no
2552                                    <1>
2553                                    <1>       ;mov   byte [esi], 0
2554 0000CE48 66C7060000                <1>       mov   word [esi], 0 ; clear
2555                                    <1>       ;mov   dword [esi+12], 0 ; clear
2556                                    <1>
2557 0000CE4D FE0D[975F0100]            <1>       dec   byte [timer_events]
2558 0000CE53 7409                      <1>       jz    short systimer_12
2559                                    <1>
2560                                    <1> systimer_11:
2561 0000CE55 FECF                      <1>       dec   bh
2562 0000CE57 7405                      <1>       jz    short systimer_12
2563 0000CE59 83C610                    <1>       add   esi, 16
2564 0000CE5C EBE0                      <1>       jmp   short systimer_10
2565                                    <1>
2566                                    <1> systimer_12:
2567 0000CE5E 0FB635[B3030300]          <1>       movzx esi, byte [u.uno]
2568 0000CE65 08DB                      <1>       or    bl, bl ; all timer events or one timer event ?
2569 0000CE67 740C                      <1>       jz    short systimer_13
2570 0000CE69 8A9E[FF000300]            <1>       mov   bl, [esi+p.timer-1]
2571 0000CE6F 20DB                      <1>       and   bl, bl ; previous number of timer events for the process
2572 0000CE71 7408                      <1>       jz    short systimer_14
2573 0000CE73 FECB                      <1>       dec   bl  ; previous number of timer events for the process - 1
2574                                    <1> systimer_13:
2575 0000CE75 889E[FF000300]            <1>       mov   [esi+p.timer-1], bl ; 0 ; no timer events for process
2576                                    <1> systimer_14:
2577 0000CE7B FB                        <1>       sti   ; enable interrupts
2578 0000CE7C E932F6FFFF                <1>       jmp   sysret
2579                                    <1>
2580                                    <1> systimer_15:
2581 0000CE81 38FB                      <1>       cmp   bl, bh ; 16
2582 0000CE83 0F8707FFFFFF              <1>         ja      systimer_4      ; max. 16 timer events !
2583                                    <1>       ;
2584 0000CE89 88DA                      <1>       mov   dl, bl
2585 0000CE8B FECA                      <1>       dec   dl  ; 16 -> 15 ... 1 -> 0
2586 0000CE8D C0E204                    <1>       shl   dl, 4 ; * 16
2587 0000CE90 0FB6FA                    <1>       movzx edi, dl
2588 0000CE93 01F7                      <1>       add   edi, esi ; timer_set
2589                                    <1>
2590 0000CE95 3A07                      <1>       cmp   al, [edi] ; process number
2591 0000CE97 0F85F3FEFFFF              <1>         jne     systimer_4
2592                                    <1>
2593                                    <1>       ; same process ID
2594 0000CE9D FA                        <1>       cli   ; disable interrupts
2595                                    <1>       ; 10/06/2016 ; 02/01/2017
2596                                    <1>       ;mov   byte [edi], 0
2597 0000CE9E 66C7070000                <1>       mov   word [edi], 0 ; clear
2598                                    <1>       ;mov   dword [edi+12], 0 ; clear
2599 0000CEA3 FE0D[975F0100]            <1>       dec   byte [timer_events]
2600 0000CEA9 EBB3                      <1>       jmp   short systimer_12
2601                                    <1>
2602                                    <1> sysmdate: ; < change the modification time of a file >
2603                                    <1>       ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
2604                                    <1>       ; temporary !
2605 0000CEAB B801000000                <1>       mov   eax, ERR_INV_FNUMBER ; 'invalid function number !'
2606 0000CEB0 A3[C8030300]              <1>         mov     [u.error], eax
2607 0000CEB5 A3[64030300]              <1>         mov     [u.r0], eax
2608 0000CEBA E9D4F5FFFF                <1>       jmp   error
2609                                    <1>
2610                                    <1> sysvideo: ; VIDEO DATA TRANSFER FUNCTIONS
2611                                    <1>       ; 12/05/2017
2612                                    <1>       ; 11/07/2016
2613                                    <1>       ; 13/06/2016
2614                                    <1>       ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
2615                                    <1>       ;
2616                                    <1>       ;
2617                                    <1>       ; VIDEO DATA TRANSFER FUNCTIONS:
2618                                    <1>       ;
2619                                    <1>       ; Inputs:
2620                                    <1>       ;     BH = 0 = VIDEO BIOS Mode 3, tty/text mode data transfers
2621                                    <1>       ;          BL =
2622                                    <1>       ;            Bits 0&1, Transfer direction
2623                                    <1>       ;                  0 - System to system
2624                                    <1>       ;                  1 - User to system
2625                                    <1>       ;                  2 - System to user
2626                                    <1>       ;                  3 - User to user
2627                                    <1>       ;            Bits 2&3, Transfer Type
2628                                    <1>       ;                  0 - Display page transfer
2629                                    <1>       ;                  1 - Display page window transfer
2630                                    <1>       ;                  2 - Frame/Viewport/Window address transfer
2631                                    <1>       ;                  3 - Window handle transfer
2632                                    <1>       ;
```

```
2633                                    <1>       ;           /// BL = 0 -> System to system (display page) transfer
2634                                    <1>       ;              CL = Source page
2635                                    <1>       ;              DL = Destination page
2636                                    <1>       ;           /// BL = 1&2 -> user to system & system to user transfer
2637                                    <1>       ;              ECX = User buffer
2638                                    <1>       ;              DL = Video page
2639                                    <1>       ;           /// BL = 5&6 -> user to system, system to user transfer
2640                                    <1>       ;             (window in current display page and in current mode)
2641                                    <1>       ;              ESI = User's buffer address
2642                                    <1>       ;              ECX Low 16 bits = Top left column (X1 position)
2643                                    <1>       ;              ECX High 16 bits = Top row (Y1 position)
2644                                    <1>       ;              EDX Low 16 bits = Bottom right column (X2 position)
2645                                    <1>       ;              EDX High 16 bits = Bottom row (Y2 position)
2646                                    <1>         ;                 If BL = 5 ->
2647                                    <1>       ;              EDI = Swap address (in user's memory space)
2648                                    <1>       ;              (If swap address > 0, previous content of the window
2649                                    <1>       ;              will be saved into swap area in user's memory space)
2650                                    <1>       ;           /// BL = 4 -> system to system transfer
2651                                    <1>       ;              ESI = System's source buffer (video page) address
2652                                    <1>       ;              ECX Low 16 bits = Top left column (X1 position)
2653                                    <1>       ;              ECX High 16 bits = Top row (Y1 position)
2654                                    <1>       ;              EDX Low 16 bits = Bottom right column (X2 position)
2655                                    <1>       ;              EDX High 16 bits = Bottom row (Y2 position)
2656                                    <1>       ;              EDI = System's destination buffer (video page) address
2657                                    <1>       ;
2658                                    <1>       ;       BH = 1 = CGA Graphics (0B8000h) data transfers
2659                                    <1>       ;            BL =
2660                                    <1>       ;              0 = Fill color (color in CL] (32K)
2661                                    <1>       ;              1 = User to system display page transfer
2662                                    <1>       ;              2 = System to user display page transfer
2663                                    <1>       ;              3 = NOT bits in window (ECX, EDX)
2664                                    <1>       ;              4 = Window copy (system to system)
2665                                    <1>       ;              5 = User to system window transfer
2666                                    <1>       ;              6 = System to user window transfer
2667                                    <1>       ;              7 = AND display page bytes with CL
2668                                    <1>       ;              8 = OR display page bytes with CL
2669                                    <1>       ;              9 = XOR display page bytes with CL
2670                                    <1>       ;
2671                                    <1>       ;           /// BL = 0 -> Fill color  (all screen pixels)
2672                                    <1>       ;              CL = Color value
2673                                    <1>       ;           /// BL = 1&2 -> user to system & system to user transfer
2674                                    <1>       ;              ECX = User buffer
2675                                    <1>       ;           /// BL = 5&6 -> user to system, system to user transfer
2676                                    <1>       ;             (window in current display page and in current mode)
2677                                    <1>       ;              ESI = User's buffer address
2678                                    <1>       ;              ECX Low 16 bits = Top left column (X1 position)
2679                                    <1>       ;              ECX High 16 bits = Top row (Y1 position)
2680                                    <1>       ;              EDX Low 16 bits = Bottom right column (X2 position)
2681                                    <1>       ;              EDX High 16 bits = Bottom row (Y2 position)
2682                                    <1>         ;           /// BL = 4 -> system to system (window) transfer
2683                                    <1>       ;              ESI = System's source buffer (video page) address
2684                                    <1>       ;              ECX Low 16 bits = Top left column (X1 position)
2685                                    <1>       ;              ECX High 16 bits = Top row (Y1 position)
2686                                    <1>       ;              EDX Low 16 bits = Bottom right column (X2 position)
2687                                    <1>       ;              EDX High 16 bits = Bottom row (Y2 position)
2688                                    <1>       ;              EDI = System's destination buffer (video page) address
2689                                    <1>       ;           /// BL = 3 -> NOT byte in display page/memory
2690                                    <1>       ;              ECX Low 16 bits = Top left column (X1 position)
2691                                    <1>       ;              ECX High 16 bits = Top row (Y1 position)
2692                                    <1>       ;              EDX Low 16 bits = Bottom right column (X2 position)
2693                                    <1>       ;              EDX High 16 bits = Bottom row (Y2 position)
2694                                    <1>       ;
2695                                    <1>       ;       BH = 2 = VGA Graphics (0A0000h) data transfers
2696                                    <1>       ;            BL =
2697                                    <1>       ;              x0h = Fill color (color in CL] (64K)
2698                                    <1>       ;              x1h = User to system display page transfer
2699                                    <1>       ;              x2h = System to user display page transfer
2700                                    <1>       ;              x3h = NOT bits in window (ECX, EDX)
2701                                    <1>       ;              x4h = Window copy (system to system)
2702                                    <1>       ;              x5h = User to system window transfer
2703                                    <1>       ;              x6h = System to user window transfer
2704                                    <1>       ;              x7h = AND display page bytes with CL
2705                                    <1>       ;              x8h = OR display page bytes with CL
2706                                    <1>       ;              x9h = XOR display page bytes with CL
2707                                    <1>       ;              x = 0 -> screen width = 320
2708                                    <1>       ;              x = 1 -> screen width = 640
2709                                    <1>       ;              x = 2 -> screen width = 800
2710                                    <1>       ;
2711                                    <1>       ;           /// BL = 0 -> Fill color  (all screen pixels)
2712                                    <1>       ;              CL = Color value
2713                                    <1>       ;           /// BL = 1&2 -> user to system & system to user transfer
2714                                    <1>       ;              ECX = User buffer
2715                                    <1>       ;           /// BL = 5&6 -> user to system, system to user transfer
2716                                    <1>       ;             (window in current display page and in current mode)
2717                                    <1>       ;              ESI = User's buffer address
2718                                    <1>       ;              ECX Low 16 bits = Top left column (X1 position)
2719                                    <1>       ;              ECX High 16 bits = Top row (Y1 position)
2720                                    <1>       ;              EDX Low 16 bits = Bottom right column (X2 position)
2721                                    <1>       ;              EDX High 16 bits = Bottom row (Y2 position)
2722                                    <1>               ;           /// BL = 4 -> system to system (window) transfer
2723                                    <1>       ;              ESI = System's source buffer (video page) address
2724                                    <1>       ;              ECX Low 16 bits = Top left column (X1 position)
2725                                    <1>       ;              ECX High 16 bits = Top row (Y1 position)
2726                                    <1>       ;              EDX Low 16 bits = Bottom right column (X2 position)
2727                                    <1>       ;              EDX High 16 bits = Bottom row (Y2 position)
2728                                    <1>       ;              EDI = System's destination buffer (video page) address
2729                                    <1>       ;           /// BL = 3 -> NOT byte in display page/memory
2730                                    <1>       ;              ECX Low 16 bits = Top left column (X1 position)
2731                                    <1>       ;              ECX High 16 bits = Top row (Y1 position)
2732                                    <1>       ;              EDX Low 16 bits = Bottom right column (X2 position)
2733                                    <1>       ;              EDX High 16 bits = Bottom row (Y2 position)
2734                                    <1>       ;
```

```
2735                    <1>      ;     BH = 3 = Super VGA, LINEAR FRAME BUFFER data transfers
2736                    <1>      ;          BL =
2737                    <1>      ;            0 = Fill color (color in ECX] (Frame buffer size)
2738                    <1>      ;            1 = User to system display page transfer
2739                    <1>      ;            2 = System to user display page transfer
2740                    <1>      ;            3 = NOT bits in window (ECX, EDX)
2741                    <1>      ;            4 = Window copy (system to system)
2742                    <1>      ;            5 = User to system window transfer
2743                    <1>      ;            6 = System to user window transfer
2744                    <1>      ;            7 = AND display page bytes with ECX
2745                    <1>      ;            8 = OR display page bytes with ECX
2746                    <1>      ;            9 = XOR display page bytes with ECX
2747                    <1>      ;
2748                    <1>      ;          /// BL = 0 -> Fill color  (all screen pixels)
2749                    <1>      ;             CL = Color value
2750                    <1>      ;          /// BL = 1&2 -> user to system & system to user transfer
2751                    <1>      ;             ECX = User buffer
2752                    <1>      ;          /// BL = 5&6 -> user to system, system to user transfer
2753                    <1>      ;            (window in current display page and in current mode)
2754                    <1>      ;             ESI = User's buffer address
2755                    <1>      ;             ECX Low 16 bits = Top left column (X1 position)
2756                    <1>      ;             ECX High 16 bits = Top row (Y1 position)
2757                    <1>      ;             EDX Low 16 bits = Bottom right column (X2 position)
2758                    <1>      ;             EDX High 16 bits = Bottom row (Y2 position)
2759                    <1>      ;          /// BL = 4 -> system to system (window) transfer
2760                    <1>      ;             ESI = System's source buffer (video page) address
2761                    <1>      ;             ECX Low 16 bits = Top left column (X1 position)
2762                    <1>      ;             ECX High 16 bits = Top row (Y1 position)
2763                    <1>      ;             EDX Low 16 bits = Bottom right column (X2 position)
2764                    <1>      ;             EDX High 16 bits = Bottom row (Y2 position)
2765                    <1>      ;             EDI = System's destination buffer (video page) address
2766                    <1>      ;          /// BL = 3 -> NOT byte in display page/memory
2767                    <1>      ;             ECX Low 16 bits = Top left column (X1 position)
2768                    <1>      ;             ECX High 16 bits = Top row (Y1 position)
2769                    <1>      ;             EDX Low 16 bits = Bottom right column (X2 position)
2770                    <1>      ;             EDX High 16 bits = Bottom row (Y2 position)
2771                    <1>      ;
2772                    <1>      ; Outputs:
2773                    <1>      ;     EAX = transfer/byte count
2774                    <1>      ;
2775                    <1>      ;     NOTE: If the source or destination address passes out of
2776                    <1>      ;     video pages (display memory limits), data will not be transferred
2777                    <1>      ;     and EAX will return as 0.
2778                    <1>      ;
2779                    <1>      ;
2780                    <1>      ; DIRECT (STANDARD VGA/CGA) DISPLAY MEMORY ACCESS FUNCTIONS:
2781                    <1>      ;
2782                    <1>      ;     BH = 4 = CGA direct video memory (0B8000h, 32K) access
2783                    <1>      ;            Page directory & page tables of the user's
2784                    <1>      ;            program will be updated to direct access to
2785                    <1>      ;            0B8000h (32K) video (CGA, color) memory; if
2786                    <1>      ;            there is not a permission  conflict or lock!
2787                    <1>      ;             (User's program/process will have permision to
2788                    <1>      ;            access locked display memory if the owner is
2789                    <1>      ;            it's parent.)
2790                    <1>        ;
2791                    <1>      ;         Screen width = 320
2792                    <1>      ;
2793                    <1>      ;     BH = 5 = VGA direct video memory (0A0000h, 64K) access
2794                    <1>      ;            Page directory & page tables of the user's
2795                    <1>      ;            program will be updated to direct access to
2796                    <1>      ;            0A0000h (64K) video (VGA) memory; if there is not
2797                    <1>      ;            a permission conflict or lock!
2798                    <1>      ;             (User's program/process will have permision to
2799                    <1>      ;            access locked display memory if the owner is
2800                    <1>      ;            it's parent.)
2801                    <1>      ;
2802                    <1>      ;         BL = Screen width (320, 640, 800)
2803                    <1>      ;
2804                    <1>      ; Outputs:
2805                    <1>      ;     EAX = Display mmory address for direct access
2806                    <1>      ;            0A0000h for VGA, 0B8000h for CGA
2807                    <1>      ;     (Display memory size: 32K for CGA, 64K for VGA)
2808                    <1>      ;     EAX = 0 if display page access permission has been denied.
2809                    <1>      ;            (Locked!)
2810                    <1>      ;
2811                    <1>      ; LINEAR FRAME BUFFER ACCESS FUNCTIONS:
2812                    <1>      ;
2813                    <1>      ;     BH = 6 = Linear Frame Buffer direct video memory access
2814                    <1>      ;
2815                    <1>      ;            Page directory & page tables of the user's
2816                    <1>      ;            program will be updated to direct access to
2817                    <1>      ;            the configured LFB (Linear Frame Buffer) address,
2818                    <1>      ;            if there is not a permission conflict or lock!
2819                    <1>      ;             (User's program/process will have permision to
2820                    <1>      ;            access locked display memory if the owner is
2821                    <1>      ;            it's parent.)
2822                    <1>      ;
2823                    <1>      ;            Return: EAX = Linear Frame Buffer address
2824                    <1>      ;                  EDX = Frame Buffer Size in bytes
2825                    <1>      ;
2826                    <1>      ;     BH = 7 = Get Linear Frame Buffer info (for current mode)
2827                    <1>      ;
2828                    <1>      ;            Return:
2829                    <1>      ;            EAX = Frame Buffer Address (0 = is not in use)
2830                    <1>      ;            EDX = Frame Buffer Size in bytes
2831                    <1>      ;            BL = Current Video Mode
2832                    <1>      ;                 BL = 0FFh -> Super VGA (Extended VGA)
2833                    <1>      ;                 If BL = 0FFh,
2834                    <1>        ;                       BH = 0 = 16 colors
2835                    <1>      ;                  BH = 1 = 256 colors
2836                    <1>      ;                  BH = 2 = 66536 colors
```

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

                                <1>
                                <1>       ; 16/05/2016
0000CEBF 31C0                   <1>       xor    eax, eax
0000CEC1 A3[64030300]           <1>       mov    [u.r0], eax
                                <1>
0000CEC6 20FF                   <1>       and    bh, bh
0000CEC8 0F8572020000           <1>       jnz    sysvideo_13 ; 11/07/2016
                                <1>
                                <1>       ; Video mode 0, 80*25 text mode, CGA 16 colors  ; [CRT_MODE] = 3
0000CECE 88DF                   <1>       mov    bh, bl
0000CED0 C0EF02                 <1>       shr    bh, 2
0000CED3 20FF                   <1>       and    bh, bh
0000CED5 0F8598000000           <1>         jnz     sysvideo_4
0000CEDB BF00800B00             <1>       mov    edi, 0B8000h
0000CEE0 20D2                   <1>       and    dl, dl
0000CEE2 7413                   <1>       jz     short sysvideo_1
0000CEE4 80FA07                 <1>       cmp    dl, 7
0000CEE7 0F87C6F5FFFF           <1>       ja     sysret
                                <1> sysvideo_0:
0000CEED 81C7A00F0000           <1>       add    edi, 80*25*2
0000CEF3 FECA                   <1>       dec    dl
0000CEF5 75F6                   <1>       jnz    short sysvideo_0
                                <1> sysvideo_1:
0000CEF7 80E303                 <1>       and    bl, 3
0000CEFA 7530                   <1>       jnz    short sysvideo_2
0000CEFC 80F907                 <1>       cmp    cl, 7
0000CEFF 0F87AEF5FFFF           <1>       ja     sysret
                                <1>       ; system to system video/display page transfer (mode 0)
0000CF05 BE00800B00             <1>       mov    esi, 0B8000h
0000CF0A 0FB6C1                 <1>       movzx  eax, cl
0000CF0D BAA00F0000             <1>       mov    edx, 80*25*2
0000CF12 F7E2                   <1>       mul    edx
0000CF14 01C6                   <1>       add    esi, eax
0000CF16 B9A00F0000             <1>       mov    ecx, (80*25*2)
0000CF1B 890D[64030300]         <1>       mov    [u.r0], ecx
0000CF21 66C1E902               <1>       shr    cx, 2 ; /4
0000CF25 F3A5                   <1>       rep    movsd
0000CF27 E987F5FFFF             <1>       jmp    sysret
                                <1> sysvideo_2:
0000CF2C 80FB02                 <1>       cmp    bl, 2
0000CF2F 0F877EF5FFFF           <1>         ja      sysret
0000CF35 721F                   <1>       jb     short sysvideo_3
                                <1>       ; system to user video/display page transfer (mode 0)
0000CF37 89FE                   <1>       mov    esi, edi
0000CF39 89CF                   <1>       mov    edi, ecx ; user buffer
0000CF3B B9A00F0000             <1>       mov    ecx, 80*25*2
0000CF40 E841190000             <1>       call   transfer_to_user_buffer ; fast transfer
0000CF45 0F8268F5FFFF           <1>       jc     sysret
```

```
2938 0000CF4B 890D[64030300]      <1>       mov    [u.r0], ecx
2939 0000CF51 E95DF5FFFF          <1>       jmp    sysret
2940                              <1> sysvideo_3:
2941                              <1>       ; user to system video/display page transfer (mode 0)
2942 0000CF56 89CE                <1>       mov    esi, ecx ; user buffer
2943                              <1>       ; edi = video page address
2944 0000CF58 B9A00F0000          <1>       mov    ecx, 80*25*2
2945 0000CF5D E86E190000          <1>       call   transfer_from_user_buffer ; fast transfer
2946 0000CF62 0F824BF5FFFF        <1>       jc     sysret
2947 0000CF68 890D[64030300]      <1>       mov    [u.r0], ecx
2948 0000CF6E E940F5FFFF          <1>       jmp    sysret
2949                              <1> sysvideo_4:
2950 0000CF73 80E303              <1>       and    bl, 3
2951 0000CF76 0F85F6000000        <1>         jnz     sysvideo_9
2952 0000CF7C 80F907              <1>       cmp    cl, 7
2953 0000CF7F 0F872EF5FFFF        <1>       ja     sysret
2954                              <1>       ; system to system video/display page window transfer (mode 0)
2955 0000CF85 81FE00800B00        <1>       cmp    esi, 0B8000h
2956 0000CF8B 0F8222F5FFFF        <1>       jb     sysret
2957 0000CF91 81FE00FD0B00        <1>       cmp    esi, 0B8000h+(80*25*2*8)
2958 0000CF97 0F8316F5FFFF        <1>       jnb    sysret
2959 0000CF9D 81FF00800B00        <1>       cmp    edi, 0B8000h
2960 0000CFA3 0F820AF5FFFF        <1>       jb     sysret
2961 0000CFA9 81FF00FD0B00        <1>         cmp     edi, 0B8000h+(80*25*2*8)
2962 0000CFAF 0F83FEF4FFFF        <1>       jnb    sysret
2963                              <1>       ;
2964 0000CFB5 51                  <1>       push   ecx
2965 0000CFB6 52                  <1>       push   edx
2966 0000CFB7 0FB7C1              <1>       movzx  eax, cx ; top left column
2967 0000CFBA 50                  <1>       push   eax
2968 0000CFBB C1E910              <1>       shr    ecx, 16 ; top row
2969 0000CFBE 66B8A000            <1>       mov    ax, 80*2 ; 80 colums, 160 bytes per row
2970 0000CFC2 F7E1                <1>       mul    ecx
2971 0000CFC4 01C6                <1>       add    esi, eax
2972 0000CFC6 01C7                <1>       add    edi, eax
2973 0000CFC8 58                  <1>       pop    eax
2974 0000CFC9 66D1E0              <1>       shl    ax, 1 ; *2
2975 0000CFCC 01C6                <1>       add    esi, eax
2976 0000CFCE 01C7                <1>       add    edi, eax
2977 0000CFD0 5A                  <1>       pop    edx
2978 0000CFD1 59                  <1>       pop    ecx
2979 0000CFD2 B800FD0B00          <1>       mov    eax, 0B8000h+(80*25*2*8)
2980 0000CFD7 39C6                <1>       cmp    esi, eax
2981 0000CFD9 0F83D4F4FFFF        <1>       jnb    sysret
2982 0000CFDF 39C6                <1>       cmp    esi, eax
2983 0000CFE1 0F83CCF4FFFF        <1>       jnb    sysret
2984                              <1>
2985 0000CFE7 56                  <1>       push   esi ; ****
2986 0000CFE8 57                  <1>       push   edi ; ***
2987 0000CFE9 52                  <1>       push   edx ; **
2988 0000CFEA 51                  <1>       push   ecx ; *
2989 0000CFEB C1E910              <1>       shr    ecx, 16 ; top row
2990 0000CFEE C1EA10              <1>       shr    edx, 16 ; bottom row
2991 0000CFF1 83F918              <1>       cmp    ecx, 24 ; max. 25 rows
2992 0000CFF4 7773                <1>       ja     short sysvideo_6
2993 0000CFF6 83FA18              <1>       cmp    edx, 24 ; max. 25 rows
2994 0000CFF9 776E                <1>       ja     short sysvideo_6
2995 0000CFFB 28CA                <1>       sub    dl, cl
2996 0000CFFD 726A                <1>       jc     short sysvideo_6
2997 0000CFFF 50                  <1>       push   eax ; *****
2998 0000D000 89D3                <1>       mov    ebx, edx ; row count - 1
2999 0000D002 B8A0000000          <1>       mov    eax, 80*2
3000 0000D007 F7E0                <1>       mul    eax
3001 0000D009 01C6                <1>       add    esi, eax
3002 0000D00B 01C7                <1>       add    edi, eax
3003 0000D00D 58                  <1>       pop    eax ; *****
3004 0000D00E 39C6                <1>       cmp    esi, eax
3005 0000D010 7757                <1>       ja     short sysvideo_6
3006 0000D012 39C7                <1>       cmp    edi, eax
3007 0000D014 7753                <1>       ja     short sysvideo_6
3008 0000D016 59                  <1>       pop    ecx ; *
3009 0000D017 5A                  <1>       pop    edx ; **
3010 0000D018 81E1FFFF0000        <1>       and    ecx, 0FFFFh
3011 0000D01E 81E2FFFF0000        <1>       and    edx, 0FFFFh
3012 0000D024 83F94F              <1>       cmp    ecx, 79 ; max. 80 columns
3013 0000D027 7742                <1>       ja     short sysvideo_7
3014 0000D029 83FA4F              <1>       cmp    edx, 79 ; max. 80 columns
3015 0000D02C 773D                <1>       ja     short sysvideo_7
3016 0000D02E 28CA                <1>       sub    dl, cl
3017 0000D030 7639                <1>       jna    short sysvideo_7
3018                              <1>       ; edx = column count (width) - 1
3019 0000D032 D0E2                <1>       shl    dl, 1
3020 0000D034 01D6                <1>       add    esi, edx
3021 0000D036 01D7                <1>       add    edi, edx
3022 0000D038 39C6                <1>       cmp    esi, eax
3023 0000D03A 772F                <1>       ja     short sysvideo_7
3024 0000D03C 39C7                <1>       cmp    edi, eax
3025 0000D03E 772B                <1>       ja     short sysvideo_7
3026 0000D040 5F                  <1>       pop    edi ; ***
3027 0000D041 5E                  <1>       pop    esi ; ****
3028 0000D042 FEC3                <1>       inc    bl
3029 0000D044 FEC2                <1>       inc    dl ; column count
3030 0000D046 88D7                <1>       mov    bh, dl
3031 0000D048 D0E2                <1>       shl    dl, 1
3032 0000D04A B8A0000000          <1>       mov    eax, 80*2
3033 0000D04F 28D0                <1>       sub    al, dl ; (80 - columns) * 2
3034                              <1> sysvideo_5:
3035 0000D051 88F9                <1>       mov    cl, bh
3036 0000D053 0115[64030300]      <1>       add    [u.r0], edx
3037 0000D059 F366A5              <1>       rep    movsw
3038 0000D05C 01C6                <1>       add    esi, eax ; next row
3039 0000D05E 01C7                <1>       add    edi, eax ; next row
```

```
3040 0000D060 FECB                <1>       dec    bl
3041 0000D062 75ED                <1>       jnz    short sysvideo_5
3042 0000D064 E94AF4FFFF          <1>       jmp    sysret
3043                              <1>
3044                              <1> sysvideo_6:
3045 0000D069 59                  <1>       pop    ecx ; *
3046 0000D06A 5A                  <1>       pop    edx ; **
3047                              <1> sysvideo_7:
3048 0000D06B 5F                  <1>       pop    edi ; ***
3049 0000D06C 5E                  <1>       pop    esi ; ****
3050 0000D06D E941F4FFFF          <1>       jmp    sysret
3051                              <1>
3052                              <1> sysvideo_9:
3053 0000D072 80FB02              <1>       cmp    bl, 2
3054 0000D075 0F8738F4FFFF        <1>         ja      sysret
3055                              <1>
3056 0000D07B 56                  <1>       push   esi ; ****
3057 0000D07C 57                  <1>       push   edi ; ***
3058 0000D07D 52                  <1>       push   edx ; **
3059 0000D07E 51                  <1>       push   ecx ; *
3060                              <1>
3061 0000D07F C1E910              <1>       shr    ecx, 16 ; top row
3062 0000D082 C1EA10              <1>       shr    edx, 16 ; bottom row
3063 0000D085 83F918              <1>       cmp    ecx, 24 ; max. 25 rows
3064 0000D088 77DF                <1>       ja     short sysvideo_6
3065 0000D08A 83FA18              <1>       cmp    edx, 24 ; max. 25 rows
3066 0000D08D 77DA                <1>       ja     short sysvideo_6
3067 0000D08F 28CA                <1>       sub    dl, cl
3068 0000D091 72D6                <1>       jc     short sysvideo_6
3069                              <1>
3070 0000D093 88CD                <1>       mov    ch, cl ; top row
3071 0000D095 8A0D[2E520100]      <1>       mov    cl, [ACTIVE_PAGE]
3072 0000D09B BFA00F0000          <1>       mov    edi, 80*25*2
3073 0000D0A0 D3E7                <1>       shl    edi, cl
3074 0000D0A2 81C760700B00        <1>       add    edi, 0B8000h - 80*25*2
3075                              <1>
3076 0000D0A8 88D7                <1>       mov    bh, dl ; row count - 1
3077 0000D0AA 88EA                <1>       mov    dl, ch ; top row
3078 0000D0AC B8A0000000          <1>       mov    eax, 80*2
3079 0000D0B1 F7E2                <1>       mul    edx
3080 0000D0B3 01C7                <1>       add    edi, eax
3081                              <1>
3082 0000D0B5 59                  <1>       pop    ecx ; *
3083 0000D0B6 5A                  <1>       pop    edx ; **
3084 0000D0B7 81E1FFFF0000        <1>       and    ecx, 0FFFFh
3085 0000D0BD 81E2FFFF0000        <1>       and    edx, 0FFFFh
3086 0000D0C3 83F94F              <1>       cmp    ecx, 79 ; max. 80 columns
3087 0000D0C6 77A3                <1>       ja     short sysvideo_7
3088 0000D0C8 83FA4F              <1>       cmp    edx, 79 ; max. 80 columns
3089 0000D0CB 779E                <1>       ja     short sysvideo_7
3090                              <1>
3091 0000D0CD 28CA                <1>       sub    dl, cl
3092 0000D0CF 769A                <1>       jna    short sysvideo_7
3093                              <1>
3094 0000D0D1 0FB6C1              <1>       movzx  eax, cl ; left column
3095 0000D0D4 D0E0                <1>       shl    al, 1  ; column * 2
3096 0000D0D6 01C7                <1>       add    edi, eax
3097                              <1>
3098 0000D0D8 FEC2                <1>       inc    dl ; column count
3099 0000D0DA D0E2                <1>       shl    dl, 1
3100 0000D0DC 88D1                <1>       mov    cl, dl ; column count * 2
3101 0000D0DE B2A0                <1>       mov    dl, 80*2
3102 0000D0E0 58                  <1>       pop    eax ; *** (swap address)
3103 0000D0E1 5E                  <1>       pop    esi ; ****
3104 0000D0E2 FEC7                <1>       inc    bh
3105                              <1>
3106                              <1>       ;mov   edx, 80*2
3107 0000D0E4 B2A0                <1>       mov    dl, 80*2
3108                              <1>       ;
3109 0000D0E6 80FB01              <1>       cmp    bl, 1
3110 0000D0E9 7735                <1>       ja     short sysvideo_11
3111                              <1>
3112                              <1>       ; user to system video/display page window transfer (mode 0)
3113 0000D0EB 21C0                <1>       and    eax, eax ; swap address
3114 0000D0ED 7413                <1>       jz     short sysvideo_10 ; no window swap
3115                              <1>       ; save previous window content in user's buffer (swap address)
3116 0000D0EF 56                  <1>       push   esi ; user buffer
3117 0000D0F0 57                  <1>       push   edi ; beginning address of the window
3118 0000D0F1 89FE                <1>       mov    esi, edi
3119 0000D0F3 89C7                <1>       mov    edi, eax
3120 0000D0F5 E88C170000          <1>       call   transfer_to_user_buffer ; fast transfer
3121 0000D0FA 5F                  <1>       pop    edi
3122 0000D0FB 5E                  <1>       pop    esi
3123 0000D0FC 0F82B1F3FFFF        <1>       jc     sysret
3124                              <1> sysvideo_10:
3125                              <1>       ; user to system video/display page window transfer (mode 0)
3126                              <1>       ; esi =      user buffer
3127 0000D102 E8C9170000          <1>       call   transfer_from_user_buffer ; fast transfer
3128 0000D107 0F82A6F3FFFF        <1>       jc     sysret
3129 0000D10D 010D[64030300]      <1>       add    [u.r0], ecx
3130 0000D113 01D7                <1>       add    edi, edx ; next row
3131 0000D115 01CE                <1>       add    esi, ecx
3132 0000D117 FECF                <1>       dec    bh
3133 0000D119 75E7                <1>       jnz    short sysvideo_10
3134 0000D11B E993F3FFFF          <1>       jmp    sysret
3135                              <1>
3136                              <1> sysvideo_11:
3137                              <1>       ; system to user video/display page window transfer (mode 0)
3138 0000D120 87FE                <1>       xchg   edi, esi
3139                              <1> sysvideo_12:
3140                              <1>       ; esi = beginning address of the window
3141                              <1>       ; edi =      user buffer
```

```
3142 0000D122 E85F170000          <1>       call   transfer_to_user_buffer ; fast transfer
3143 0000D127 0F8286F3FFFF        <1>       jc     sysret
3144 0000D12D 010D[64030300]      <1>       add    [u.r0], ecx
3145 0000D133 01D6                <1>       add    esi, edx ; next row
3146 0000D135 01CF                <1>       add    edi, ecx
3147 0000D137 FECF                <1>       dec    bh
3148 0000D139 75E7                <1>       jnz    short sysvideo_12
3149 0000D13B E973F3FFFF          <1>       jmp    sysret
3150                              <1>
3151                              <1> sysvideo_13:
3152 0000D140 80FF01              <1>       cmp    bh, 1
3153 0000D143 0F871F030000        <1>         ja      sysvideo_38
3154                              <1>       ; BH = 1 = CGA Graphics (0B8000h) data transfers
3155                              <1>
3156 0000D149 20DB                <1>       and    bl, bl
3157 0000D14B 751A                <1>       jnz    short sysvideo_14
3158                              <1>
3159                              <1>       ; BL = 0 = Fill color (color in CL] (32K)
3160                              <1>
3161 0000D14D 88C8                <1>       mov    al, cl
3162 0000D14F B900800000          <1>       mov    ecx, 32768
3163 0000D154 66890D[64030300]    <1>       mov    [u.r0], cx
3164 0000D15B BF00800B00          <1>       mov    edi, 0B8000h
3165 0000D160 F3AB                <1>       rep    stosd
3166 0000D162 E94CF3FFFF          <1>       jmp    sysret
3167                              <1>
3168                              <1> sysvideo_14:
3169 0000D167 80FB01              <1>       cmp    bl, 1
3170 0000D16A 7723                <1>       ja     short sysvideo_16
3171                              <1>
3172 0000D16C 89CE                <1>       mov    esi, ecx ; user buffer
3173                              <1>       ; BL = 1 = user to system video/display page transfer
3174                              <1> sysvideo_15:
3175 0000D16E BF00800B00          <1>       mov    edi, 0B8000h
3176                              <1>       ; edi = video page address
3177 0000D173 B900800000          <1>       mov    ecx, 32768
3178 0000D178 E853170000          <1>       call   transfer_from_user_buffer ; fast transfer
3179 0000D17D 0F8230F3FFFF        <1>       jc     sysret ; [u.r0] = 0
3180 0000D183 66890D[64030300]    <1>       mov    [u.r0], cx
3181 0000D18A E924F3FFFF          <1>       jmp    sysret
3182                              <1>
3183                              <1> sysvideo_16:
3184 0000D18F 80FB02              <1>       cmp    bl, 2
3185 0000D192 7723                <1>       ja     short sysvideo_18
3186                              <1>
3187 0000D194 89CF                <1>       mov    edi, ecx ; user buffer
3188                              <1>       ; BL = 2 = system to user video/display page transfer
3189                              <1> sysvideo_17:
3190 0000D196 BE00800B00          <1>       mov    esi, 0B8000h
3191 0000D19B B900800000          <1>       mov    ecx, 32768
3192 0000D1A0 E8E1160000          <1>       call   transfer_to_user_buffer ; fast transfer
3193 0000D1A5 0F8208F3FFFF        <1>       jc     sysret ; [u.r0] = 0
3194 0000D1AB 66890D[64030300]    <1>       mov    [u.r0], cx
3195 0000D1B2 E9FCF2FFFF          <1>       jmp    sysret
3196                              <1>
3197                              <1> sysvideo_18:
3198 0000D1B7 80FB03              <1>       cmp    bl, 3
3199 0000D1BA 777E                <1>       ja     short sysvideo_23
3200                              <1>
3201                              <1>       ; BL = 3 = NOT bits in window (ECX, EDX)
3202                              <1>
3203 0000D1BC BF00800B00          <1>       mov    edi, 0B8000h
3204 0000D1C1 89FE                <1>       mov    esi, edi
3205                              <1>
3206 0000D1C3 39CA                <1>       cmp    edx, ecx ; bottom-right > top-left ?
3207 0000D1C5 7716                <1>       ja     short sysvideo_20 ; window
3208                              <1>       ; full screen (update)
3209 0000D1C7 B900800000          <1>       mov    ecx, 32768
3210 0000D1CC 66890D[64030300]    <1>       mov    [u.r0], cx
3211                              <1> sysvideo_19:
3212 0000D1D3 F616                <1>       not    byte [esi] ; NOT operation
3213 0000D1D5 46                  <1>       inc    esi
3214 0000D1D6 E2FB                <1>       loop   sysvideo_19
3215 0000D1D8 E9D6F2FFFF          <1>       jmp    sysret
3216                              <1> sysvideo_20:
3217 0000D1DD 0FB7C2              <1>       movzx  eax, dx ; bottom right column
3218 0000D1E0 6629C8              <1>       sub    ax, cx  ; - top left column
3219 0000D1E3 0F82CAF2FFFF        <1>         jb      sysret ; invalid
3220 0000D1E9 6640                <1>       inc    ax ; same column no == 1 column
3221 0000D1EB 50                  <1>       push   eax ; byte count per window row
3222 0000D1EC 52                  <1>       push   edx
3223 0000D1ED BB40010000          <1>       mov    ebx, 320 ; screen width
3224 0000D1F2 89C8                <1>       mov    eax, ecx
3225 0000D1F4 C1E810              <1>       shr    eax, 16  ; top row
3226 0000D1F7 F7E3                <1>       mul    ebx
3227 0000D1F9 6689CA              <1>       mov    dx, cx ; top left column
3228 0000D1FC 01D0                <1>       add    eax, edx
3229 0000D1FE 01C6                <1>       add    esi, eax ; start address
3230 0000D200 59                  <1>       pop    ecx ; edx
3231 0000D201 89C8                <1>       mov    eax, ecx
3232 0000D203 C1E810              <1>       shr    eax, 16 ; bottom row
3233 0000D206 F7E3                <1>       mul    ebx
3234 0000D208 6689CA              <1>       mov    dx, cx ; bottom right column
3235 0000D20B 01D0                <1>       add    eax, edx
3236 0000D20D 01C7                <1>       add    edi, eax ; stop address (included)
3237 0000D20F 5A                  <1>       pop    edx ; byte count per window row
3238 0000D210 81FFFFFF0B00        <1>       cmp    edi, 0BFFFFh
3239 0000D216 0F8797F2FFFF        <1>       ja     sysret
3240 0000D21C 56                  <1>       push   esi
3241 0000D21D 4E                  <1>       dec    esi
3242                              <1> sysvideo_21:
3243 0000D21E 89D1                <1>       mov    ecx, edx
```

```
3244                                    <1> sysvideo_22:
3245 0000D220 46                        <1>         inc     esi
3246 0000D221 F616                      <1>         not     byte [esi]
3247 0000D223 E2FB                      <1>         loop    sysvideo_22
3248 0000D225 01DE                      <1>         add     esi, ebx ; bytes per screen row
3249                                    <1>         ;
3250 0000D227 39FE                      <1>         cmp     esi, edi ; stop address (included in loop)
3251 0000D229 76F3                      <1>         jna     short sysvideo_21
3252 0000D22B 5E                        <1>         pop     esi
3253 0000D22C 29F7                      <1>         sub     edi, esi
3254 0000D22E 66893D[64030300]          <1>         mov     [u.r0], di
3255 0000D235 E979F2FFFF                <1>         jmp     sysret
3256                                    <1>
3257                                    <1> sysvideo_23:
3258 0000D23A 80FB04                    <1>         cmp     bl, 4
3259 0000D23D 0F87A7000000              <1>           ja      sysvideo_26
3260                                    <1>
3261                                    <1>         ; BL = 4 = window copy (system to system)
3262                                    <1>
3263 0000D243 B800800B00                <1>         mov     eax, 0B8000h
3264 0000D248 39C6                      <1>         cmp     esi, eax
3265 0000D24A 0F8263F2FFFF              <1>         jb      sysret
3266 0000D250 39C7                      <1>         cmp     edi, eax
3267 0000D252 0F825BF2FFFF              <1>         jb      sysret
3268 0000D258 6605FF7F                  <1>         add     ax, 7FFFh ; 32767
3269 0000D25C 39C6                      <1>         cmp     esi, eax
3270 0000D25E 0F874FF2FFFF              <1>         ja      sysret
3271 0000D264 39C7                      <1>         cmp     edi, eax
3272 0000D266 0F8747F2FFFF              <1>         ja      sysret
3273                                    <1>
3274 0000D26C 39CA                      <1>         cmp     edx, ecx ; bottom-right > top-left ?
3275 0000D26E 7714                      <1>         ja      short sysvideo_24 ; window
3276                                    <1>         ; full screen copy
3277 0000D270 89C1                      <1>         mov     ecx, eax
3278 0000D272 29F9                      <1>         sub     ecx, edi
3279 0000D274 6641                      <1>         inc     cx
3280 0000D276 66890D[64030300]          <1>         mov     [u.r0], cx
3281 0000D27D F3A4                      <1>         rep     movsb
3282 0000D27F E92FF2FFFF                <1>         jmp     sysret
3283                                    <1> sysvideo_24:
3284 0000D284 0FB7C2                    <1>         movzx   eax, dx ; bottom right column
3285 0000D287 6629C8                    <1>         sub     ax, cx  ; - top left column
3286 0000D28A 0F8223F2FFFF              <1>           jb      sysret ; invalid
3287 0000D290 6640                      <1>         inc     ax ; same column no == 1 column
3288 0000D292 50                        <1>         push    eax ; byte count per window row
3289                                    <1>         ;
3290 0000D293 52                        <1>         push    edx
3291 0000D294 BB40010000                <1>         mov     ebx, 320 ; screen width
3292 0000D299 89C8                      <1>         mov     eax, ecx
3293 0000D29B C1E810                    <1>         shr     eax, 16      ; top row
3294 0000D29E F7E3                      <1>         mul     ebx
3295 0000D2A0 6689CA                    <1>         mov     dx, cx ; top left column
3296 0000D2A3 01D0                      <1>         add     eax, edx
3297 0000D2A5 01C7                      <1>         add     edi, eax ; start address
3298 0000D2A7 01C6                      <1>         add     esi, eax
3299 0000D2A9 59                        <1>         pop     ecx ; edx
3300 0000D2AA 89C8                      <1>         mov     eax, ecx
3301 0000D2AC C1E810                    <1>         shr     eax, 16 ; bottom row
3302 0000D2AF F7E3                      <1>         mul     ebx
3303 0000D2B1 6689CA                    <1>         mov     dx, cx ; bottom right column
3304 0000D2B4 01D0                      <1>         add     eax, edx
3305 0000D2B6 5A                        <1>         pop     edx ; byte count per window row
3306 0000D2B7 0500800B00                <1>         add     eax, 0B8000h
3307 0000D2BC 3DFFFF0B00                <1>         cmp     eax, 0BFFFFh
3308 0000D2C1 0F87ECF1FFFF              <1>         ja      sysret
3309 0000D2C7 57                        <1>         push    edi ; start address
3310 0000D2C8 50                        <1>         push    eax ; stop address (included)
3311                                    <1> sysvideo_25:
3312 0000D2C9 89D1                      <1>         mov     ecx, edx
3313 0000D2CB F3A4                      <1>         rep     movsb
3314 0000D2CD 4F                        <1>         dec     edi
3315 0000D2CE 4E                        <1>         dec     esi
3316 0000D2CF 01DF                      <1>         add     edi, ebx ; bytes per screen row
3317 0000D2D1 01DE                      <1>         add     esi, ebx
3318                                    <1>         ;
3319 0000D2D3 3B3C24                    <1>         cmp     edi, [esp] ; stop addr(included in loop)
3320 0000D2D6 76F1                      <1>         jna     short sysvideo_25
3321 0000D2D8 5B                        <1>         pop     ebx ; stop address
3322 0000D2D9 5F                        <1>         pop     edi ; start address
3323 0000D2DA 29FB                      <1>         sub     ebx, edi
3324 0000D2DC 6643                      <1>         inc     bx
3325 0000D2DE 66891D[64030300]          <1>         mov     [u.r0], bx
3326 0000D2E5 E9C9F1FFFF                <1>         jmp     sysret
3327                                    <1>
3328                                    <1> sysvideo_26:
3329 0000D2EA 80FB05                    <1>         cmp     bl, 5
3330 0000D2ED 0F8795000000              <1>           ja      sysvideo_29
3331                                    <1>
3332                                    <1>         ; BL = 5 = window copy (user to system)
3333                                    <1>
3334 0000D2F3 B800800B00                <1>         mov     eax, 0B8000h
3335 0000D2F8 39C7                      <1>         cmp     edi, eax
3336 0000D2FA 0F82B3F1FFFF              <1>         jb      sysret
3337 0000D300 6605FF7F                  <1>         add     ax, 7FFFh ; 32767
3338 0000D304 39C7                      <1>         cmp     edi, eax
3339 0000D306 0F87A7F1FFFF              <1>         ja      sysret
3340                                    <1>
3341                                    <1>         ; esi = user buffer (in user's memory space)
3342 0000D30C 39CA                      <1>         cmp     edx, ecx ; bottom-right > top-left ?
3343 0000D30E 0F865AFEFFFF              <1>           jna     sysvideo_15 ; full screen copy
3344                                    <1>
3345 0000D314 0FB7C2                    <1>         movzx   eax, dx ; bottom right column
```

```
3346 0000D317 6629C8              <1>       sub    ax, cx  ; - top left column
3347 0000D31A 0F8293F1FFFF        <1>         jb      sysret ; invalid
3348 0000D320 6640                <1>       inc    ax ; same column no == 1 column
3349 0000D322 50                  <1>       push   eax ; byte count per window row
3350                              <1>
3351 0000D323 52                  <1>       push   edx
3352 0000D324 BB40010000          <1>       mov    ebx, 320 ; screen width
3353 0000D329 89C8                <1>       mov    eax, ecx
3354 0000D32B C1E810              <1>       shr    eax, 16      ; top row
3355 0000D32E F7E3                <1>       mul    ebx
3356 0000D330 6689CA              <1>       mov    dx, cx ; top left column
3357 0000D333 01D0                <1>       add    eax, edx
3358 0000D335 01C7                <1>       add    edi, eax ; start address
3359 0000D337 59                  <1>       pop    ecx ; edx
3360 0000D338 89C8                <1>       mov    eax, ecx
3361 0000D33A C1E810              <1>       shr    eax, 16 ; bottom row
3362 0000D33D F7E3                <1>       mul    ebx
3363 0000D33F 6689CA              <1>       mov    dx, cx ; bottom right column
3364 0000D342 01D0                <1>       add    eax, edx
3365 0000D344 5A                  <1>       pop    edx ; byte count per window row
3366 0000D345 0500800B00          <1>       add    eax, 0B8000h
3367 0000D34A 3DFFFF0B00          <1>       cmp    eax, 0BFFFFh
3368 0000D34F 0F875EF1FFFF        <1>       ja     sysret
3369 0000D355 57                  <1>       push   edi ; start address
3370 0000D356 50                  <1>       push   eax ; stop address (included)
3371                              <1> sysvideo_27:
3372 0000D357 89D1                <1>       mov    ecx, edx ; byte count
3373                              <1>       ; user to system video/display page window transfer
3374                              <1>       ; esi =      user buffer
3375 0000D359 E872150000          <1>       call   transfer_from_user_buffer ; fast transfer
3376 0000D35E 7221                <1>       jc     short sysvideo_28
3377 0000D360 010D[64030300]      <1>       add    [u.r0], ecx
3378 0000D366 01DF                <1>       add    edi, ebx ; next row
3379 0000D368 01CE                <1>       add    esi, ecx
3380 0000D36A 3B3C24              <1>       cmp    edi, [esp] ; stop addr(included in loop)
3381 0000D36D 76E8                <1>       jna    short sysvideo_27
3382 0000D36F 5B                  <1>       pop    ebx ; stop address
3383 0000D370 5F                  <1>       pop    edi ; start address
3384 0000D371 29FB                <1>       sub    ebx, edi
3385 0000D373 6643                <1>       inc    bx
3386 0000D375 66891D[64030300]    <1>       mov    [u.r0], bx
3387 0000D37C E932F1FFFF          <1>       jmp    sysret
3388                              <1> sysvideo_28:
3389 0000D381 58                  <1>       pop    eax
3390 0000D382 5A                  <1>       pop    edx
3391 0000D383 E92BF1FFFF          <1>       jmp    sysret
3392                              <1>
3393                              <1> sysvideo_29:
3394 0000D388 80FB06              <1>       cmp    bl, 6
3395 0000D38B 0F8797000000        <1>         ja      sysvideo_32
3396                              <1>
3397                              <1>       ; BL = 6 = window copy (system to user)
3398                              <1>
3399 0000D391 89F7                <1>       mov    edi, esi ; user buffer
3400                              <1>
3401 0000D393 B800800B00          <1>       mov    eax, 0B8000h
3402 0000D398 39C6                <1>       cmp    esi, eax
3403 0000D39A 0F8213F1FFFF        <1>       jb     sysret
3404 0000D3A0 6605FF7F            <1>       add    ax, 7FFFh ; 32767
3405 0000D3A4 39C6                <1>       cmp    esi, eax
3406 0000D3A6 0F8707F1FFFF        <1>       ja     sysret
3407                              <1>
3408                              <1>       ; edi = user buffer (in user's memory space)
3409 0000D3AC 39CA                <1>       cmp    edx, ecx ; bottom-right > top-left ?
3410 0000D3AE 0F86E2FDFFFF        <1>         jna     sysvideo_17 ; full screen copy
3411                              <1>
3412 0000D3B4 0FB7C2              <1>       movzx  eax, dx ; bottom right column
3413 0000D3B7 6629C8              <1>       sub    ax, cx  ; - top left column
3414 0000D3BA 0F82F3F0FFFF        <1>         jb      sysret ; invalid
3415 0000D3C0 6640                <1>       inc    ax ; same column no == 1 column
3416 0000D3C2 50                  <1>       push   eax ; byte count per window row
3417                              <1>
3418 0000D3C3 52                  <1>       push   edx
3419 0000D3C4 BB40010000          <1>       mov    ebx, 320 ; screen width
3420 0000D3C9 89C8                <1>       mov    eax, ecx
3421 0000D3CB C1E810              <1>       shr    eax, 16      ; top row
3422 0000D3CE F7E3                <1>       mul    ebx
3423 0000D3D0 6689CA              <1>       mov    dx, cx ; top left column
3424 0000D3D3 01D0                <1>       add    eax, edx
3425 0000D3D5 01C6                <1>       add    esi, eax ; start address
3426 0000D3D7 59                  <1>       pop    ecx ; edx
3427 0000D3D8 89C8                <1>       mov    eax, ecx
3428 0000D3DA C1E810              <1>       shr    eax, 16 ; bottom row
3429 0000D3DD F7E3                <1>       mul    ebx
3430 0000D3DF 6689CA              <1>       mov    dx, cx ; bottom right column
3431 0000D3E2 01D0                <1>       add    eax, edx
3432 0000D3E4 5A                  <1>       pop    edx ; byte count per window row
3433 0000D3E5 0500800B00          <1>       add    eax, 0B8000h
3434 0000D3EA 3DFFFF0B00          <1>       cmp    eax, 0BFFFFh
3435 0000D3EF 0F87BEF0FFFF        <1>       ja     sysret
3436 0000D3F5 56                  <1>       push   esi ; start address
3437 0000D3F6 50                  <1>       push   eax ; stop address (included)
3438                              <1> sysvideo_30:
3439 0000D3F7 89D1                <1>       mov    ecx, edx ; byte count
3440                              <1>       ; user to system video/display page window transfer
3441                              <1>       ; esi =      user buffer
3442 0000D3F9 E888140000          <1>       call   transfer_to_user_buffer ; fast transfer
3443 0000D3FE 7221                <1>       jc     short sysvideo_31
3444 0000D400 010D[64030300]      <1>       add    [u.r0], ecx
3445 0000D406 01DF                <1>       add    edi, ebx ; next row
3446 0000D408 01CE                <1>       add    esi, ecx
3447 0000D40A 3B3C24              <1>       cmp    edi, [esp] ; stop addr(included in loop)
```

```
3448 0000D40D 76E8              <1>       jna    short sysvideo_30
3449 0000D40F 5B                <1>       pop    ebx ; stop address
3450 0000D410 5F                <1>       pop    edi ; start address
3451 0000D411 29FB              <1>       sub    ebx, edi
3452 0000D413 6643              <1>       inc    bx
3453 0000D415 66891D[64030300]  <1>       mov    [u.r0], bx
3454 0000D41C E992F0FFFF        <1>       jmp    sysret
3455                            <1> sysvideo_31:
3456 0000D421 58                <1>       pop    eax
3457 0000D422 5A                <1>       pop    edx
3458 0000D423 E98BF0FFFF        <1>       jmp    sysret
3459                            <1>
3460                            <1> sysvideo_32:
3461 0000D428 80FB07            <1>       cmp    bl, 7
3462 0000D42B 770F              <1>       ja     short sysvideo_34
3463                            <1>
3464                            <1>       ; BL = 7 = AND display page bytes with CL
3465                            <1>
3466 0000D42D BE00800B00        <1>       mov    esi, 0B8000h
3467 0000D432 B900800000        <1>       mov    ecx, 32768
3468                            <1> sysvideo_33:
3469 0000D437 200E              <1>       and    byte [esi], cl
3470 0000D439 46                <1>       inc    esi
3471 0000D43A E2FB              <1>       loop   sysvideo_33
3472                            <1>
3473                            <1> sysvideo_34:
3474 0000D43C 80FB08            <1>       cmp    bl, 8
3475 0000D43F 770F              <1>       ja     short sysvideo_36
3476                            <1>
3477                            <1>       ; BL = 8 = OR display page bytes with CL
3478                            <1>
3479 0000D441 BE00800B00        <1>       mov    esi, 0B8000h
3480 0000D446 B900800000        <1>       mov    ecx, 32768
3481                            <1> sysvideo_35:
3482 0000D44B 080E              <1>       or     byte [esi], cl
3483 0000D44D 46                <1>       inc    esi
3484 0000D44E E2FB              <1>       loop   sysvideo_35
3485                            <1>
3486                            <1> sysvideo_36:
3487 0000D450 80FB09            <1>       cmp    bl, 9
3488 0000D453 0F875AF0FFFF      <1>         ja      sysret ; nothing to do
3489                            <1>
3490                            <1>       ; BL = 9 = XOR display page bytes with CL
3491                            <1>
3492 0000D459 BE00800B00        <1>       mov    esi, 0B8000h
3493 0000D45E B900800000        <1>       mov    ecx, 32768
3494                            <1> sysvideo_37:
3495 0000D463 300E              <1>       xor    byte [esi], cl
3496 0000D465 46                <1>       inc    esi
3497 0000D466 E2FB              <1>       loop   sysvideo_37
3498                            <1>
3499                            <1> sysvideo_38:
3500 0000D468 80FF02            <1>       cmp    bh, 2
3501 0000D46B 0F8733030000      <1>         ja      sysvideo_64
3502                            <1>       ; BH = 2 = VGA Graphics (0A0000h) data transfers
3503                            <1>
3504 0000D471 88DC              <1>       mov    ah, bl
3505 0000D473 80E30F            <1>       and    bl, 0Fh
3506 0000D476 C0EC04            <1>       shr    ah, 4
3507 0000D479 C1E310            <1>       shl    ebx, 16
3508 0000D47C 66BB4001          <1>       mov    bx, 320 ; 320*200, 320*240
3509 0000D480 20E4              <1>       and    ah, ah
3510 0000D482 7413              <1>       jz     short sysvideo_39
3511 0000D484 66D1E3            <1>       shl    bx, 1 ; 640*200, 640 * 400, 640*480
3512 0000D487 80FC02            <1>       cmp    ah, 2
3513 0000D48A 720B              <1>       jb     short sysvideo_39
3514 0000D48C 0F8721F0FFFF      <1>       ja     sysret ; invalid
3515                            <1>       ; 800*600
3516 0000D492 6681C3A000        <1>       add    bx, 160 ; 800
3517                            <1> sysvideo_39:
3518 0000D497 C1CB10            <1>       ror    ebx, 16
3519                            <1>
3520 0000D49A 20DB              <1>       and    bl, bl
3521 0000D49C 7519              <1>       jnz    short sysvideo_40
3522                            <1>
3523                            <1>       ; BL = 0 = Fill color (color in CL] (64K)
3524                            <1>
3525 0000D49E 88C8              <1>       mov    al, cl
3526 0000D4A0 B900000100        <1>       mov    ecx, 65536
3527 0000D4A5 890D[64030300]    <1>       mov    [u.r0], ecx
3528 0000D4AB BF00000A00        <1>       mov    edi, 0A0000h
3529 0000D4B0 F3AB              <1>       rep    stosd
3530 0000D4B2 E9FCEFFFFF        <1>       jmp    sysret
3531                            <1>
3532                            <1> sysvideo_40:
3533 0000D4B7 80FB01            <1>       cmp    bl, 1
3534 0000D4BA 7722              <1>       ja     short sysvideo_42
3535                            <1>
3536 0000D4BC 89CE              <1>       mov    esi, ecx ; user buffer
3537                            <1>       ; BL = 1 = user to system video/display page transfer
3538                            <1> sysvideo_41:
3539 0000D4BE BF00000A00        <1>       mov    edi, 0A0000h
3540                            <1>       ; edi = video page address
3541 0000D4C3 B900000100        <1>       mov    ecx, 65536
3542 0000D4C8 E803140000        <1>       call   transfer_from_user_buffer ; fast transfer
3543 0000D4CD 0F82E0EFFFFF      <1>       jc     sysret ; [u.r0] = 0
3544 0000D4D3 890D[64030300]    <1>       mov    [u.r0], ecx
3545 0000D4D9 E9D5EFFFFF        <1>       jmp    sysret
3546                            <1>
3547                            <1> sysvideo_42:
3548 0000D4DE 80FB02            <1>       cmp    bl, 2
3549 0000D4E1 7722              <1>       ja     short sysvideo_44
```

```
3550                                <1>
3551 0000D4E3 89CF                 <1>       mov    edi, ecx ; user buffer
3552                                <1>       ; BL = 2 = system to user video/display page transfer
3553                                <1> sysvideo_43:
3554 0000D4E5 BE00000A00           <1>       mov    esi, 0A0000h
3555 0000D4EA B900000100           <1>       mov    ecx, 65536
3556 0000D4EF E892130000           <1>       call   transfer_to_user_buffer ; fast transfer
3557 0000D4F4 0F82B9EFFFFF         <1>       jc     sysret ; [u.r0] = 0
3558 0000D4FA 890D[64030300]       <1>       mov    [u.r0], ecx
3559 0000D500 E9AEEFFFFF           <1>       jmp    sysret
3560                                <1>
3561                                <1> sysvideo_44:
3562 0000D505 80FB03               <1>       cmp    bl, 3
3563 0000D508 777A                 <1>       ja     short sysvideo_49
3564                                <1>
3565                                <1>       ; BL = 3 = NOT bits in window (ECX, EDX)
3566                                <1>
3567 0000D50A BF00000A00           <1>       mov    edi, 0A0000h
3568 0000D50F 89FE                 <1>       mov    esi, edi
3569                                <1>
3570 0000D511 39CA                 <1>       cmp    edx, ecx ; bottom-right > top-left ?
3571 0000D513 770B                 <1>       ja     short sysvideo_45 ; window
3572                                <1>       ; full screen (update)
3573 0000D515 B900000100           <1>       mov    ecx, 65536
3574 0000D51A 890D[64030300]       <1>       mov    [u.r0], ecx
3575                                <1> sysvideo_45:
3576 0000D520 F616                 <1>       not    byte [esi] ; NOT operation
3577 0000D522 46                   <1>       inc    esi
3578 0000D523 E2FB                 <1>       loop   sysvideo_45
3579 0000D525 E989EFFFFF           <1>       jmp    sysret
3580                                <1> sysvideo_46:
3581 0000D52A 0FB7C2               <1>       movzx  eax, dx ; bottom right column
3582 0000D52D 6629C8               <1>       sub    ax, cx  ; - top left column
3583 0000D530 0F827DEFFFFF         <1>         jb      sysret ; invalid
3584 0000D536 6640                 <1>       inc    ax ; same column no == 1 column
3585 0000D538 50                   <1>       push   eax ; byte count per window row
3586 0000D539 52                   <1>       push   edx
3587 0000D53A C1EB10               <1>       shr    ebx, 16 ; 320,640,800 : screen width
3588 0000D53D 89C8                 <1>       mov    eax, ecx
3589 0000D53F C1E810               <1>       shr    eax, 16  ; top row
3590 0000D542 F7E3                 <1>       mul    ebx
3591 0000D544 6689CA               <1>       mov    dx, cx ; top left column
3592 0000D547 01D0                 <1>       add    eax, edx
3593 0000D549 01C6                 <1>       add    esi, eax ; start address
3594 0000D54B 59                   <1>       pop    ecx ; edx
3595 0000D54C 89C8                 <1>       mov    eax, ecx
3596 0000D54E C1E810               <1>       shr    eax, 16 ; bottom row
3597 0000D551 F7E3                 <1>       mul    ebx
3598 0000D553 6689CA               <1>       mov    dx, cx ; bottom right column
3599 0000D556 01D0                 <1>       add    eax, edx
3600 0000D558 01C7                 <1>       add    edi, eax ; stop address (included)
3601 0000D55A 5A                   <1>       pop    edx ; byte count per window row
3602 0000D55B 81FFFFFF0A00         <1>       cmp    edi, 0AFFFFh
3603 0000D561 0F874CEFFFFF         <1>       ja     sysret
3604 0000D567 56                   <1>       push   esi
3605 0000D568 4E                   <1>       dec    esi
3606                                <1> sysvideo_47:
3607 0000D569 89D1                 <1>       mov    ecx, edx
3608                                <1> sysvideo_48:
3609 0000D56B 46                   <1>       inc    esi
3610 0000D56C F616                 <1>       not    byte [esi]
3611 0000D56E E2FB                 <1>       loop   sysvideo_48
3612 0000D570 01DE                 <1>       add    esi, ebx ; bytes per screen row
3613                                <1>       ;
3614 0000D572 39FE                 <1>       cmp    esi, edi ; stop address (included in loop)
3615 0000D574 76F3                 <1>       jna    short sysvideo_47
3616 0000D576 5E                   <1>       pop    esi
3617 0000D577 29F7                 <1>       sub    edi, esi
3618 0000D579 893D[64030300]       <1>       mov    [u.r0], edi
3619 0000D57F E92FEFFFFF           <1>       jmp    sysret
3620                                <1>
3621                                <1> sysvideo_49:
3622 0000D584 80FB04               <1>       cmp    bl, 4
3623 0000D587 0F87A1000000         <1>         ja      sysvideo_52
3624                                <1>
3625                                <1>       ; BL = 4 = window copy (system to system)
3626                                <1>
3627 0000D58D B800000A00           <1>       mov    eax, 0A0000h
3628 0000D592 39C6                 <1>       cmp    esi, eax
3629 0000D594 0F8219EFFFFF         <1>       jb     sysret
3630 0000D59A 39C7                 <1>       cmp    edi, eax
3631 0000D59C 0F8211EFFFFF         <1>       jb     sysret
3632 0000D5A2 6683C0FF             <1>       add    ax, 0FFFFh ; 65535
3633 0000D5A6 39C6                 <1>       cmp    esi, eax
3634 0000D5A8 0F8705EFFFFF         <1>       ja     sysret
3635 0000D5AE 39C7                 <1>       cmp    edi, eax
3636 0000D5B0 0F87FDEEFFFF         <1>       ja     sysret
3637                                <1>
3638 0000D5B6 39CA                 <1>       cmp    edx, ecx ; bottom-right > top-left ?
3639 0000D5B8 7712                 <1>       ja     short sysvideo_50 ; window
3640                                <1>       ; full screen copy
3641 0000D5BA 89C1                 <1>       mov    ecx, eax
3642 0000D5BC 29F9                 <1>       sub    ecx, edi
3643 0000D5BE 41                   <1>       inc    ecx
3644 0000D5BF 890D[64030300]       <1>       mov    [u.r0], ecx
3645 0000D5C5 F3A4                 <1>       rep    movsb
3646 0000D5C7 E9E7EEFFFF           <1>       jmp    sysret
3647                                <1> sysvideo_50:
3648 0000D5CC 0FB7C2               <1>       movzx  eax, dx ; bottom right column
3649 0000D5CF 6629C8               <1>       sub    ax, cx  ; - top left column
3650 0000D5D2 0F82DBEEFFFF         <1>         jb      sysret ; invalid
3651 0000D5D8 6640                 <1>       inc    ax ; same column no == 1 column
```

```
3652 0000D5DA 50                <1>       push   eax ; byte count per window row
3653                            <1>       ;
3654 0000D5DB 52                <1>       push   edx
3655 0000D5DC C1EB10            <1>       shr    ebx, 16 ; 320,640,800 : screen width
3656 0000D5DF 89C8              <1>       mov    eax, ecx
3657 0000D5E1 C1E810            <1>       shr    eax, 16      ; top row
3658 0000D5E4 F7E3              <1>       mul    ebx
3659 0000D5E6 6689CA            <1>       mov    dx, cx ; top left column
3660 0000D5E9 01D0              <1>       add    eax, edx
3661 0000D5EB 01C7              <1>       add    edi, eax ; start address
3662 0000D5ED 01C6              <1>       add    esi, eax
3663 0000D5EF 59                <1>       pop    ecx ; edx
3664 0000D5F0 89C8              <1>       mov    eax, ecx
3665 0000D5F2 C1E810            <1>       shr    eax, 16 ; bottom row
3666 0000D5F5 F7E3              <1>       mul    ebx
3667 0000D5F7 6689CA            <1>       mov    dx, cx ; bottom right column
3668 0000D5FA 01D0              <1>       add    eax, edx
3669 0000D5FC 5A                <1>       pop    edx ; byte count per window row
3670 0000D5FD 0500000A00        <1>       add    eax, 0A0000h
3671 0000D602 3DFFFF0A00        <1>       cmp    eax, 0AFFFFh
3672 0000D607 0F87A6EEFFFF      <1>       ja     sysret
3673 0000D60D 57                <1>       push   edi ; start address
3674 0000D60E 50                <1>       push   eax ; stop address (included)
3675                            <1> sysvideo_51:
3676 0000D60F 89D1              <1>       mov    ecx, edx
3677 0000D611 F3A4              <1>       rep    movsb
3678 0000D613 4F                <1>       dec    edi
3679 0000D614 4E                <1>       dec    esi
3680 0000D615 01DF              <1>       add    edi, ebx ; bytes per screen row
3681 0000D617 01DE              <1>       add    esi, ebx
3682                            <1>       ;
3683 0000D619 3B3C24            <1>       cmp    edi, [esp] ; stop addr(included in loop)
3684 0000D61C 76F1              <1>       jna    short sysvideo_51
3685 0000D61E 5B                <1>       pop    ebx ; stop address
3686 0000D61F 5F                <1>       pop    edi ; start address
3687 0000D620 29FB              <1>       sub    ebx, edi
3688 0000D622 43                <1>       inc    ebx
3689 0000D623 891D[64030300]    <1>       mov    [u.r0], ebx
3690 0000D629 E985EEFFFF        <1>       jmp    sysret
3691                            <1>
3692                            <1> sysvideo_52:
3693 0000D62E 80FB05            <1>       cmp    bl, 5
3694 0000D631 0F8791000000      <1>         ja     sysvideo_55
3695                            <1>
3696                            <1>       ; BL = 5 = window copy (user to system)
3697                            <1>
3698 0000D637 B800000A00        <1>       mov    eax, 0A0000h
3699 0000D63C 39C7              <1>       cmp    edi, eax
3700 0000D63E 0F826FEEFFFF      <1>       jb     sysret
3701 0000D644 6683C0FF          <1>       add    ax, 0FFFFh ; 65535
3702 0000D648 39C7              <1>       cmp    edi, eax
3703 0000D64A 0F8763EEFFFF      <1>       ja     sysret
3704                            <1>
3705                            <1>       ; esi = user buffer (in user's memory space)
3706 0000D650 39CA              <1>       cmp    edx, ecx ; bottom-right > top-left ?
3707 0000D652 0F8666FEFFFF      <1>         jna    sysvideo_41 ; full screen copy
3708                            <1>
3709 0000D658 0FB7C2            <1>       movzx  eax, dx ; bottom right column
3710 0000D65B 6629C8            <1>       sub    ax, cx  ; - top left column
3711 0000D65E 0F824FEEFFFF      <1>         jb     sysret ; invalid
3712 0000D664 6640              <1>       inc    ax ; same column no == 1 column
3713 0000D666 50                <1>       push   eax ; byte count per window row
3714                            <1>
3715 0000D667 52                <1>       push   edx
3716 0000D668 C1EB10            <1>       shr    ebx, 16 ; 320,640,800 : screen width
3717 0000D66B 89C8              <1>       mov    eax, ecx
3718 0000D66D C1E810            <1>       shr    eax, 16      ; top row
3719 0000D670 F7E3              <1>       mul    ebx
3720 0000D672 6689CA            <1>       mov    dx, cx ; top left column
3721 0000D675 01D0              <1>       add    eax, edx
3722 0000D677 01C7              <1>       add    edi, eax ; start address
3723 0000D679 59                <1>       pop    ecx ; edx
3724 0000D67A 89C8              <1>       mov    eax, ecx
3725 0000D67C C1E810            <1>       shr    eax, 16 ; bottom row
3726 0000D67F F7E3              <1>       mul    ebx
3727 0000D681 6689CA            <1>       mov    dx, cx ; bottom right column
3728 0000D684 01D0              <1>       add    eax, edx
3729 0000D686 5A                <1>       pop    edx ; byte count per window row
3730 0000D687 0500000A00        <1>       add    eax, 0A0000h
3731 0000D68C 3DFFFF0A00        <1>       cmp    eax, 0AFFFFh
3732 0000D691 0F871CEEFFFF      <1>       ja     sysret
3733 0000D697 57                <1>       push   edi ; start address
3734 0000D698 50                <1>       push   eax ; stop address (included)
3735                            <1> sysvideo_53:
3736 0000D699 89D1              <1>       mov    ecx, edx ; byte count
3737                            <1>       ; user to system video/display page window transfer
3738                            <1>       ; esi =      user buffer
3739 0000D69B E830120000        <1>       call   transfer_from_user_buffer ; fast transfer
3740 0000D6A0 721F              <1>       jc     short sysvideo_54
3741 0000D6A2 010D[64030300]    <1>       add    [u.r0], ecx
3742 0000D6A8 01DF              <1>       add    edi, ebx ; next row
3743 0000D6AA 01CE              <1>       add    esi, ecx
3744 0000D6AC 3B3C24            <1>       cmp    edi, [esp] ; stop addr(included in loop)
3745 0000D6AF 76E8              <1>       jna    short sysvideo_53
3746 0000D6B1 5B                <1>       pop    ebx ; stop address
3747 0000D6B2 5F                <1>       pop    edi ; start address
3748 0000D6B3 29FB              <1>       sub    ebx, edi
3749 0000D6B5 43                <1>       inc    ebx
3750 0000D6B6 891D[64030300]    <1>       mov    [u.r0], ebx
3751 0000D6BC E9F2EDFFFF        <1>       jmp    sysret
3752                            <1> sysvideo_54:
3753 0000D6C1 58                <1>       pop    eax
```

```
3754 0000D6C2 5A                <1>       pop    edx
3755 0000D6C3 E9EBEDFFFF        <1>       jmp    sysret
3756                            <1>
3757                            <1> sysvideo_55:
3758 0000D6C8 80FB06            <1>       cmp    bl, 6
3759 0000D6CB 0F8793000000      <1>         ja      sysvideo_58
3760                            <1>
3761                            <1>       ; BL = 6 = window copy (system to user)
3762                            <1>
3763 0000D6D1 89F7              <1>       mov    edi, esi ; user buffer
3764                            <1>
3765 0000D6D3 B800000A00        <1>       mov    eax, 0A0000h
3766 0000D6D8 39C6              <1>       cmp    esi, eax
3767 0000D6DA 0F82D3EDFFFF      <1>       jb     sysret
3768 0000D6E0 6683C0FF          <1>       add    ax, 0FFFFh ; 65535
3769 0000D6E4 39C6              <1>       cmp    esi, eax
3770 0000D6E6 0F87C7EDFFFF      <1>       ja     sysret
3771                            <1>
3772                            <1>       ; edi = user buffer (in user's memory space)
3773 0000D6EC 39CA              <1>       cmp    edx, ecx ; bottom-right > top-left ?
3774 0000D6EE 0F86A2FAFFFF      <1>         jna     sysvideo_17 ; full screen copy
3775                            <1>
3776 0000D6F4 0FB7C2            <1>       movzx  eax, dx ; bottom right column
3777 0000D6F7 6629C8            <1>       sub    ax, cx  ; - top left column
3778 0000D6FA 0F82B3EDFFFF      <1>         jb      sysret ; invalid
3779 0000D700 6640              <1>       inc    ax ; same column no == 1 column
3780 0000D702 50                <1>       push   eax ; byte count per window row
3781                            <1>
3782 0000D703 52                <1>       push   edx
3783 0000D704 C1EB10            <1>       shr    ebx, 16 ; 320, 640,800 ; screen width
3784 0000D707 89C8              <1>       mov    eax, ecx
3785 0000D709 C1E810            <1>       shr    eax, 16      ; top row
3786 0000D70C F7E3              <1>       mul    ebx
3787 0000D70E 6689CA            <1>       mov    dx, cx ; top left column
3788 0000D711 01D0              <1>       add    eax, edx
3789 0000D713 01C6              <1>       add    esi, eax ; start address
3790 0000D715 59                <1>       pop    ecx ; edx
3791 0000D716 89C8              <1>       mov    eax, ecx
3792 0000D718 C1E810            <1>       shr    eax, 16 ; bottom row
3793 0000D71B F7E3              <1>       mul    ebx
3794 0000D71D 6689CA            <1>       mov    dx, cx ; bottom right column
3795 0000D720 01D0              <1>       add    eax, edx
3796 0000D722 5A                <1>       pop    edx ; byte count per window row
3797 0000D723 0500000A00        <1>       add    eax, 0A0000h
3798 0000D728 3DFFFF0A00        <1>       cmp    eax, 0AFFFFh
3799 0000D72D 0F8780EDFFFF      <1>       ja     sysret
3800 0000D733 56                <1>       push   esi ; start address
3801 0000D734 50                <1>       push   eax ; stop address (included)
3802                            <1> sysvideo_56:
3803 0000D735 89D1              <1>       mov    ecx, edx ; byte count
3804                            <1>       ; user to system video/display page window transfer
3805                            <1>       ; esi =      user buffer
3806 0000D737 E84A110000        <1>       call   transfer_to_user_buffer ; fast transfer
3807 0000D73C 721F              <1>       jc     short sysvideo_57
3808 0000D73E 010D[64030300]    <1>       add    [u.r0], ecx
3809 0000D744 01DF              <1>       add    edi, ebx ; next row
3810 0000D746 01CE              <1>       add    esi, ecx
3811 0000D748 3B3C24            <1>       cmp    edi, [esp] ; stop addr(included in loop)
3812 0000D74B 76E8              <1>       jna    short sysvideo_56
3813 0000D74D 5B                <1>       pop    ebx ; stop address
3814 0000D74E 5F                <1>       pop    edi ; start address
3815 0000D74F 29FB              <1>       sub    ebx, edi
3816 0000D751 43                <1>       inc    ebx
3817 0000D752 891D[64030300]    <1>       mov    [u.r0], ebx
3818 0000D758 E956EDFFFF        <1>       jmp    sysret
3819                            <1> sysvideo_57:
3820 0000D75D 58                <1>       pop    eax
3821 0000D75E 5A                <1>       pop    edx
3822 0000D75F E94FEDFFFF        <1>       jmp    sysret
3823                            <1>
3824                            <1> sysvideo_58:
3825 0000D764 80FB07            <1>       cmp    bl, 7
3826 0000D767 770F              <1>       ja     short sysvideo_60
3827                            <1>
3828                            <1>       ; BL = 7 = AND display page bytes with CL
3829                            <1>
3830 0000D769 BE00000A00        <1>       mov    esi, 0A0000h
3831 0000D76E B900000100        <1>       mov    ecx, 65536
3832                            <1> sysvideo_59:
3833 0000D773 200E              <1>       and    byte [esi], cl
3834 0000D775 46                <1>       inc    esi
3835 0000D776 E2FB              <1>       loop   sysvideo_59
3836                            <1>
3837                            <1> sysvideo_60:
3838 0000D778 80FB08            <1>       cmp    bl, 8
3839 0000D77B 770F              <1>       ja     short sysvideo_62
3840                            <1>
3841                            <1>       ; BL = 8 = OR display page bytes with CL
3842                            <1>
3843 0000D77D BE00000A00        <1>       mov    esi, 0A0000h
3844 0000D782 B900000100        <1>       mov    ecx, 65536
3845                            <1> sysvideo_61:
3846 0000D787 080E              <1>       or     byte [esi], cl
3847 0000D789 46                <1>       inc    esi
3848 0000D78A E2FB              <1>       loop   sysvideo_61
3849                            <1>
3850                            <1> sysvideo_62:
3851 0000D78C 80FB09            <1>       cmp    bl, 9
3852 0000D78F 0F871EEDFFFF      <1>         ja      sysret ; nothing to do
3853                            <1>
3854                            <1>       ; BL = 9 = XOR display page bytes with CL
3855                            <1>
```

```
3856 0000D795 BE00000A00          <1>       mov    esi, 0A0000h
3857 0000D79A B900000100          <1>       mov    ecx, 65536
3858                              <1> sysvideo_63:
3859 0000D79F 300E                <1>       xor    byte [esi], cl
3860 0000D7A1 46                  <1>       inc    esi
3861 0000D7A2 E2FB                <1>       loop   sysvideo_63
3862                              <1>
3863                              <1> sysvideo_64:
3864 0000D7A4 80FF03              <1>       cmp    bh, 3
3865 0000D7A7 7464                <1>       je     short sysvideo_68
3866 0000D7A9 80FF04              <1>       cmp    bh, 4
3867 0000D7AC 7721                <1>       ja     short sysvideo_65
3868                              <1>
3869                              <1>       ; BH = 4
3870                              <1>       ; Direct User Access for CGA video memory.
3871                              <1>       ; Setup user's page tables for direct access to 0B8000h.
3872                              <1>       ;
3873                              <1>       ; Permission checks are not implemented yet !
3874                              <1>       ; (11/07/2016)
3875                              <1>
3876 0000D7AE B800800B00          <1>       mov    eax, 0B8000h
3877 0000D7B3 B908000000          <1>       mov    ecx, 8 ; 8 pages (8*4K=32K)
3878 0000D7B8 89C3                <1>       mov    ebx, eax ; 12/05/2017 ; virtual = physical
3879 0000D7BA E8E17EFFFF          <1>       call   direct_memory_access
3880 0000D7BF 0F82EEECFFFF        <1>       jc     sysret
3881                              <1>       ; eax = 0B8000h if there is not an error
3882 0000D7C5 A3[64030300]        <1>       mov    [u.r0], eax
3883 0000D7CA E9E4ECFFFF          <1>       jmp    sysret
3884                              <1>
3885                              <1> sysvideo_65:
3886 0000D7CF 80FF05              <1>       cmp    bh, 5
3887 0000D7D2 7721                <1>       ja     short sysvideo_66
3888                              <1>
3889                              <1>       ; BH = 5
3890                              <1>       ; Direct User Access for VGA video memory.
3891                              <1>       ; Setup user's page tables for direct access to 0A0000h.
3892                              <1>       ;
3893                              <1>       ; Permission checks are not implemented yet !
3894                              <1>       ; (11/07/2016)
3895                              <1>
3896 0000D7D4 B800000A00          <1>       mov    eax, 0A0000h
3897 0000D7D9 B910000000          <1>       mov    ecx, 16 ; 16 pages (16*4K=64K)
3898 0000D7DE 89C3                <1>       mov    ebx, eax ; 12/05/2017 ; virtual = physical
3899 0000D7E0 E8BB7EFFFF          <1>       call   direct_memory_access
3900 0000D7E5 0F82C8ECFFFF        <1>       jc     sysret
3901                              <1>       ; eax = 0A0000h if there is not an error
3902 0000D7EB A3[64030300]        <1>       mov    [u.r0], eax
3903 0000D7F0 E9BEECFFFF          <1>       jmp    sysret
3904                              <1>
3905                              <1> sysvideo_66:
3906 0000D7F5 80FF06              <1>       cmp    bh, 6
3907 0000D7F8 7705                <1>       ja     short sysvideo_67
3908                              <1>       ; BH = 6
3909                              <1>       ; Direct User Access for (Super VGA) Linear Frame Buffer.
3910                              <1>       ; Setup user's page tables for direct access to LFB.
3911                              <1>       ;
3912                              <1>       ; Not implemented yet !
3913                              <1>       ; (11/07/2016)
3914 0000D7FA E9B4ECFFFF          <1>       jmp    sysret
3915                              <1>
3916                              <1> sysvideo_67:
3917 0000D7FF 80FF07              <1>       cmp    bh, 7
3918 0000D802 0F87ABECFFFF        <1>       ja     sysret ; invalid !
3919                              <1>
3920                              <1>       ; BH = 7
3921                              <1>       ; Get (Super/Extended VGA) Linear Frame Buffer info.
3922                              <1>       ;
3923                              <1>       ; Not implemented yet !
3924                              <1>       ; (11/07/2016)
3925 0000D808 E9A6ECFFFF          <1>       jmp    sysret
3926                              <1>
3927                              <1> sysvideo_68:
3928                              <1>       ; BH = 3
3929                              <1>       ; Super VGA, LINEAR FRAME BUFFER data transfers
3930                              <1>       ; Not implemented for yet ! (11/07/2016)
3931 0000D80D E9A1ECFFFF          <1>       jmp    sysret
3932                              <1>
3933                              <1> syslink:
3934                              <1>       ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
3935                              <1>       ; temporary !
3936 0000D812 B801000000          <1>       mov    eax, ERR_INV_FNUMBER ; 'invalid function number !'
3937 0000D817 A3[C8030300]        <1>         mov     [u.error], eax
3938 0000D81C A3[64030300]        <1>         mov     [u.r0], eax
3939 0000D821 E96DECFFFF          <1>       jmp    error
3940                              <1>
3941                              <1> isdir:
3942                              <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
3943                              <1>       ; 04/05/2013 - 02/08/2013 (Retro UNIX 8086 v1)
3944                              <1>       ;
3945                              <1>       ; 'isdir' check to see if the i-node whose i-number is in r1
3946                              <1>       ;  is a directory. If it is, an error occurs, because 'isdir'
3947                              <1>       ;  called by syslink and sysunlink to make sure directories
3948                              <1>       ;  are not linked. If the user is the super user (u.uid=0),
3949                              <1>       ; 'isdir' does not bother checking. The current i-node
3950                              <1>       ;  is not disturbed.
3951                              <1>       ;
3952                              <1>       ; INPUTS ->
3953                              <1>       ;    r1 - contains the i-number whose i-node is being checked.
3954                              <1>       ;    u.uid - user id
3955                              <1>       ; OUTPUTS ->
3956                              <1>       ;    r1 - contains current i-number upon exit
3957                              <1>       ;       (current i-node back in core)
```

```
                                <1>       ;
                                <1>       ; ((AX = R1))
                                <1>       ;
                                <1>         ; ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
                                <1>       ;
                                <1>
                                <1>       ; / if the i-node whose i-number is in r1 is a directory
                                <1>       ; / there is an error unless super user made the call
                                <1>
0000D826 803D[B0030300]00       <1>       cmp    byte [u.uid], 0
                                <1>              ; tstb u.uid / super user
0000D82D 762D                   <1>       jna    short isdir1
                                <1>              ; beq 1f / yes, don't care
0000D82F 66FF35[51040300]       <1>       push   word [ii]
                                <1>              ; mov ii,-(sp) / put current i-number on stack
0000D836 E85E190000             <1>       call   iget
                                <1>              ; jsr r0,iget / get i-node into core (i-number in r1)
0000D83B 66F705[00000300]00-    <1>       test   word [i.flgs], 4000h ; Bit 14 : Directory flag
0000D843 40                     <1>
                                <1>              ; bit $40000,i.flgs / is it a directory
                                <1>       ;jnz   error
                                <1>              ; bne error9 / yes, error
0000D844 740F                   <1>       jz     short isdir0
0000D846 C705[C8030300]0B00-    <1>       mov    dword [u.error], ERR_NOT_FILE  ; 11 ; ERR_DIR_ACCESS
0000D84E 0000                   <1>
                                <1>                                ; 'permission denied !' error
                                <1>       ; pop  ax
0000D850 E93EECFFFF             <1>       jmp    error
                                <1> isdir0:
0000D855 6658                   <1>       pop    ax
                                <1>              ; mov (sp)+,r1 / no, put current i-number in r1 (ii)
0000D857 E83D190000             <1>       call   iget
                                <1>              ; jsr r0,iget / get it back in
                                <1> isdir1: ; 1:
0000D85C C3                     <1>       retn
                                <1>              ; rts r0
                                <1>
                                <1> sysunlink:
                                <1>       ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; temporary !
0000D85D B801000000             <1>       mov    eax, ERR_INV_FNUMBER ; 'invalid function number !'
0000D862 A3[C8030300]           <1>         mov     [u.error], eax
0000D867 A3[64030300]           <1>         mov     [u.r0], eax
0000D86C E922ECFFFF             <1>       jmp    error
                                <1> mkdir:
                                <1>       ; 04/12/2015 (14 byte directory names)
                                <1>       ; 12/10/2015
                                <1>       ; 17/06/2015 (Retro UNIX 386 v1 - Beginning)
                                <1>       ; 29/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
                                <1>       ;
                                <1>       ; 'mkdir' makes a directory entry from the name pointed to
                                <1>       ; by u.namep into the current directory.
                                <1>       ;
                                <1>       ; INPUTS ->
                                <1>       ;    u.namep - points to a file name
                                <1>       ;                 that is about to be a directory entry.
                                <1>       ;    ii - current directory's i-number.
                                <1>       ; OUTPUTS ->
                                <1>       ;    u.dirbuf+2 - u.dirbuf+10 - contains file name.
                                <1>       ;    u.off - points to entry to be filled
                                <1>       ;           in the current directory
                                <1>       ;    u.base - points to start of u.dirbuf.
                                <1>       ;    r1 - contains i-number of current directory
                                <1>       ;
                                <1>       ; ((AX = R1)) output
                                <1>       ;
                                <1>       ;    (Retro UNIX Prototype : 11/11/2012, UNIXCOPY.ASM)
                                <1>         ;    ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
                                <1>       ;
                                <1>
                                <1>       ; 17/06/2015 - 32 bit modifications (Retro UNIX 386 v1)
0000D871 31C0                   <1>       xor    eax, eax
0000D873 BF[9A030300]           <1>       mov     edi, u.dirbuf+2
0000D878 89FE                   <1>       mov    esi, edi
0000D87A AB                     <1>       stosd
0000D87B AB                     <1>       stosd
                                <1>       ; 04/12/2015 (14 byte directory names)
0000D87C AB                     <1>       stosd
0000D87D 66AB                   <1>       stosw
                                <1>              ; jsr r0,copyz; u.dirbuf+2; u.dirbuf+10. / clear this
0000D87F 89F7                   <1>       mov    edi, esi ; offset to u.dirbuf
                                <1>       ; 12/10/2015 ([u.namep] -> ebp)
                                <1>       ;mov   ebp, [u.namep]
0000D881 E829040000             <1>       call   trans_addr_nmbp ; convert virtual address to physical
                                <1>              ; esi = physical address (page start + offset)
                                <1>              ; ecx = byte count in the page (1 - 4096)
                                <1>       ; edi = offset to u.dirbuf (edi is not modified in trans_addr_nm)
                                <1>              ; mov u.namep,r2 / r2 points to name of directory entry
                                <1>              ; mov $u.dirbuf+2,r3 / r3 points to u.dirbuf+2
                                <1> mkdir_1: ; 1:
0000D886 45                     <1>       inc    ebp ; 12/10/2015
                                <1>       ;
                                <1>       ; / put characters in the directory name in u.dirbuf+2 - u.dirbuf+10
                                <1>        ; 01/08/2013
0000D887 AC                     <1>       lodsb
                                <1>              ; movb (r2)+,r1 / move character in name to r1
0000D888 20C0                   <1>       and    al, al
0000D88A 7427                   <1>       jz     short mkdir_3
                                <1>              ; beq 1f / if null, done
0000D88C 3C2F                   <1>       cmp    al, '/'
                                <1>              ; cmp r1,$'/ / is it a "/"?
0000D88E 7414                   <1>       je     short mkdir_err
```

```
4058                                    <1>       ;je    error
4059                                    <1>              ; beq error9 / yes, error
4060                                    <1>       ; 12/10/2015
4061 0000D890 6649                      <1>       dec    cx
4062 0000D892 7505                      <1>       jnz    short mkdir_2
4063                                    <1>       ; 12/10/2015 ([u.namep] -> ebp)
4064 0000D894 E81C040000                <1>       call   trans_addr_nm ; convert virtual address to physical
4065                                    <1>              ; esi = physical address (page start + offset)
4066                                    <1>              ; ecx = byte count in the page
4067                                    <1>       ; edi = offset to u.dirbuf (edi is not modified in trans_addr_nm)
4068                                    <1> mkdir_2:
4069 0000D899 81FF[A8030300]            <1>       cmp     edi, u.dirbuf+16 ; ; 04/12/2015 (10 -> 16)
4070                                    <1>              ; cmp r3,$u.dirbuf+10. / have we reached the last slot for
4071                                    <1>                                     ; / a char?
4072 0000D89F 74E5                      <1>       je     short mkdir_1
4073                                    <1>              ; beq 1b / yes, go back
4074 0000D8A1 AA                        <1>       stosb
4075                                    <1>              ; movb r1,(r3)+ / no, put the char in the u.dirbuf
4076 0000D8A2 EBE2                      <1>       jmp    short mkdir_1
4077                                    <1>              ; br 1b / get next char
4078                                    <1> mkdir_err:
4079                                    <1>       ; 17/06/2015
4080 0000D8A4 C705[C8030300]1300-       <1>       mov    dword [u.error], ERR_NOT_DIR ; 'not a valid directory !'
4080 0000D8AC 0000                      <1>
4081 0000D8AE E9E0EBFFFF                <1>       jmp    error
4082                                    <1>
4083                                    <1> mkdir_3: ; 1:
4084 0000D8B3 A1[78030300]              <1>       mov    eax, [u.dirp]
4085 0000D8B8 A3[80030300]              <1>       mov    [u.off], eax
4086                                    <1>              ; mov u.dirp,u.off / pointer to empty current directory
4087                                    <1>                                  ; / slot to u.off
4088                                    <1> wdir: ; 29/04/2013
4089 0000D8BD C705[84030300]-           <1>         mov     dword [u.base], u.dirbuf
4089 0000D8C3 [98030300]                <1>
4090                                    <1>             ; mov $u.dirbuf,u.base / u.base points to created file name
4091 0000D8C7 C705[88030300]1000-       <1>         mov     dword [u.count], 16 ; 04/12/2015 (10 -> 16)
4091 0000D8CF 0000                      <1>
4092                                    <1>             ; mov $10.,u.count / u.count = 10
4093 0000D8D1 66A1[51040300]            <1>       mov    ax, [ii]
4094                                    <1>              ; mov ii,r1 / r1 has i-number of current directory
4095 0000D8D7 B201                      <1>       mov    dl, 1 ; owner flag mask ; RETRO UNIX 8086 v1 modification !
4096 0000D8D9 E8C1180000                <1>       call   access
4097                                    <1>              ; jsr r0,access; 1 / get i-node and set its file up
4098                                    <1>                                 ; / for writing
4099                                    <1>       ; AX = i-number of current directory
4100                                    <1>       ; 01/08/2013
4101 0000D8DE FE05[C6030300]            <1>       inc     byte [u.kcall] ; the caller is 'mkdir' sign
4102 0000D8E4 E8FF110000                <1>       call   writei
4103                                    <1>              ; jsr r0,writei / write into directory
4104 0000D8E9 C3                        <1>       retn
4105                                    <1>              ; rts r0
4106                                    <1>
4107                                    <1> sysexec:
4108                                    <1>       ; 04/01/2017
4109                                    <1>       ; 24/10/2016
4110                                    <1>       ; 24/04/2016 - TRDOS 386 (TRDOS v2.0)
4111                                    <1>       ; 23/06/2015 - 23/10/2015 (Retro UNIX 386 v1)
4112                                    <1>       ; 03/06/2013 - 06/12/2013 (Retro UNIX 8086 v1)
4113                                    <1>       ;
4114                                    <1>       ; 'sysexec' initiates execution of a file whose path name if
4115                                    <1>       ; pointed to by 'name' in the sysexec call.
4116                                    <1>       ; 'sysexec' performs the following operations:
4117                                    <1>       ;    1. obtains i-number of file to be executed via 'namei'.
4118                                    <1>       ;    2. obtains i-node of file to be exceuted via 'iget'.
4119                                    <1>       ;    3. sets trap vectors to system routines.
4120                                    <1>       ;    4. loads arguments to be passed to executing file into
4121                                    <1>       ;      highest locations of user's core
4122                                    <1>       ;    5. puts pointers to arguments in locations immediately
4123                                    <1>       ;      following arguments.
4124                                    <1>       ;    6.      saves number of arguments in next location.
4125                                    <1>       ;    7. intializes user's stack area so that all registers
4126                                    <1>       ;      will be zeroed and the PS is cleared and the PC set
4127                                    <1>       ;      to core when 'sysret' restores registers
4128                                    <1>       ;      and does an rti.
4129                                    <1>       ;    8. inializes u.r0 and u.sp
4130                                    <1>       ;    9. zeros user's core down to u.r0
4131                                    <1>       ;   10.      reads executable file from storage device into core
4132                                    <1>       ;      starting at location 'core'.
4133                                    <1>       ;   11.      sets u.break to point to end of user's code with
4134                                    <1>       ;      data area appended.
4135                                    <1>       ;   12.      calls 'sysret' which returns control at location
4136                                    <1>       ;      'core' via 'rti' instruction.
4137                                    <1>       ;
4138                                    <1>       ; Calling sequence:
4139                                    <1>       ;      sysexec; namep; argp
4140                                    <1>       ; Arguments:
4141                                    <1>       ;      namep - points to pathname of file to be executed
4142                                    <1>       ;      argp  - address of table of argument pointers
4143                                    <1>       ;      argp1... argpn - table of argument pointers
4144                                    <1>       ;      argp1:<...0> ... argpn:<...0> - argument strings
4145                                    <1>       ; Inputs: (arguments)
4146                                    <1>       ; Outputs: -
4147                                    <1>       ; ............................................................
4148                                    <1>       ;
4149                                    <1>       ; Retro UNIX 386 v1 modification:
4150                                    <1>       ;      User application runs in it's own virtual space
4151                                    <1>       ;      which is izolated from kernel memory (and other
4152                                    <1>       ;      memory pages) via 80386   paging in ring 3
4153                                    <1>       ;      privilige mode. Virtual start address is always 0.
4154                                    <1>       ;      User's core memory starts at linear address 400000h
4155                                    <1>       ;      (the end of the 1st 4MB).
4156                                    <1>       ;
```

```
4157                                    <1>       ; Retro UNIX 8086 v1 modification:
4158                                    <1>       ;     user/application segment and system/kernel segment
4159                                    <1>       ;     are different and sysenter/sysret/sysrele routines
4160                                    <1>       ;     are different (user's registers are saved to
4161                                    <1>       ;     and then restored from system's stack.)
4162                                    <1>       ;
4163                                    <1>       ;     NOTE: Retro UNIX 8086 v1 'arg2' routine gets these
4164                                    <1>       ;           arguments which were in these registers;
4165                                    <1>       ;           but, it returns by putting the 1st argument
4166                                    <1>       ;           in 'u.namep' and the 2nd argument
4167                                    <1>       ;           on top of stack. (1st argument is offset of the
4168                                    <1>       ;           file/path name in the user's program segment.)
4169                                    <1>
4170                                    <1>       ;call  arg2
4171                                    <1>       ; * name - 'u.namep' points to address of file/path name
4172                                    <1>       ;          in the user's program segment ('u.segmnt')
4173                                    <1>       ;          with offset in BX register (as sysopen argument 1).
4174                                    <1>       ; * argp - sysexec argument 2 is in CX register
4175                                    <1>       ;          which is on top of stack.
4176                                    <1>       ;
4177                                    <1>             ; jsr r0,arg2 / arg0 in u.namep,arg1 on top of stack
4178                                    <1>
4179                                    <1>       ; 23/06/2015 (32 bit modifications)
4180                                    <1>
4181 0000D8EA 891D[7C030300]           <1>       mov    [u.namep], ebx ; argument 1
4182                                    <1>         ; 18/10/2015
4183 0000D8F0 890D[4C040300]           <1>       mov     [argv], ecx  ; * ; argument 2
4184 0000D8F6 E86F020000               <1>       call   namei
4185                                    <1>             ; jsr r0,namei / namei returns i-number of file
4186                                    <1>                         ; / named in sysexec call in r1
4187                                    <1>       ;jc    error
4188                                    <1>             ; br error9
4189 0000D8FB 731E                     <1>       jnc    short sysexec_0
4190                                    <1>       ;
4191                                    <1>       ; 'file not found !' error
4192 0000D8FD C705[C8030300]0C00-      <1>       mov    dword [u.error], ERR_FILE_NOT_FOUND
4192 0000D905 0000                     <1>
4193 0000D907 E987EBFFFF               <1>       jmp    error
4194                                    <1> sysexec_not_exf:
4195                                    <1>       ; 'not executable file !' error
4196 0000D90C C705[C8030300]1600-      <1>       mov    dword [u.error], ERR_NOT_EXECUTABLE
4196 0000D914 0000                     <1>
4197 0000D916 E978EBFFFF               <1>       jmp    error
4198                                    <1> sysexec_0:
4199 0000D91B E879180000               <1>       call   iget
4200                                    <1>             ; jsr r0,iget / get i-node for file to be executed
4201 0000D920 66F705[00000300]10-      <1>         test    word [i.flgs], 10h
4201 0000D928 00                       <1>
4202                                    <1>             ; bit $20,i.flgs / is file executable
4203 0000D929 74E1                     <1>       jz     short sysexec_not_exf
4204                                    <1>       ;jz    error
4205                                    <1>             ; beq error9
4206                                    <1>       ;;
4207 0000D92B E86B180000               <1>       call   iopen
4208                                    <1>             ; jsr r0,iopen / gets i-node for file with i-number
4209                                    <1>                         ; / given in r1 (opens file)
4210                                    <1>       ; AX = i-number of the file
4211 0000D930 66F705[00000300]20-      <1>       test   word [i.flgs], 20h
4211 0000D938 00                       <1>
4212                                    <1>             ; bit $40,i.flgs / test user id on execution bit
4213 0000D939 7415                     <1>       jz     short sysexec_1
4214                                    <1>             ; beq 1f
4215 0000D93B 803D[B0030300]00         <1>       cmp    byte [u.uid], 0 ; 02/08/2013
4216                                    <1>             ; tstb u.uid / test user id
4217 0000D942 760C                     <1>       jna    short sysexec_1
4218                                    <1>             ; beq 1f / super user
4219 0000D944 8A0D[03000300]           <1>       mov    cl, [i.uid]
4220 0000D94A 880D[B0030300]           <1>       mov    [u.uid], cl ; 02/08/2013
4221                                    <1>             ; movb i.uid,u.uid / put user id of owner of file
4222                                    <1>                           ; / as process user id
4223                                    <1> sysexec_1:
4224                                    <1>       ; 18/10/2215
4225                                    <1>       ; 10/10/2015
4226                                    <1>       ; 24/07/2015
4227                                    <1>       ; 21/07/2015
4228                                    <1>       ; 25/06/2015
4229                                    <1>       ; 24/06/2015
4230                                    <1>         ; Moving arguments to the end of [u.upage]
4231                                    <1>       ; (by regarding page borders in user's memory space)
4232                                    <1>       ;
4233                                    <1>       ; 10/10/2015
4234                                    <1>       ; 21/07/2015
4235 0000D950 89E5                     <1>       mov    ebp, esp ; (**)
4236                                    <1>       ; 18/10/2015
4237 0000D952 89EF                     <1>       mov    edi, ebp
4238 0000D954 B900010000               <1>       mov    ecx, MAX_ARG_LEN ; 256
4239                                    <1>       ;sub   edi, MAX_ARG_LEN ; 256
4240 0000D959 29CF                     <1>       sub    edi, ecx
4241 0000D95B 89FC                     <1>       mov    esp, edi
4242 0000D95D 31C0                     <1>       xor    eax, eax
4243 0000D95F A3[8C030300]             <1>       mov    [u.nread], eax ; 0
4244 0000D964 49                       <1>       dec    ecx ; 256 - 1
4245 0000D965 890D[88030300]           <1>       mov    [u.count], ecx ; MAX_ARG_LEN - 1 ; 255
4246                                    <1>       ;mov   dword [u.count], MAX_ARG_LEN - 1 ; 255
4247                                    <1> sysexec_2:
4248 0000D96B 8B35[4C040300]           <1>       mov    esi, [argv] ; 18/10/2015
4249 0000D971 E866000000               <1>       call   get_argp
4250 0000D976 B904000000               <1>       mov    ecx, 4 ; mov ecx, 4
4251                                    <1> sysexec_3:
4252 0000D97B 21C0                     <1>       and    eax, eax
4253 0000D97D 0F84E3070000             <1>         jz      sysexec_6
4254                                    <1>       ; 18/10/2015
```

```
4255 0000D983 010D[4C040300]      <1>       add    [argv], ecx ; 4
4256 0000D989 66FF05[4A040300]    <1>       inc    word [argc]
4257                              <1>       ;
4258 0000D990 A3[84030300]        <1>       mov    [u.base], eax
4259                              <1>       ; 23/10/2015
4260 0000D995 66C705[C4030300]00- <1>       mov    word [u.pcount], 0
4260 0000D99D 00                  <1>
4261                              <1> sysexec_4:
4262 0000D99E E8830E0000          <1>       call   cpass ; get a character from user's core memory
4263 0000D9A3 750E                <1>         jnz      short sysexec_5
4264                              <1>              ; (max. 255 chars + null)
4265                              <1>       ; 18/10/2015
4266 0000D9A5 28C0                <1>       sub    al, al
4267 0000D9A7 AA                  <1>       stosb
4268 0000D9A8 FF05[8C030300]      <1>       inc    dword [u.nread]
4269 0000D9AE E9B3070000          <1>       jmp    sysexec_6 ; 24/04/2016
4270                              <1> sysexec_5:
4271 0000D9B3 AA                  <1>       stosb
4272 0000D9B4 20C0                <1>       and    al, al
4273 0000D9B6 75E6                <1>       jnz    short sysexec_4
4274 0000D9B8 B904000000          <1>       mov    ecx, 4
4275 0000D9BD 390D[48040300]      <1>       cmp    [ncount], ecx ; 4
4276 0000D9C3 72A6                <1>       jb     short sysexec_2
4277 0000D9C5 8B35[44040300]      <1>       mov    esi, [nbase]
4278 0000D9CB 010D[44040300]      <1>       add    [nbase], ecx ; 4
4279 0000D9D1 66290D[48040300]    <1>       sub    [ncount], cx
4280 0000D9D8 8B06                <1>       mov    eax, [esi]
4281 0000D9DA EB9F                <1>       jmp    short sysexec_3
4282                              <1>
4283                              <1> get_argp:
4284                              <1>       ; 18/10/2015 (nbase, ncount)
4285                              <1>       ; 21/07/2015
4286                              <1>       ; 24/06/2015 (Retro UNIX 386 v1)
4287                              <1>       ; Get (virtual) address of argument from user's core memory
4288                              <1>       ;
4289                              <1>       ; INPUT:
4290                              <1>       ;     esi = virtual address of argument pointer
4291                              <1>       ; OUTPUT:
4292                              <1>       ;     eax = virtual address of argument
4293                              <1>       ;
4294                              <1>       ; Modified registers: EAX, EBX, ECX, EDX, ESI
4295                              <1>       ;
4296 0000D9DC 833D[BC030300]00    <1>       cmp     dword [u.ppgdir], 0 ; /etc/init ?
4297                              <1>                                   ; (the caller is kernel)
4298 0000D9E3 7667                <1>         jna     short get_argpk
4299                              <1>       ;
4300 0000D9E5 89F3                <1>       mov    ebx, esi
4301 0000D9E7 E8A278FFFF          <1>       call   get_physical_addr ; get physical address
4302 0000D9EC 0F8289000000        <1>         jc     get_argp_err
4303 0000D9F2 A3[44040300]        <1>       mov    [nbase], eax ; physical address
4304 0000D9F7 66890D[48040300]    <1>       mov    [ncount], cx ; remain byte count in page (1-4096)
4305 0000D9FE B804000000          <1>       mov    eax, 4 ; 21/07/2015
4306 0000DA03 6639C1              <1>       cmp    cx, ax ; 4
4307 0000DA06 735D                <1>       jnb    short get_argp2
4308 0000DA08 89F3                <1>       mov    ebx, esi
4309 0000DA0A 01CB                <1>       add    ebx, ecx
4310 0000DA0C E87D78FFFF          <1>       call   get_physical_addr ; get physical address
4311 0000DA11 7268                <1>       jc     short get_argp_err
4312                              <1>       ;push  esi
4313 0000DA13 89C6                <1>       mov    esi, eax
4314 0000DA15 66870D[48040300]    <1>       xchg   cx, [ncount]
4315 0000DA1C 8735[44040300]      <1>       xchg   esi, [nbase]
4316 0000DA22 B504                <1>       mov    ch, 4
4317 0000DA24 28CD                <1>       sub    ch, cl
4318                              <1> get_argp0:
4319 0000DA26 AC                  <1>       lodsb
4320 0000DA27 6650                <1>       push   ax
4321 0000DA29 FEC9                <1>       dec    cl
4322 0000DA2B 75F9                <1>         jnz     short get_argp0
4323 0000DA2D 8B35[44040300]      <1>       mov    esi, [nbase]
4324                              <1>       ; 21/07/2015
4325 0000DA33 0FB6C5              <1>       movzx  eax, ch
4326 0000DA36 0105[44040300]      <1>       add    [nbase], eax
4327 0000DA3C 662905[48040300]    <1>       sub    [ncount], ax
4328                              <1> get_argp1:
4329 0000DA43 AC                  <1>       lodsb
4330 0000DA44 FECD                <1>       dec    ch
4331 0000DA46 743D                <1>         jz      short get_argp3
4332 0000DA48 6650                <1>         push ax
4333 0000DA4A EBF7                <1>       jmp     short get_argp1
4334                              <1> get_argpk:
4335                              <1>       ; Argument is in kernel's memory space
4336 0000DA4C 66C705[48040300]00- <1>       mov    word [ncount], PAGE_SIZE ; 4096
4336 0000DA54 10                  <1>
4337 0000DA55 8935[44040300]      <1>       mov    [nbase], esi
4338 0000DA5B 8305[44040300]04    <1>       add    dword [nbase], 4
4339 0000DA62 8B06                <1>       mov    eax, [esi] ; virtual addr. = physcal addr.
4340 0000DA64 C3                  <1>       retn
4341                              <1> get_argp2:
4342                              <1>       ; 21/07/2015
4343                              <1>       ;mov   eax, 4
4344 0000DA65 8B15[44040300]      <1>       mov    edx, [nbase] ; 18/10/2015
4345 0000DA6B 0105[44040300]      <1>       add    [nbase], eax
4346 0000DA71 662905[48040300]    <1>       sub    [ncount], ax
4347                              <1>       ;
4348 0000DA78 8B02                <1>       mov    eax, [edx]
4349 0000DA7A C3                  <1>       retn
4350                              <1> get_argp_err:
4351 0000DA7B A3[C8030300]        <1>       mov    [u.error], eax
4352 0000DA80 E90EEAFFFF          <1>       jmp    error
4353                              <1> get_argp3:
4354 0000DA85 B103                <1>       mov    cl, 3
```

```
                                <1> get_argp4:
0000DA87 C1E008                 <1>       shl    eax, 8
0000DA8A 665A                   <1>       pop    dx
0000DA8C 88D0                   <1>       mov    al, dl
0000DA8E E2F7                   <1>         loop    get_argp4
                                <1>       ;pop   esi
0000DA90 C3                     <1>       retn
                                <1>
                                <1> sysstat:
                                <1>       ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; temporary !
0000DA91 B801000000             <1>       mov    eax, ERR_INV_FNUMBER ; 'invalid function number !'
0000DA96 A3[C8030300]           <1>         mov     [u.error], eax
0000DA9B A3[64030300]           <1>         mov     [u.r0], eax
0000DAA0 E9EEE9FFFF             <1>       jmp    error
                                <1>
                                <1> sysfstat:
                                <1>       ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; temporary !
0000DAA5 B801000000             <1>       mov    eax, ERR_INV_FNUMBER ; 'invalid function number !'
0000DAAA A3[C8030300]           <1>         mov     [u.error], eax
0000DAAF A3[64030300]           <1>         mov     [u.r0], eax
0000DAB4 E9DAE9FFFF             <1>       jmp    error
                                <1>
                                <1> fclose:
                                <1>       ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>       ;
                                <1>       ; 18/06/2015 (Retro UNIX 386 v1 - Beginning)
                                <1>       ;            (32 bit offset pointer modification)
                                <1>       ; 19/04/2013 - 12/01/2014 (Retro UNIX 8086 v1)
                                <1>       ;
                                <1>       ; Given the file descriptor (index to the u.fp list)
                                <1>       ; 'fclose' first gets the i-number of the file via 'getf'.
                                <1>       ; If i-node is active (i-number > 0) the entry in
                                <1>       ; u.fp list is cleared. If all the processes that opened
                                <1>       ; that file close it, then fsp etry is freed and the file
                                <1>       ; is closed. If not a return is taken.
                                <1>       ; If the file has been deleted while open, 'anyi' is called
                                <1>       ; to see anyone else has it open, i.e., see if it is appears
                                <1>       ; in another entry in the fsp table. Upon return from 'anyi'
                                <1>       ; a check is made to see if the file is special.
                                <1>       ;
                                <1>       ; INPUTS ->
                                <1>       ;    r1 - contains the file descriptor (value=0,1,2...)
                                <1>       ;    u.fp - list of entries in the fsp table
                                <1>       ;    fsp - table of entries (4 words/entry) of open files.
                                <1>       ; OUTPUTS ->
                                <1>       ;    r1 - contains the same file descriptor
                                <1>       ;    r2 - contains i-number
                                <1>       ;
                                <1>       ; ((AX = R1))
                                <1>       ; ((Modified registers: eDX, eBX, eCX, eSI, eDI, eBP))
                                <1>       ;
                                <1>       ; Retro UNIX 8086 v1 modification : CF = 1
                                <1>       ;             if i-number of the file is 0. (error)
                                <1>       ;
                                <1>       ; TRDOS 386 (06/10/2016)
                                <1>       ;
                                <1>       ; INPUT:
                                <1>       ;     EAX = File Handle (File Descriptor, File Index)
                                <1>       ;
                                <1>       ; OUTPUT:
                                <1>       ;     CF = 1 -> File not open !
                                <1>       ;     CF = 0 -> OK!
                                <1>       ;          EBX = File Number (System)
                                <1>       ;          [cdev] = Logical DOS Drive Number
                                <1>       ;          EAX = File Handle/Number (user)
                                <1>       ;
                                <1>       ; Modified Registers: EBX
                                <1>
0000DAB9 50                     <1>       push   eax ; File handle
                                <1>
0000DABA E846000000             <1>       call   getf
0000DABF 0F82C01F0000           <1>       jc     device_close ; eax = device number
                                <1>
0000DAC5 80BB[2A630100]01       <1>       cmp    byte [ebx+OF_MODE], 1 ; open mode ; 0 = empty entry
0000DACC 722E                   <1>       jb     short fclose_1            ; 1 = read, 2 = write
                                <1>
0000DACE 83F801                 <1>       cmp    eax, 1 ; is the first cluster number > 0
0000DAD1 7229                   <1>       jb     short fclose_1 ; no, this is empty entry
                                <1>
                                <1> fclose_0:
0000DAD3 FE8B[3E630100]         <1>       dec    byte [ebx+OF_OPENCOUNT] ; decrement the number of processes
                                <1>                                      ; that have opened the file
0000DAD9 7921                   <1>       jns    short fclose_1 ; jump if not negative (jump if bit 7 is 0)
                                <1>                     ; if all processes haven't closed the file, return
                                <1>       ;
                                <1>       ; eax ; First cluster
0000DADB 31C0                   <1>       xor    eax, eax ; 0
0000DADD 8883[2A630100]         <1>       mov    [ebx+OF_MODE], al ; 0 = empty entry
                                <1>       ;mov   [ebx+OF_STATUS], al ; 0 = empty entry
0000DAE3 66C1E302               <1>       shl    bx, 2
0000DAE7 8983[F8620100]         <1>       mov    [ebx+OF_FCLUSTER], eax ; 0
0000DAED 8983[10640100]         <1>       mov    [ebx+OF_CCLUSTER], eax ; 0
                                <1>       ;mov   [ebx+OF_CCINDEX], eax ; 0
0000DAF3 A3[74030300]           <1>       mov    [u.fofp], eax ; 0
0000DAF8 66C1EB02               <1>       shr    bx, 2
                                <1> fclose_1: ; 1:
0000DAFC 58                     <1>       pop    eax ; File handle (File Descriptor, File Index)
0000DAFD C680[6A030300]00       <1>       mov    byte [eax+u.fp], 0 ; clear that entry in the u.fp list
0000DB04 C3                     <1>       retn
                                <1>
```

```
                                <1> getf:
                                <1>       ; 12/10/2016
                                <1>       ; 11/10/2016
                                <1>       ; 08/10/2016
                                <1>       ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>       ; / get the device number and the i-number of an open file
                                <1>       ; 13/05/2015
                                <1>       ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
                                <1>       ; 19/04/2013 - 18/11/2013 (Retro UNIX 8086 v1)
                                <1>       ;
0000DB05 89C3                   <1>       mov    ebx, eax
                                <1> getf1:
0000DB07 83FB0A                 <1>       cmp    ebx, 10
0000DB0A 730A                   <1>         jnb  short getf2
0000DB0C 8A9B[6A030300]         <1>       mov    bl, [ebx+u.fp]
0000DB12 08DB                   <1>       or     bl, bl
0000DB14 7503                   <1>       jnz    short getf3
                                <1> getf2:
                                <1>       ; 'File not open !' error (ax=0)
0000DB16 29C0                   <1>       sub    eax, eax
0000DB18 C3                     <1>       retn
                                <1> getf3:
0000DB19 F6C380                 <1>       test   bl, 80h
0000DB1C 7530                   <1>       jnz    short getf5 ; device
0000DB1E FECB                   <1>       dec    bl ; 0 based
0000DB20 8A83[20630100]         <1>       mov    al, [ebx+OF_DRIVE]
0000DB26 A2[46030300]           <1>       mov    [cdev], al
0000DB2B C0E302                 <1>       shl    bl, 2 ; *4 (dword offset)
0000DB2E 8B83[70630100]         <1>       mov    eax, [ebx+OF_SIZE]
0000DB34 A3[55040300]           <1>       mov    [i.size], eax ; file size
0000DB39 8D83[48630100]         <1>       lea    eax, [ebx+OF_POINTER] ;12/10/2016
0000DB3F A3[74030300]           <1>       mov    [u.fofp], eax
0000DB44 8B83[F8620100]         <1>       mov    eax, [ebx+OF_FCLUSTER]
0000DB4A C0EB02                 <1>       shr    bl, 2 ; /4 (byte offset)
                                <1> getf4:
0000DB4D C3                     <1>       retn
                                <1> getf5:
                                <1>       ; get device number
0000DB4E 80E37F                 <1>       and    bl, 7Fh ; 1 to 7Fh
0000DB51 FECB                   <1>       dec    bl ; 0 based (0 to 7Eh)
0000DB53 8A83[52610100]         <1>       mov    al, [ebx+DEV_DRIVER]
0000DB59 8AAB[BC600100]         <1>       mov    ch, [ebx+DEV_ACCESS]
0000DB5F 8A8B[70610100]         <1>       mov    cl, [ebx+DEV_OPENMODE]
0000DB65 80E5FE                 <1>       and    ch, 0FEh ; reset bit 0 ; dev_close
0000DB68 F9                     <1>       stc ; cf = 1
0000DB69 C3                     <1>       retn
                                <1>
                                <1> namei:
                                <1>       ; 04/12/2015 (14 byte file names)
                                <1>       ; 18/10/2015 (nbase, ncount)
                                <1>       ; 12/10/2015
                                <1>       ; 21/08/2015
                                <1>       ; 18/07/2015
                                <1>       ; 02/07/2015
                                <1>       ; 17/06/2015
                                <1>       ; 16/06/2015 (Retro UNIX 386 v1 - Beginning)
                                <1>       ; 24/04/2013 - 31/07/2013 (Retro UNIX 8086 v1)
                                <1>       ;
                                <1>       ; 'namei' takes a file path name and returns i-number of
                                <1>       ; the file in the current directory or the root directory
                                <1>       ; (if the first character of the pathname is '/').
                                <1>       ;
                                <1>       ; INPUTS ->
                                <1>       ;    u.namep - points to a file path name
                                <1>       ;    u.cdir - i-number of users directory
                                <1>       ;    u.cdev - device number on which user directory resides
                                <1>       ; OUTPUTS ->
                                <1>       ;    r1 - i-number of file
                                <1>       ;    cdev
                                <1>       ;    u.dirbuf - points to directory entry where a match
                                <1>       ;               occurs in the search for file path name.
                                <1>       ;               If no match u.dirb points to the end of
                                <1>       ;                the directory and r1 = i-number of the current
                                <1>       ;               directory.
                                <1>       ; ((AX = R1))
                                <1>       ;
                                <1>       ; (Retro UNIX Prototype : 07/10/2012 - 05/01/2013, UNIXCOPY.ASM)
                                <1>         ; ((Modified registers: eDX, eBX, eCX, eSI, eDI, eBP))
                                <1>       ;
                                <1>
0000DB6A 66A1[68030300]         <1>       mov    ax, [u.cdir]
                                <1>              ; mov u.cdir,r1 / put the i-number of current directory
                                <1>                           ; / in r1
0000DB70 668B15[AE030300]       <1>       mov    dx, [u.cdrv]
0000DB77 668915[46030300]       <1>       mov    [cdev], dx       ; NOTE: Retro UNIX 8086 v1
                                <1>                               ; device/drive number is in 1 byte,
                                <1>                               ; not in 1 word!
                                <1>              ; mov u.cdev,cdev / device number for users directory
                                <1>                            ; / into cdev
                                <1>       ; 12/10/2015
                                <1>       ; 16/06/2015 - 32 bit modifications (Retro UNIX 386 v1)
                                <1>              ; convert virtual (pathname) addr to physical address
0000DB7E E82C010000             <1>       call    trans_addr_nmbp ; 12/10/2015
                                <1>              ; esi = physical address of [u.namep]
                                <1>              ; ecx = byte count in the page
0000DB83 803E2F                 <1>       cmp    byte [esi], '/'
                                <1>              ; cmpb *u.namep,$'/ / is first char in file name a /
0000DB86 751E                   <1>       jne    short namei_1
                                <1>              ; bne 1f
0000DB88 FF05[7C030300]         <1>       inc    dword [u.namep]
                                <1>              ; inc u.namep / go to next char
0000DB8E 6649                   <1>       dec    cx ; remain byte count in the page
```

```
4559 0000DB90 7506                <1>       jnz    short namei_0
4560                              <1>       ; 12/10/2015
4561 0000DB92 E818010000          <1>       call   trans_addr_nmbp ; convert virtual address to physical
4562                              <1>              ; esi = physical address (page start + offset)
4563                              <1>              ; ecx = byte count in the page
4564 0000DB97 4E                  <1>       dec    esi
4565                              <1> namei_0:
4566 0000DB98 46                  <1>       inc    esi  ; go to next char
4567 0000DB99 66A1[50030300]      <1>       mov    ax, [rootdir] ; 09/07/2013
4568                              <1>              ; mov rootdir,r1 / put i-number of rootdirectory in r1
4569 0000DB9F C605[46030300]00    <1>       mov    byte [cdev], 0
4570                              <1>              ; clr cdev / clear device number
4571                              <1> namei_1: ; 1:
4572 0000DBA6 F606FF              <1>       test   byte [esi], 0FFh
4573 0000DBA9 74A2                <1>       jz     short getf4
4574                              <1>       ;jz      nig
4575                              <1>              ; tstb *u.namep / is the character in file name a nul
4576                              <1>              ; beq nig / yes, end of file name reached;
4577                              <1>                   ; / branch to "nig"
4578                              <1> namei_2: ; 1:
4579                              <1>       ; 18/10/2015
4580 0000DBAB 8935[44040300]      <1>       mov    [nbase], esi
4581 0000DBB1 66890D[48040300]    <1>       mov    [ncount], cx
4582                              <1>       ;
4583                              <1>       ;mov   dx, 2
4584 0000DBB8 B202                <1>       mov    dl, 2 ; user flag (read, non-owner)
4585 0000DBBA E8E0150000          <1>       call   access
4586                              <1>              ; jsr r0,access; 2 / get i-node with i-number r1
4587                              <1>       ; 'access' will not return here if user has not "r" permission !
4588 0000DBBF 66F705[00000300]00- <1>       test   word [i.flgs], 4000h
4588 0000DBC7 40                  <1>
4589                              <1>              ; bit $40000,i.flgs / directory i-node?
4590 0000DBC8 746A                <1>         jz      short namei_err
4591                              <1>              ; beq error3 / no, got an error
4592                              <1>       ; 16/06/2015 - 32 bit modifications (Retro UNIX 386 v1)
4593 0000DBCA 31C0                <1>       xor    eax, eax
4594 0000DBCC A3[80030300]        <1>       mov    [u.off], eax ; 0
4595 0000DBD1 66A1[55040300]      <1>       mov    ax, [i.size]
4596 0000DBD7 A3[78030300]        <1>       mov    [u.dirp], eax
4597                              <1>              ; mov i.size,u.dirp / put size of directory in u.dirp
4598                              <1>              ; clr u.off / u.off is file offset used by user
4599 0000DBDC C705[74030300]-     <1>       mov    dword [u.fofp], u.off
4599 0000DBE2 [80030300]          <1>
4600                              <1>              ; mov $u.off,u.fofp / u.fofp is a pointer to
4601                              <1>                                  ; / the offset portion of fsp entry
4602                              <1> namei_3: ; 2:
4603 0000DBE6 C705[84030300]-     <1>       mov    dword [u.base], u.dirbuf
4603 0000DBEC [98030300]          <1>
4604                              <1>              ; mov $u.dirbuf,u.base / u.dirbuf holds a file name
4605                              <1>                                    ; / copied from a directory
4606 0000DBF0 C705[88030300]1000- <1>       mov    dword [u.count], 16 ; 04/12/2015 (10 -> 16)
4606 0000DBF8 0000                <1>
4607                              <1>              ; mov $10.,u.count / u.count is byte count
4608                              <1>                                 ; / for reads and writes
4609 0000DBFA 66A1[51040300]      <1>       mov    ax, [ii]
4610                              <1>       ; 31/07/2013 ('namei_r') - 16/06/2015 ('u.kcall')
4611 0000DC00 FE05[C6030300]      <1>       inc     byte [u.kcall] ; the caller is 'namei' sign
4612 0000DC06 E8A2070000          <1>       call   readi
4613                              <1>              ; jsr r0,readi / read 10. bytes of file
4614                              <1>                   ; with i-number (r1); i.e. read a directory entry
4615 0000DC0B 8B0D[8C030300]      <1>       mov    ecx, [u.nread]
4616 0000DC11 09C9                <1>       or     ecx, ecx
4617                              <1>              ; tst u.nread
4618 0000DC13 741B                <1>       jz     short nib
4619                              <1>              ; ble nib / gives error return
4620                              <1>       ;
4621 0000DC15 668B1D[98030300]    <1>       mov    bx, [u.dirbuf]
4622 0000DC1C 6621DB              <1>       and    bx, bx
4623                              <1>              ; tst u.dirbuf /
4624 0000DC1F 7522                <1>       jnz    short namei_4
4625                              <1>              ; bne 3f / branch when active directory entry
4626                              <1>                       ; / (i-node word in entry non zero)
4627 0000DC21 A1[80030300]        <1>       mov    eax, [u.off]
4628 0000DC26 83E810              <1>       sub    eax, 16 ; 04/12/2015 (10 -> 16)
4629 0000DC29 A3[78030300]        <1>       mov    [u.dirp], eax
4630                              <1>              ; mov u.off,u.dirp
4631                              <1>              ; sub $10.,u.dirp
4632 0000DC2E EBB6                <1>       jmp    short namei_3
4633                              <1>              ; br 2b
4634                              <1>
4635                              <1>       ; 18/07/2013
4636                              <1> nib:
4637 0000DC30 31C0                <1>       xor    eax, eax  ; xor ax, ax ; ax = 0 -> file not found
4638 0000DC32 F9                  <1>       stc
4639                              <1> nig:
4640 0000DC33 C3                  <1>       retn
4641                              <1>
4642                              <1> namei_err:
4643                              <1>       ; 16/06/2015
4644 0000DC34 C705[C8030300]1300- <1>       mov    dword [u.error], ERR_NOT_DIR ; 'not a directory !' error
4644 0000DC3C 0000                <1>
4645 0000DC3E E950E8FFFF          <1>       jmp    error
4646                              <1>
4647                              <1> namei_4: ; 3:
4648                              <1>       ; 18/10/2015
4649                              <1>       ; 12/10/2015
4650                              <1>       ; 21/08/2015
4651                              <1>       ; 18/07/2015
4652 0000DC43 8B2D[7C030300]      <1>       mov    ebp, [u.namep]
4653                              <1>              ; mov u.namep,r2 / u.namep points into a file name string
4654 0000DC49 BF[9A030300]        <1>       mov    edi, u.dirbuf + 2
4655                              <1>              ; mov $u.dirbuf+2,r3 / points to file name of directory entry
```

```
                                <1>       ; 18/10/2015
0000DC4E 8B35[44040300]         <1>       mov    esi, [nbase]
0000DC54 668B0D[48040300]       <1>       mov    cx, [ncount]
                                <1>       ;
0000DC5B 6621C9                 <1>       and    cx, cx
0000DC5E 7505                   <1>       jnz    short namei_5
                                <1>       ;
0000DC60 E850000000             <1>       call   trans_addr_nm ; convert virtual address to physical
                                <1>              ; esi = physical address (page start + offset)
                                <1>              ; ecx = byte count in the page
                                <1> namei_5: ; 3:
0000DC65 45                     <1>       inc    ebp ; 18/07/2015
0000DC66 AC                     <1>       lodsb   ; mov al, [esi] ; inc esi (al = r4)
                                <1>              ; movb (r2)+,r4 / move a character from u.namep string into r4
0000DC67 08C0                   <1>       or     al, al
0000DC69 741D                   <1>       jz     short namei_7
                                <1>              ; beq 3f / if char is nul, then the last char in string
                                <1>                    ; / has been moved
0000DC6B 3C2F                   <1>       cmp    al, '/'
                                <1>              ; cmp r4,$'/ / is char a </>
0000DC6D 7419                   <1>       je     short namei_7
                                <1>              ; beq 3f
                                <1>       ; 12/10/2015
0000DC6F 6649                   <1>       dec    cx ; remain byte count in the page
0000DC71 7505                   <1>       jnz    short namei_6
0000DC73 E83D000000             <1>       call   trans_addr_nm ; convert virtual address to physical
                                <1>              ; esi = physical address (page start + offset)
                                <1>              ; ecx = byte count in the page
                                <1> namei_6:
0000DC78 81FF[A8030300]         <1>         cmp     edi, u.dirbuf + 16 ; 04/12/2015 (10 -> 16)
                                <1>              ; cmp r3,$u.dirbuf+10. / have I checked
                                <1>                                   ; / all 8 bytes of file name
0000DC7E 74E5                   <1>       je     short namei_5
                                <1>              ; beq 3b
0000DC80 AE                     <1>       scasb
                                <1>              ; cmpb (r3)+,r4 / compare char in u.namep string to file name
                                <1>                           ; / char read from directory
0000DC81 74E2                   <1>       je     short namei_5
                                <1>              ; beq 3b / branch if chars match
                                <1>
0000DC83 E95EFFFFFF             <1>         jmp    namei_3 ; 2b
                                <1>              ; br 2b / file names do not match go to next directory entry
                                <1> namei_7: ; 3:
0000DC88 81FF[A8030300]         <1>       cmp    edi, u.dirbuf + 16 ; 04/12/2015 (10 -> 16)
                                <1>              ; cmp r3,$u.dirbuf+10. / if equal all 8 bytes were matched
0000DC8E 740A                   <1>       je     short namei_8
                                <1>              ; beq 3f
0000DC90 8A27                   <1>       mov    ah, [edi]
                                <1>       ;inc   edi
0000DC92 20E4                   <1>       and    ah, ah
                                <1>              ; tstb (r3)+ /
0000DC94 0F854CFFFFFF           <1>         jnz     namei_3
                                <1>              ; bne 2b
                                <1> namei_8: ; 3
0000DC9A 892D[7C030300]         <1>       mov    [u.namep], ebp ; 18/07/2015
                                <1>              ; mov r2,u.namep / u.namep points to char
                                <1>                                ; / following a / or nul
                                <1>       ;mov   bx, [u.dirbuf]
                                <1>              ; mov u.dirbuf,r1 / move i-node number in directory
                                <1>                              ; / entry to r1
0000DCA0 20C0                   <1>       and    al, al
                                <1>              ; tst r4 / if r4 = 0 the end of file name reached,
                                <1>                    ;  / if r4 = </> then go to next directory
                                <1>       ; mov  ax, bx
0000DCA2 66A1[98030300]         <1>       mov    ax, [u.dirbuf] ; 17/06/2015
0000DCA8 0F85FDFEFFFF           <1>         jnz     namei_2
                                <1>              ; bne 1b
                                <1>       ; AX = i-number of the file
                                <1> ;;nig:
0000DCAE C3                     <1>       retn
                                <1>              ; tst (r0)+ / gives non-error return
                                <1> ;;nib:
                                <1> ;;    xor    ax, ax ; Retro UNIX 8086 v1 modification !
                                <1>                     ; ax = 0 -> file not found
                                <1> ;;    stc    ; 27/05/2013
                                <1> ;;    retn
                                <1>              ; rts r0
                                <1>
                                <1> trans_addr_nmbp:
                                <1>       ; 18/10/2015
                                <1>       ; 12/10/2015
0000DCAF 8B2D[7C030300]         <1>       mov    ebp, [u.namep]
                                <1> trans_addr_nm:
                                <1>       ; Convert virtual (pathname) address to physical address
                                <1>       ; (Retro UNIX 386 v1 feature only !)
                                <1>       ; 18/10/2015
                                <1>       ; 12/10/2015 (u.pnbase & u.pncount has been removed from code)
                                <1>       ; 02/07/2015
                                <1>       ; 17/06/2015
                                <1>       ; 16/06/2015
                                <1>       ;
                                <1>       ; INPUTS:
                                <1>       ;     ebp = pathname address (virtual) ; [u.namep]
                                <1>       ;     [u.pgdir] = user's page directory
                                <1>       ; OUTPUT:
                                <1>       ;       esi = physical address of the pathname
                                <1>       ;     ecx = remain byte count in the page
                                <1>       ;
                                <1>       ; (Modified registers: EAX, EBX, ECX, EDX, ESI)
                                <1>       ;
0000DCB5 833D[BC030300]00       <1>         cmp     dword [u.ppgdir], 0  ; /etc/init ? (sysexec)
0000DCBC 7618                   <1>       jna    short trans_addr_nmk ; the caller is os kernel;
```

```
                                <1>                                     ; it is already physical address
0000DCBE 50                     <1>         push   eax
0000DCBF 89EB                   <1>         mov    ebx, ebp ; [u.namep] ; pathname address (virtual)
0000DCC1 E8C875FFFF             <1>                call   get_physical_addr ; get physical address
0000DCC6 7204                   <1>         jc     short tr_addr_nm_err
                                <1>         ; 18/10/2015
                                <1>         ; eax = physical address
                                <1>         ; cx = remain byte count in page (1-4096)
                                <1>                ; 12/10/2015 (cx = [u.pncount])
0000DCC8 89C6                   <1>         mov    esi, eax ; 12/10/2015 (esi=[u.pnbase])
0000DCCA 58                     <1>         pop    eax
0000DCCB C3                     <1>         retn
                                <1>
                                <1> tr_addr_nm_err:
0000DCCC A3[C8030300]           <1>         mov    [u.error], eax
                                <1>         ;pop   eax
0000DCD1 E9BDE7FFFF             <1>         jmp    error
                                <1>
                                <1> trans_addr_nmk:
                                <1>         ; 12/10/2015
                                <1>         ; 02/07/2015
0000DCD6 8B35[7C030300]         <1>         mov    esi, [u.namep]  ; [u.pnbase]
0000DCDC 66B90010               <1>         mov    cx, PAGE_SIZE ; 4096 ; [u.pncount]
0000DCE0 C3                     <1>         retn
                                <1>
                                <1> syschdir:
                                <1>         ; / makes the directory specified in the argument
                                <1>         ; / the current directory
                                <1>         ;
                                <1>         ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
                                <1>         ; 19/06/2013 (Retro UNIX 8086 v1)
                                <1>         ;
                                <1>         ; 'syschdir' makes the directory specified in its argument
                                <1>         ; the current working directory.
                                <1>         ;
                                <1>         ; Calling sequence:
                                <1>         ;      syschdir; name
                                <1>         ; Arguments:
                                <1>         ;      name - address of the path name of a directory
                                <1>         ;             terminated by nul byte.
                                <1>         ; Inputs: -
                                <1>         ; Outputs: -
                                <1>         ; ..........................................................
                                <1>         ;
                                <1>         ; Retro UNIX 8086 v1 modification:
                                <1>         ;       The user/application program puts address of
                                <1>         ;       the path name in BX register as 'syschdir'
                                <1>         ;       system call argument.
                                <1>
0000DCE1 891D[7C030300]         <1>         mov    [u.namep], ebx
                                <1>                ;jsr r0,arg; u.namep / u.namep points to path name
0000DCE7 E87EFEFFFF             <1>         call   namei
                                <1>                ; jsr r0,namei / find its i-number
                                <1>         ;jc    error
                                <1>                ; br error3
0000DCEC 730F                   <1>         jnc    short syschdir0
                                <1>         ; 'directory not found !' error
0000DCEE C705[C8030300]0C00-    <1>         mov    dword [u.error], ERR_DIR_NOT_FOUND ; 12
0000DCF6 0000                   <1>
0000DCF8 E996E7FFFF             <1>         jmp    error
                                <1> syschdir0:
0000DCFD E89D140000             <1>         call   access
                                <1>                ; jsr r0,access; 2 / get i-node into core
0000DD02 66F705[00000300]00-    <1>         test   word [i.flgs], 4000h
0000DD0A 40                     <1>
                                <1>                ; bit $40000,i.flgs / is it a directory?
                                <1>         ;jz    error
                                <1>                ; beq error3 / no error
0000DD0B 750F                   <1>         jnz    short syschdir1
0000DD0D C705[C8030300]1300-    <1>         mov    dword [u.error], ERR_NOT_DIR ; 'not a valid directory !'
0000DD15 0000                   <1>
0000DD17 E977E7FFFF             <1>         jmp    error
                                <1> syschdir1:
0000DD1C 66A3[68030300]         <1>         mov    [u.cdir], ax
                                <1>                ; mov r1,u.cdir / move i-number to users
                                <1>                                ; / current directory
0000DD22 66A1[46030300]         <1>         mov    ax, [cdev]
0000DD28 66A3[AE030300]         <1>         mov    [u.cdrv], ax
                                <1>                ; mov cdev,u.cdev / move its device to users
                                <1>                                  ; / current device
0000DD2E E980E7FFFF             <1>         jmp    sysret
                                <1>                ; br sysret3
                                <1>
                                <1> syschmod: ; < change mode of file >
                                <1>         ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
                                <1>         ; temporary !
0000DD33 B801000000             <1>         mov    eax, ERR_INV_FNUMBER ; 'invalid function number !'
0000DD38 A3[C8030300]           <1>           mov     [u.error], eax
0000DD3D A3[64030300]           <1>           mov     [u.r0], eax
0000DD42 E94CE7FFFF             <1>         jmp    error
                                <1>
                                <1> isown:
                                <1>         ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                                <1>         ; 04/05/2013 - 07/07/2013 (Retro UNIX 8086 v1)
                                <1>         ;
                                <1>         ; 'isown' is given a file name (the 1st argument).
                                <1>         ;  It find the i-number of that file via 'namei'
                                <1>         ;  then gets the i-node into core via 'iget'.
                                <1>         ;  It then tests to see if the user is super user.
                                <1>         ;  If not, it cheks to see if the user is owner of
                                <1>         ;  the file. If he is not an error occurs.
                                <1>         ;  If user is the owner 'setimod' is called to indicate
```

```
4857                                    <1>       ;  the inode has been modificed and the 2nd argument of
4858                                    <1>       ;  the call is put in r2.
4859                                    <1>       ;
4860                                    <1>       ; INPUTS ->
4861                                    <1>       ;    arguments of syschmod and syschown calls
4862                                    <1>       ; OUTPUTS ->
4863                                    <1>       ;    u.uid - id of user
4864                                    <1>       ;    imod - set to a 1
4865                                    <1>       ;    r2 - contains second argument of the system call
4866                                    <1>       ;
4867                                    <1>       ;   ((AX=R2) output as 2nd argument)
4868                                    <1>       ;
4869                                    <1>         ; ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
4870                                    <1>       ;
4871                                    <1>             ; jsr r0,arg2 / u.namep points to file name
4872                                    <1>       ;; ! 2nd argument on top of stack !
4873                                    <1>       ;; 22/06/2015 - 32 bit modifications
4874                                    <1>       ;; 07/07/2013
4875 0000DD47 891D[7C030300]            <1>       mov   [u.namep], ebx ;; 1st argument
4876 0000DD4D 51                       <1>       push  ecx ;; 2nd argument
4877                                    <1>       ;;
4878 0000DD4E E817FEFFFF               <1>       call  namei
4879                                    <1>             ; jsr r0,namei / get its i-number
4880                                    <1>        ; Retro UNIX 8086 v1 modification !
4881                                    <1>        ; ax = 0 -> file not found
4882                                    <1>       ;and   ax, ax
4883                                    <1>       ;jz    error
4884                                    <1>       ;jc    error ; 27/05/2013
4885                                    <1>             ; br error3
4886 0000DD53 730F                     <1>       jnc   short isown0
4887                                    <1>       ; 'file not found !' error
4888 0000DD55 C705[C8030300]0C00-       <1>       mov   dword [u.error], ERR_FILE_NOT_FOUND ; 12
4888 0000DD5D 0000                     <1>
4889 0000DD5F E92FE7FFFF               <1>             jmp   error
4890                                    <1> isown0:
4891 0000DD64 E830140000               <1>             call  iget
4892                                    <1>             ; jsr r0,iget / get i-node into core
4893 0000DD69 A0[B0030300]             <1>             mov   al, [u.uid] ; 02/08/2013
4894 0000DD6E 08C0                     <1>             or    al, al
4895                                    <1>             ; tstb u.uid / super user?
4896 0000DD70 7417                     <1>             jz    short isown1
4897                                    <1>             ; beq 1f / yes, branch
4898 0000DD72 3A05[03000300]           <1>             cmp   al, [i.uid]
4899                                    <1>             ; cmpb i.uid,u.uid / no, is this the owner of
4900                                    <1>                                ; / the file
4901                                    <1>             ;jne  error
4902                                    <1>             ; beq 1f / yes
4903                                    <1>             ; jmp error3 / no, error
4904 0000DD78 740F                     <1>             je    short isown1
4905                                    <1>
4906 0000DD7A C705[C8030300]0B00-       <1>             mov   dword [u.error], ERR_NOT_OWNER  ; 11
4906 0000DD82 0000                     <1>
4907                                    <1>                           ;  'permission denied !' error
4908 0000DD84 E90AE7FFFF               <1>             jmp   error
4909                                    <1> isown1: ; 1:
4910 0000DD89 E80F140000               <1>             call  setimod
4911                                    <1>             ; jsr r0,setimod / indicates
4912                                    <1>             ;            ; / i-node has been modified
4913 0000DD8E 58                       <1>             pop   eax ; 2nd argument
4914                                    <1>             ; mov (sp)+,r2 / mode is put in r2
4915                                    <1>                    ; / (u.off put on stack with 2nd arg)
4916 0000DD8F C3                       <1>             retn
4917                                    <1>             ; rts r0
4918                                    <1>
4919                                    <1> ;;arg:  ; < get system call arguments >
4920                                    <1>       ; 'arg' extracts an argument for a routine whose call is
4921                                    <1>       ; of form:
4922                                    <1>       ;     sys 'routine' ; arg1
4923                                    <1>       ;           or
4924                                    <1>       ;     sys 'routine' ; arg1 ; arg2
4925                                    <1>       ;           or
4926                                    <1>       ;     sys 'routine' ; arg1;...;arg10 (sys exec)
4927                                    <1>       ;
4928                                    <1>       ; INPUTS ->
4929                                    <1>       ;    u.sp+18 - contains a pointer to one of arg1..argn
4930                                    <1>       ;     This pointers's value is actually the value of
4931                                    <1>       ;     update pc at the the trap to sysent (unkni) is
4932                                    <1>       ;     made to process the sys instruction
4933                                    <1>       ;    r0 - contains the return address for the routine
4934                                    <1>       ;     that called arg. The data in the word pointer
4935                                    <1>       ;     to by the return address is used as address
4936                                    <1>       ;     in which the extracted argument is stored
4937                                    <1>       ;
4938                                    <1>       ; OUTPUTS ->
4939                                    <1>       ;    'address' - contains the extracted argument
4940                                    <1>       ;    u.sp+18 - is incremented by 2
4941                                    <1>       ;    r1 - contains the extracted argument
4942                                    <1>       ;    r0 - points to the next instruction to be
4943                                    <1>       ;      executed in the calling routine.
4944                                    <1>       ;
4945                                    <1>
4946                                    <1>       ; mov u.sp,r1
4947                                    <1>       ; mov *18.(r1),*(r0)+ / put argument of system call
4948                                    <1>                    ; / into argument of arg2
4949                                    <1>       ; add $2,18.(r1) / point pc on stack
4950                                    <1>                          ; / to next system argument
4951                                    <1>       ; rts r0
4952                                    <1>
4953                                    <1> ;;arg2: ; < get system calls arguments - with file name pointer>
4954                                    <1>       ; 'arg2' takes first argument in system call
4955                                    <1>       ;  (pointer to name of the file) and puts it in location
4956                                    <1>       ;  u.namep; takes second argument and puts it in u.off
```

```
                                <1>       ;  and on top of the stack
                                <1>       ;
                                <1>       ; INPUTS ->
                                <1>       ;    u.sp, r0
                                <1>       ;
                                <1>       ; OUTPUTS ->
                                <1>       ;    u.namep
                                <1>       ;    u.off
                                <1>       ;    u.off pushed on stack
                                <1>       ;    r1
                                <1>       ;
                                <1>
                                <1>       ; jsr  r0,arg; u.namep / u.namep contains value of
                                <1>                      ; / first arg in sys call
                                <1>       ; jsr r0,arg; u.off / u.off contains value of
                                <1>                      ; / second arg in sys call
                                <1>       ; mov r0,r1 / r0 points to calling routine
                                <1>       ; mov (sp),r0 / put operation code back in r0
                                <1>       ; mov u.off,(sp) / put pointer to second argument
                                <1>                  ; / on stack
                                <1>       ; jmp (r1) / return to calling routine
                                <1>
                                <1> syschown: ; < change owner of file >
                                <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
                                <1>       ; 20/06/2013 - 02/08/2013 (Retro UNIX 8086 v1)
                                <1>       ;
                                <1>       ; 'syschown' changes the owner of the file whose name is given
                                <1>       ; as null terminated string pointed to by 'name' has it's owner
                                <1>       ; changed to 'owner'
                                <1>       ;
                                <1>       ; Calling sequence:
                                <1>       ;     syschown; name; owner
                                <1>       ; Arguments:
                                <1>       ;     name - address of the file name
                                <1>       ;            terminated by null byte.
                                <1>       ;     owner - (new) owner (number/ID)
                                <1>       ;
                                <1>       ; Inputs: -
                                <1>       ; Outputs: -
                                <1>       ; ..........................................................
                                <1>       ;
                                <1>       ; Retro UNIX 8086 v1 modification:
                                <1>       ;       'syschown' system call has two arguments; so,
                                <1>       ;     * 1st argument, name is pointed to by BX register
                                <1>       ;     * 2nd argument, owner number is in CX register
                                <1>       ;
                                <1>       ; / name; owner
0000DD90 E8B2FFFFFF             <1>       call   isown
                                <1>              ; jsr r0,isown / get the i-node and check user status
0000DD95 803D[B0030300]00       <1>       cmp    byte [u.uid], 0 ; 02/08/2013
                                <1>              ; tstb u.uid / super user
0000DD9C 7418                   <1>       jz     short syschown1
                                <1>              ; beq 2f / yes, 2f
0000DD9E F605[00000300]20       <1>         test    byte [i.flgs], 20h ; 32
                                <1>              ; bit $40,i.flgs / no, set userid on execution?
                                <1>       ;jnz   error
                                <1>              ; bne 3f / yes error, could create Trojan Horses
0000DDA5 740F                   <1>       jz     short syschown1
                                <1>       ; 'permission denied !'
0000DDA7 C705[C8030300]0B00-    <1>       mov    dword [u.error], ERR_FILE_ACCESS  ; 11
0000DDAF 0000                   <1>
0000DDB1 E9DDE6FFFF             <1>       jmp    error
                                <1> syschown1: ; 2:
                                <1>       ; AL = owner (number/ID)
0000DDB6 A2[03000300]           <1>       mov    [i.uid], al ; 23/06/2015
                                <1>              ;  movb      r2,i.uid / no, put the new owners id
                                <1>                           ; / in the i-node
0000DDBB E9F3E6FFFF             <1>       jmp    sysret
                                <1>       ; 1:
                                <1>              ; jmp sysret4
                                <1>       ; 3:
                                <1>              ; jmp  error
                                <1>
                                <1> systime: ; / get time of year
                                <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
                                <1>       ; 20/06/2013 (Retro UNIX 8086 v1)
                                <1>       ;
                                <1>       ; 20/06/2013
                                <1>       ; 'systime' gets the time of the year.
                                <1>       ; The present time is put on the stack.
                                <1>       ;
                                <1>       ; Calling sequence:
                                <1>       ;     systime
                                <1>       ; Arguments: -
                                <1>       ;
                                <1>       ; Inputs: -
                                <1>       ; Outputs: sp+2, sp+4 - present time
                                <1>       ; ..........................................................
                                <1>       ;
                                <1>       ; Retro UNIX 8086 v1 modification:
                                <1>       ;       'systime' system call will return to the user
                                <1>       ;     with unix time (epoch) in DX:AX register pair
                                <1>       ;
                                <1>       ;     !! Major modification on original Unix v1 'systime'
                                <1>       ;     system call for PC compatibility !!
                                <1>
0000DDC0 E8DB130000             <1>       call   epoch
0000DDC5 A3[64030300]           <1>       mov    [u.r0], eax
                                <1>              ; mov s.time,4(sp)
                                <1>              ; mov s.time+2,2(sp) / put the present time
                                <1>                              ; / on the stack
                                <1>              ; br sysret4
```

```
5058 0000DDCA E9E4E6FFFF          <1>       jmp    sysret
5059                              <1>
5060                              <1> sysstime: ; / set time
5061                              <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
5062                              <1>       ; 20/06/2013 - 02/08/2013 (Retro UNIX 8086 v1)
5063                              <1>       ;
5064                              <1>       ; 'sysstime' sets the time. Only super user can use this call.
5065                              <1>       ;
5066                              <1>       ; Calling sequence:
5067                              <1>       ;     sysstime
5068                              <1>       ; Arguments: -
5069                              <1>       ;
5070                              <1>       ; Inputs: sp+2, sp+4 - time system is to be set to.
5071                              <1>       ; Outputs: -
5072                              <1>       ; ............................................................
5073                              <1>       ;
5074                              <1>       ; Retro UNIX 8086 v1 modification:
5075                              <1>       ;     the user calls 'sysstime' with unix (epoch) time
5076                              <1>       ;     (to be set) is in CX:BX register pair as two arguments.
5077                              <1>       ;
5078                              <1>       ;     Retro UNIX 8086 v1 argument transfer method 2 is used
5079                              <1>       ;     to get sysstime system call arguments from the user;
5080                              <1>       ;     * 1st argument, lowword of unix time is in BX register
5081                              <1>       ;     * 2nd argument, highword of unix time is in CX register
5082                              <1>       ;
5083                              <1>       ;     !! Major modification on original Unix v1 'sysstime'
5084                              <1>       ;     system call for PC compatibility !!
5085                              <1>
5086 0000DDCF 803D[B0030300]00    <1>       cmp    byte [u.uid], 0
5087                              <1>              ; tstb u.uid / is user the super user
5088                              <1>       ;ja    error
5089                              <1>              ; bne error4 / no, error
5090 0000DDD6 760F                <1>       jna    short systime1
5091                              <1>       ; 'permission denied !'
5092 0000DDD8 C705[C8030300]0B00- <1>       mov    dword [u.error], ERR_NOT_SUPERUSER  ; 11
5092 0000DDE0 0000                <1>
5093 0000DDE2 E9ACE6FFFF          <1>       jmp    error
5094                              <1> systime1:
5095                              <1>       ; 23/06/2015 (Retro UNIX 386 v1 - 32 bit version)
5096                              <1>       ; EBX = unix (epoch) time (from user)
5097 0000DDE7 89D8                <1>       mov    eax, ebx
5098 0000DDE9 E8B4130000          <1>       call   set_date_time
5099                              <1>              ; mov 4(sp),s.time
5100                              <1>              ; mov 2(sp),s.time+2 / set the system time
5101 0000DDEE E9C0E6FFFF          <1>       jmp    sysret
5102                              <1>              ; br sysret4
5103                              <1>
5104                              <1> sysbreak:
5105                              <1>       ; 18/10/2015
5106                              <1>       ; 07/10/2015
5107                              <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
5108                              <1>       ; 20/06/2013 - 24/03/2014 (Retro UNIX 8086 v1)
5109                              <1>       ;
5110                              <1>       ; 'sysbreak' sets the programs break points.
5111                              <1>       ; It checks the current break point (u.break) to see if it is
5112                              <1>       ; between "core" and the stack (sp). If it is, it is made an
5113                              <1>       ; even address (if it was odd) and the area between u.break
5114                              <1>       ; and the stack is cleared. The new breakpoint is then put
5115                              <1>       ; in u.break and control is passed to 'sysret'.
5116                              <1>       ;
5117                              <1>       ; Calling sequence:
5118                              <1>       ;     sysbreak; addr
5119                              <1>       ; Arguments: -
5120                              <1>       ;
5121                              <1>       ; Inputs: u.break - current breakpoint
5122                              <1>       ; Outputs: u.break - new breakpoint
5123                              <1>       ;     area between old u.break and the stack (sp) is cleared.
5124                              <1>       ; ............................................................
5125                              <1>       ;
5126                              <1>       ; Retro UNIX 8086 v1 modification:
5127                              <1>       ;     The user/application program puts breakpoint address
5128                              <1>       ;      in BX register as 'sysbreak' system call argument.
5129                              <1>       ;     (argument transfer method 1)
5130                              <1>       ;
5131                              <1>       ;  NOTE: Beginning of core is 0 in Retro UNIX 8086 v1 !
5132                              <1>       ;     ((!'sysbreak' is not needed in Retro UNIX 8086 v1!))
5133                              <1>       ;  NOTE:
5134                              <1>       ;     'sysbreak' clears extended part (beyond of previous
5135                              <1>       ;     'u.break' address) of user's memory for original unix's
5136                              <1>       ;     'bss' compatibility with Retro UNIX 8086 v1 (19/11/2013)
5137                              <1>
5138                              <1>              ; mov u.break,r1 / move users break point to r1
5139                              <1>              ; cmp r1,$core / is it the same or lower than core?
5140                              <1>              ; blos 1f / yes, 1f
5141                              <1>       ; 23/06/2015
5142 0000DDF3 8B2D[90030300]      <1>       mov    ebp, [u.break] ; virtual address (offset)
5143                              <1>       ;and   ebp, ebp
5144                              <1>       ;jz    short sysbreak_3
5145                              <1>       ; Retro UNIX 386 v1 NOTE: u.break points to virtual address !!!
5146                              <1>       ; (Even break point address is not needed for Retro UNIX 386 v1)
5147 0000DDF9 8B15[5C030300]      <1>       mov    edx, [u.sp] ; kernel stack at the beginning of sys call
5148 0000DDFF 83C20C              <1>       add    edx, 12 ; EIP -4-> CS -4-> EFLAGS -4-> ESP (user)
5149                              <1>       ; 07/10/2015
5150 0000DE02 891D[90030300]      <1>       mov    [u.break], ebx ; virtual address !!!
5151                              <1>       ;
5152 0000DE08 3B1A                <1>       cmp    ebx, [edx] ; compare new break point with
5153                              <1>                         ; with top of user's stack (virtual!)
5154 0000DE0A 7327                <1>       jnb    short sysbreak_3
5155                              <1>              ; cmp r1,sp / is it the same or higher
5156                              <1>                      ; / than the stack?
5157                              <1>              ; bhis 1f / yes, 1f
5158 0000DE0C 89DE                <1>       mov    esi, ebx
```

```
5159 0000DE0E 29EE                <1>       sub    esi, ebp ; new break point - old break point
5160 0000DE10 7621                <1>       jna    short sysbreak_3
5161                              <1>       ;push  ebx
5162                              <1> sysbreak_1:
5163 0000DE12 89EB                <1>       mov    ebx, ebp
5164 0000DE14 E87574FFFF          <1>       call   get_physical_addr ; get physical address
5165 0000DE19 0F82ADFEFFFF        <1>       jc     tr_addr_nm_err
5166                              <1>       ; 18/10/2015
5167 0000DE1F 89C7                <1>       mov    edi, eax
5168 0000DE21 29C0                <1>       sub    eax, eax ; 0
5169                              <1>              ; ECX = remain byte count in page (1-4096)
5170 0000DE23 39CE                <1>       cmp    esi, ecx
5171 0000DE25 7302                <1>       jnb    short sysbreak_2
5172 0000DE27 89F1                <1>       mov    ecx, esi
5173                              <1> sysbreak_2:
5174 0000DE29 29CE                <1>       sub    esi, ecx
5175 0000DE2B 01CD                <1>       add    ebp, ecx
5176 0000DE2D F3AA                <1>       rep    stosb
5177 0000DE2F 09F6                <1>       or     esi, esi
5178 0000DE31 75DF                <1>       jnz    short sysbreak_1
5179                              <1>       ;
5180                              <1>              ; bit $1,r1 / is it an odd address
5181                              <1>              ; beq 2f / no, its even
5182                              <1>              ; clrb (r1)+ / yes, make it even
5183                              <1>       ; 2: / clear area between the break point and the stack
5184                              <1>              ; cmp r1,sp / is it higher or same than the stack
5185                              <1>              ; bhis 1f / yes, quit
5186                              <1>              ; clr (r1)+ / clear word
5187                              <1>              ; br 2b / go back
5188                              <1>       ;pop   ebx
5189                              <1> sysbreak_3: ; 1:
5190                              <1>       ;mov   [u.break], ebx ; virtual address !!!
5191                              <1>              ; jsr r0,arg; u.break / put the "address"
5192                              <1>                    ; / in u.break (set new break point)
5193                              <1>              ; br sysret4 / br sysret
5194 0000DE33 E97BE6FFFF          <1>       jmp    sysret
5195                              <1>
5196                              <1> sysseek: ; / moves read write pointer in an fsp entry
5197                              <1>       ; 06/11/2016 - TRDOS 386 (TRDOS v2.0)
5198                              <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
5199                              <1>       ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
5200                              <1>       ;
5201                              <1>       ; 'sysseek' changes the r/w pointer of (3rd word of in an
5202                              <1>       ; fsp entry) of an open file whose file descriptor is in u.r0.
5203                              <1>       ; The file descriptor refers to a file open for reading or
5204                              <1>       ; writing. The read (or write) pointer is set as follows:
5205                              <1>       ;     * if 'ptrname' is 0, the pointer is set to offset.
5206                              <1>       ;     * if 'ptrname' is 1, the pointer is set to its
5207                              <1>       ;       current location plus offset.
5208                              <1>       ;     * if 'ptrname' is 2, the pointer is set to the
5209                              <1>       ;       size of file plus offset.
5210                              <1>       ; The error bit (e-bit) is set for an undefined descriptor.
5211                              <1>       ;
5212                              <1>       ; Calling sequence:
5213                              <1>       ;     sysseek; offset; ptrname
5214                              <1>       ; Arguments:
5215                              <1>       ;     offset - number of bytes desired to move
5216                              <1>       ;              the r/w pointer
5217                              <1>       ;     ptrname - a switch indicated above
5218                              <1>       ;
5219                              <1>       ; Inputs: r0 - file descriptor
5220                              <1>       ; Outputs: -
5221                              <1>       ; ..........................................................
5222                              <1>       ;
5223                              <1>       ; Retro UNIX 8086 v1 modification:
5224                              <1>       ;       'sysseek' system call has three arguments; so,
5225                              <1>       ;     * 1st argument, file descriptor is in BX (BL) register
5226                              <1>       ;     * 2nd argument, offset is in CX register
5227                              <1>       ;     * 3rd argument, ptrname/switch is in DX (DL) register
5228                              <1>
5229 0000DE38 E821000000          <1>       call   seektell
5230                              <1>       ; EAX = Current R/W pointer of the file
5231                              <1>       ; EBX = [u.fofp]
5232                              <1>       ; [u.base] = offset (ECX input)
5233                              <1>
5234 0000DE3D 0305[84030300]      <1>       add    eax, [u.base]
5235 0000DE43 8903                <1>       mov    [ebx], eax
5236 0000DE45 E969E6FFFF          <1>       jmp    sysret
5237                              <1>
5238                              <1> systell: ; / get the r/w pointer
5239                              <1>       ; 06/11/2016 - TRDOS 386 (TRDOS v2.0) - temporary !-
5240                              <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
5241                              <1>       ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
5242                              <1>       ;
5243                              <1>       ; Retro UNIX 8086 v1 modification:
5244                              <1>       ; ! 'systell' does not work in original UNIX v1,
5245                              <1>       ;          it returns with error !
5246                              <1>       ; Inputs: r0 - file descriptor
5247                              <1>       ; Outputs: r0 - file r/w pointer
5248                              <1>
5249                              <1>       ;xor   ecx, ecx ; 0
5250 0000DE4A BA01000000          <1>       mov    edx, 1 ; 05/08/2013
5251                              <1>       ;call  seektell
5252 0000DE4F E810000000          <1>       call   seektell0 ; 05/08/2013
5253                              <1>       ;; 06/11/2016
5254                              <1>       ;; mov eax, [ebx]
5255 0000DE54 A3[64030300]        <1>       mov    [u.r0], eax
5256 0000DE59 E955E6FFFF          <1>       jmp    sysret
5257                              <1>
5258                              <1> ; Original unix v1 'systell' system call:
5259                              <1>              ; jsr r0,seektell
5260                              <1>              ; br error4
```

```
                                <1>
                                <1> seektell:
                                <1>       ; 06/11/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; 03/01/2016
                                <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                                <1>       ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
                                <1>       ;
                                <1>       ; 'seektell' puts the arguments from sysseek and systell
                                <1>       ; call in u.base and u.count. It then gets the i-number of
                                <1>       ; the file from the file descriptor in u.r0 and by calling
                                <1>       ; getf. The i-node is brought into core and then u.count
                                <1>       ; is checked to see it is a 0, 1, or 2.
                                <1>       ; If it is 0 - u.count stays the same
                                <1>       ;         1 - u.count = offset (u.fofp)
                                <1>       ;        2 - u.count = i.size (size of file)
                                <1>       ;
                                <1>       ; !! Retro UNIX 8086 v1 modification:
                                <1>       ;     Argument 1, file descriptor is in BX;
                                <1>       ;     Argument 2, offset is in CX;
                                <1>       ;     Argument 3, ptrname/switch is in DX register.
                                <1>       ;
                                <1>       ; ((Return -> eax = base for offset (position= base+offset))
                                <1>       ;
0000DE5E 890D[84030300]         <1>       mov    [u.base], ecx ; offset
                                <1> seektell0:
0000DE64 8915[88030300]         <1>       mov    [u.count], edx
                                <1>       ; EBX = file descriptor (file number)
0000DE6A E898FCFFFF             <1>       call   getf1
                                <1>       ; EAX = First cluster of the file
                                <1>       ; EBX = File number (Open file number)
                                <1>       ; [u.fofp] = Pointer to File pointer
                                <1>       ; [i.size] = File size
                                <1>
0000DE6F 09C0                   <1>       or     eax, eax
0000DE71 7514                   <1>       jnz    short seektell1
                                <1>
0000DE73 B80A000000             <1>       mov    eax, ERR_FILE_NOT_OPEN
0000DE78 A3[64030300]           <1>       mov    [u.r0], eax
0000DE7D A3[C8030300]           <1>       mov    dword [u.error], eax ; 'file not open !'
0000DE82 E90CE6FFFF             <1>       jmp    error
                                <1>
                                <1> seektell1:
0000DE87 8B1D[74030300]         <1>         mov     ebx, [u.fofp]
0000DE8D 803D[88030300]01       <1>       cmp    byte [u.count], 1
0000DE94 7705                   <1>       ja     short seektell2
0000DE96 7409                   <1>       je     short seektell3
0000DE98 31C0                   <1>       xor    eax, eax
0000DE9A C3                     <1>       retn
                                <1>
                                <1> seektell2:
0000DE9B A1[55040300]           <1>         mov        eax, [i.size]
0000DEA0 C3                     <1>       retn
                                <1>
                                <1> seektell3:
0000DEA1 8B03                   <1>       mov    eax, [ebx]
0000DEA3 C3                     <1>       retn
                                <1>
                                <1> sysintr: ; / set interrupt handling
                                <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                                <1>       ; 07/07/2013 (Retro UNIX 8086 v1)
                                <1>       ;
                                <1>       ; 'sysintr' sets the interrupt handling value. It puts
                                <1>       ; argument of its call in u.intr then branches into 'sysquit'
                                <1>       ; routine. u.tty is checked if to see if a control tty exists.
                                <1>       ; If one does the interrupt character in the tty buffer is
                                <1>       ; cleared and 'sysret'is called. If one does not exits
                                <1>       ; 'sysret' is just called.
                                <1>       ;
                                <1>       ; Calling sequence:
                                <1>       ;     sysintr; arg
                                <1>       ; Argument:
                                <1>       ;     arg - if 0, interrupts (ASCII DELETE) are ignored.
                                <1>       ;         - if 1, intterupts cause their normal result
                                <1>       ;           i.e force an exit.
                                <1>       ;         - if arg is a location within the program,
                                <1>       ;           control is passed to that location when
                                <1>       ;           an interrupt occurs.
                                <1>       ; Inputs: -
                                <1>       ; Outputs: -
                                <1>       ; ..........................................................
                                <1>       ;
                                <1>       ; Retro UNIX 8086 v1 modification:
                                <1>       ;      'sysintr' system call sets u.intr to value of BX
                                <1>       ;     then branches into sysquit.
                                <1>       ;
0000DEA4 66891D[AA030300]       <1>       mov    [u.intr], bx
                                <1>             ; jsr r0,arg; u.intr / put the argument in u.intr
                                <1>             ; br 1f / go into quit routine
0000DEAB E903E6FFFF             <1>       jmp    sysret
                                <1>
                                <1> sysquit:
                                <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                                <1>       ; 07/07/2013 (Retro UNIX 8086 v1)
                                <1>       ;
                                <1>       ; 'sysquit' turns off the quit signal. it puts the argument of
                                <1>       ; the call in u.quit. u.tty is checked if to see if a control
                                <1>       ; tty exists. If one does the interrupt character in the tty
                                <1>       ; buffer is cleared and 'sysret'is called. If one does not exits
                                <1>       ; 'sysret' is just called.
                                <1>       ;
                                <1>       ; Calling sequence:
                                <1>       ;     sysquit; arg
```

```
5363                                    <1>       ; Argument:
5364                                    <1>       ;     arg - if 0, this call diables quit signals from the
5365                                    <1>       ;           typewriter (ASCII FS)
5366                                    <1>       ;         - if 1, quits are re-enabled and cause execution to
5367                                    <1>       ;           cease and a core image to be produced.
5368                                    <1>       ;            i.e force an exit.
5369                                    <1>       ;         - if arg is an addres in the program,
5370                                    <1>       ;           a quit causes control to sent to that
5371                                    <1>       ;           location.
5372                                    <1>       ; Inputs: -
5373                                    <1>       ; Outputs: -
5374                                    <1>       ; ..........................................................
5375                                    <1>       ;
5376                                    <1>       ; Retro UNIX 8086 v1 modification:
5377                                    <1>       ;      'sysquit' system call sets u.quit to value of BX
5378                                    <1>       ;     then branches into 'sysret'.
5379                                    <1>       ;
5380 0000DEB0 66891D[AC030300]         <1>       mov   [u.quit], bx
5381 0000DEB7 E9F7E5FFFF               <1>       jmp   sysret
5382                                    <1>             ; jsr r0,arg; u.quit / put argument in u.quit
5383                                    <1>       ;1:
5384                                    <1>             ; mov u.ttyp,r1 / move pointer to control tty buffer
5385                                    <1>                         ; / to r1
5386                                    <1>             ; beq sysret4 / return to user
5387                                    <1>             ; clrb 6(r1) / clear the interrupt character
5388                                    <1>                     ; / in the tty buffer
5389                                    <1>             ; br sysret4 / return to user
5390                                    <1>
5391                                    <1> syssetuid: ; / set process id
5392                                    <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
5393                                    <1>       ; 07/07/2013 - 02/08/2013 (Retro UNIX 8086 v1)
5394                                    <1>       ;
5395                                    <1>       ; 'syssetuid' sets the user id (u.uid) of the current process
5396                                    <1>       ; to the process id in (u.r0). Both the effective user and
5397                                    <1>       ; u.uid and the real user u.ruid are set to this.
5398                                    <1>       ; Only the super user can make this call.
5399                                    <1>       ;
5400                                    <1>       ; Calling sequence:
5401                                    <1>       ;     syssetuid
5402                                    <1>       ; Arguments: -
5403                                    <1>       ;
5404                                    <1>       ; Inputs: (u.r0) - contains the process id.
5405                                    <1>       ; Outputs: -
5406                                    <1>       ; ..........................................................
5407                                    <1>       ;
5408                                    <1>       ; Retro UNIX 8086 v1 modification:
5409                                    <1>       ;       BL contains the (new) user ID of the current process
5410                                    <1>
5411                                    <1>             ; movb *u.r0,r1 / move process id (number) to r1
5412 0000DEBC 3A1D[B1030300]           <1>       cmp   bl, [u.ruid]
5413                                    <1>             ; cmpb r1,u.ruid / is it equal to the real user
5414                                    <1>                         ; / id number
5415 0000DEC2 741E                     <1>       je    short setuid1
5416                                    <1>             ; beq 1f / yes
5417 0000DEC4 803D[B0030300]00         <1>       cmp   byte [u.uid], 0 ; 02/08/2013
5418                                    <1>             ; tstb u.uid / no, is current user the super user?
5419                                    <1>       ;ja   error
5420                                    <1>             ; bne error4 / no, error
5421 0000DECB 760F                     <1>       jna   short setuid0
5422 0000DECD C705[C8030300]0B00-      <1>       mov   dword [u.error], ERR_NOT_SUPERUSER  ; 11
5422 0000DED5 0000                     <1>
5423                                    <1>                           ;  'permission denied !' error
5424 0000DED7 E9B7E5FFFF               <1>       jmp   error
5425                                    <1> setuid0:
5426 0000DEDC 881D[B1030300]           <1>       mov   [u.ruid], bl
5427                                    <1> setuid1: ; 1:
5428 0000DEE2 881D[B0030300]           <1>       mov   [u.uid], bl ; 02/08/2013
5429                                    <1>             ; movb r1,u.uid / put process id in u.uid
5430                                    <1>             ; movb r1,u.ruid / put process id in u.ruid
5431 0000DEE8 E9C6E5FFFF               <1>       jmp   sysret
5432                                    <1>             ; br sysret4 / system return
5433                                    <1>
5434                                    <1> sysgetuid: ; < get user id >
5435                                    <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
5436                                    <1>       ; 07/07/2013 (Retro UNIX 8086 v1)
5437                                    <1>       ;
5438                                    <1>       ; 'sysgetuid' returns the real user ID of the current process.
5439                                    <1>       ; The real user ID identifies the person who is logged in,
5440                                    <1>       ; in contradistinction to the effective user ID, which
5441                                    <1>       ; determines his access permission at each moment. It is thus
5442                                    <1>       ; useful to programs which operate using the 'set user ID'
5443                                    <1>       ; mode, to find out who invoked them.
5444                                    <1>       ;
5445                                    <1>       ; Calling sequence:
5446                                    <1>       ;     syssetuid
5447                                    <1>       ; Arguments: -
5448                                    <1>       ;
5449                                    <1>       ; Inputs: -
5450                                    <1>       ; Outputs: (u.r0) - contains the real user's id.
5451                                    <1>       ; ..........................................................
5452                                    <1>       ;
5453                                    <1>       ; Retro UNIX 8086 v1 modification:
5454                                    <1>       ;       AL contains the real user ID at return.
5455                                    <1>       ;
5456 0000DEED 0FB605[B1030300]         <1>       movzx  eax, byte [u.ruid]
5457 0000DEF4 A3[64030300]             <1>       mov   [u.r0], eax
5458                                    <1>             ; movb u.ruid,*u.r0 / move the real user id to (u.r0)
5459 0000DEF9 E9B5E5FFFF               <1>       jmp   sysret
5460                                    <1>             ; br sysret4 / systerm return, sysret
5461                                    <1>
5462                                    <1> anyi:
5463                                    <1>       ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
```

```
                        <1>       ; Major Modification!
                        <1>       ; TRDOS 386 does not permit to delete a file while it is open
                        <1>       ; The role of 'anyi' procedure has beeen changed to ensure that.
                        <1>       ;
                        <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                        <1>       ; 25/04/2013 (Retro UNIX 8086 v1)
                        <1>       ;
                        <1>       ; 'anyi' is called if a file deleted while open.
                        <1>       ; "anyi" checks to see if someone else has opened this file.
                        <1>       ;
                        <1>       ; INPUTS ->
                        <1>       ;    r1 - contains an i-number
                        <1>       ;    fsp - start of table containing open files
                        <1>       ;
                        <1>       ; OUTPUTS ->
                        <1>       ;    "deleted" flag set in fsp entry of another occurrence of
                        <1>       ;         this file and r2 points 1st word of this fsp entry.
                        <1>       ;    if file not found - bit in i-node map is cleared
                        <1>       ;                        (i-node is freed)
                        <1>       ;                 all blocks related to i-node are freed
                        <1>       ;                all flags in i-node are cleared
                        <1>       ; ((AX = R1)) input
                        <1>       ;
                        <1>       ;    (Retro UNIX Prototype : 02/12/2012, UNIXCOPY.ASM)
                        <1>         ;    ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
                        <1>       ;
                        <1>       ; / r1 contains an i-number
                        <1>
                        <1>       ; TRDOS 386 (06/10/2016)
                        <1>       ;
                        <1>       ; INPUT:
                        <1>       ;     EAX = First Cluster
                        <1>       ;      DL = Logical DOS Drive Number
                        <1>       ;
                        <1>       ; OUTPUT:
                        <1>       ;     CF = 1 -> EBX = File Handle/Number/Index
                        <1>       ;     CF = 0 -> EBX = 0
                        <1>       ;
                        <1>       ; Modified Registers: EBX
                        <1>
0000DEFE 31DB           <1>       xor    ebx, ebx
                        <1> anyi_0:
0000DF00 80BB[2A630100]00 <1>     cmp    byte [ebx+OF_MODE], 0 ; 0 = empty entry
0000DF07 770A           <1>       ja     short anyi_2 ; 1 (r), 2 (w) or 3 (r&w)
                        <1> anyi_1:
0000DF09 FEC3           <1>       inc    bl
0000DF0B 80FB0A         <1>       cmp    bl, OPENFILES ; max. count of open files
0000DF0E 72F0           <1>       jb     short anyi_0
0000DF10 31C0           <1>       xor    eax, eax
0000DF12 C3             <1>       retn
                        <1> anyi_2:
0000DF13 3A93[20630100] <1>       cmp    dl, [ebx+OF_DRIVE]
0000DF19 75EE           <1>       jne    short anyi_1
0000DF1B 66C1E302       <1>       shl    bx, 2 ; *4 (dword offset)
0000DF1F 3B83[F8620100] <1>       cmp    eax, [ebx+OF_FCLUSTER]
0000DF25 7406           <1>       je     short anyi_3
0000DF27 66C1EB02       <1>       shr    bx, 2 ; /4 (byte offset)
0000DF2B EBDC           <1>       jmp    short anyi_1
                        <1> anyi_3:
0000DF2D 66C1EB02       <1>       shr    bx, 2 ; /4 (bytes offset) (index)
0000DF31 F9             <1>       stc
0000DF32 C3             <1>       retn
                        <1>
                        <1> ; Retro UNIX 386 v1 Kernel (v0.2) - u7.s
                        <1> ; Last Modification: 14/11/2015
                        <1>
                        <1> sysmount: ; / mount file system
                        <1>       ; 24/10/2016 - TRDOS 386 (TRDOS v2.0)
                        <1>       ; temporary !
0000DF33 B80F000000     <1>       mov    eax, ERR_DRV_NOT_RDY ; drive not ready !
0000DF38 A3[C8030300]   <1>         mov     [u.error], eax
0000DF3D A3[64030300]   <1>         mov     [u.r0], eax
0000DF42 E94CE5FFFF     <1>       jmp    error
                        <1>
                        <1> sysumount: ; / special dismount file system
                        <1>       ; 24/10/2016 - TRDOS 386 (TRDOS v2.0)
                        <1>       ; temporary !
0000DF47 B80F000000     <1>       mov    eax, ERR_DRV_NOT_RDY ; drive not ready !
0000DF4C A3[C8030300]   <1>         mov     [u.error], eax
0000DF51 A3[64030300]   <1>         mov     [u.r0], eax
0000DF56 E938E5FFFF     <1>       jmp    error
                        <1>
                        <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS9.INC
                        <1> ; Last Modification: 09/12/2015
                        <1>
                        <1> syssleep:
                        <1>       ; 29/06/2015 - (Retro UNIX 386 v1)
                        <1>       ; 11/06/2014 - (Retro UNIX 8086 v1)
                        <1>       ;
                        <1>       ; Retro UNIX 8086 v1 feature only
                        <1>       ; (INPUT -> none)
                        <1>       ;
0000DF5B 0FB61D[B3030300] <1>     movzx  ebx, byte [u.uno] ; process number
0000DF62 8AA3[7F000300] <1>       mov    ah, [ebx+p.ttyc-1] ; current/console tty
0000DF68 E834120000     <1>       call   sleep
0000DF6D E941E5FFFF     <1>       jmp    sysret
                        <1>
                        <1> _vp_clr:
                        <1>       ; Reset/Clear Video Page
                        <1>       ;
                        <1>       ; 30/06/2015 - (Retro UNIX 386 v1)
                        <1>       ; 21/05/2013 - 30/10/2013(Retro UNIX 8086 v1) (U0.ASM)
```

```
                                <1>       ;
                                <1>       ; Retro UNIX 8086 v1 feature only !
                                <1>       ;
                                <1>       ; INPUTS ->
                                <1>       ;   BH = video page number
                                <1>       ;
                                <1>       ; OUTPUT ->
                                <1>       ;   none
                                <1>       ; ((Modified registers: eAX, BH, eCX, eDX, eSI, eDI))
                                <1>       ;
                                <1>       ; 04/12/2013
0000DF72 28C0                   <1>       sub    al, al
                                <1>       ; al = 0 (clear video page)
                                <1>       ; bh = video page ; 13/05/2016
0000DF74 B407                   <1>       mov    ah, 07h
                                <1>       ; ah = 7 (attribute/color)
0000DF76 6631C9                 <1>       xor    cx, cx ; 0, left upper column (cl) & row (cl)
0000DF79 66BA4F18               <1>       mov    dx, 184Fh ; right lower column & row (dl=24, dh=79)
0000DF7D E8883AFFFF             <1>       call   _scroll_up
                                <1>       ; bh = video page
0000DF82 6631D2                 <1>       xor    dx, dx ; 0 (cursor position)
0000DF85 E9BE3DFFFF             <1>       jmp    _set_cpos
                                <1>
                                <1> sysmsg:
                                <1>       ; 13/05/2016
                                <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; 01/07/2015 - 11/11/2015 (Retro UNIX 386 v1)
                                <1>       ; Print user-application message on user's console tty
                                <1>       ;
                                <1>       ; Input -> EBX = Message address
                                <1>       ;         ECX = Message length (max. 255)
                                <1>       ;         DL = Color (IBM PC Rombios color attributes)
                                <1>       ;
0000DF8A 81F9FF000000           <1>       cmp    ecx, MAX_MSG_LEN ; 255
0000DF90 0F871DE5FFFF           <1>       ja     sysret ; nothing to do with big message size
0000DF96 08C9                   <1>       or     cl, cl
0000DF98 0F8415E5FFFF           <1>       jz     sysret
0000DF9E 20D2                   <1>       and    dl, dl
0000DFA0 7502                   <1>       jnz    short sysmsg0
0000DFA2 B207                   <1>       mov    dl, 07h ; default color
                                <1>              ; (black background, light gray character)
                                <1> sysmsg0:
0000DFA4 891D[84030300]         <1>       mov    [u.base], ebx
0000DFAA 8815[2F520100]         <1>       mov    [ccolor], dl ; color attributes
0000DFB0 89E5                   <1>       mov    ebp, esp
0000DFB2 31DB                   <1>       xor    ebx, ebx ; 0
0000DFB4 891D[8C030300]         <1>       mov    [u.nread], ebx ; 0
                                <1>       ;
0000DFBA 381D[C6030300]         <1>       cmp    [u.kcall], bl ; 0
0000DFC0 7769                   <1>       ja     short sysmsgk ; Temporary (01/07/2015)
                                <1>       ;
0000DFC2 890D[88030300]         <1>       mov    [u.count], ecx
0000DFC8 41                     <1>       inc    ecx ; + 00h ; ASCIIZ
0000DFC9 29CC                   <1>       sub    esp, ecx
0000DFCB 89E7                   <1>       mov    edi, esp
0000DFCD 89E6                   <1>       mov    esi, esp
0000DFCF 66891D[C4030300]       <1>       mov    [u.pcount], bx ; reset page (phy. addr.) counter
                                <1>       ; 11/11/2015
0000DFD6 8A25[94030300]         <1>       mov    ah, [u.ttyp] ; recent open tty
                                <1>       ; 0 = none
0000DFDC FECC                   <1>       dec    ah
0000DFDE 790C                   <1>       jns    short sysmsg1
0000DFE0 8A1D[B3030300]         <1>       mov    bl, [u.uno] ; process number
0000DFE6 8AA3[7F000300]         <1>       mov    ah, [ebx+p.ttyc-1] ; user's (process's) console tty
                                <1> sysmsg1:
0000DFEC 8825[96030300]         <1>       mov    [u.ttyn], ah
                                <1> sysmsg2:
0000DFF2 E82F080000             <1>       call   cpass
0000DFF7 7416                   <1>       jz     short sysmsg5
0000DFF9 AA                     <1>       stosb
0000DFFA 20C0                   <1>       and    al, al
0000DFFC 75F4                   <1>       jnz    short sysmsg2
                                <1> sysmsg3:
0000DFFE 80FC07                 <1>       cmp    ah, 7 ; tty number
0000E001 7711                   <1>       ja     short sysmsg6 ; serial port
0000E003 E83E000000             <1>       call   print_cmsg
                                <1> sysmsg4:
0000E008 89EC                   <1>       mov    esp, ebp
0000E00A E9A4E4FFFF             <1>       jmp    sysret
                                <1> sysmsg5:
0000E00F C60700                 <1>       mov    byte [edi], 0
0000E012 EBEA                   <1>       jmp    short sysmsg3
                                <1> sysmsg6:
0000E014 8A06                   <1>       mov    al, [esi]
0000E016 E883110000             <1>       call   sndc
0000E01B 72EB                   <1>       jc     short sysmsg4
0000E01D 803E00                 <1>       cmp    byte [esi], 0  ; 0 is stop character
0000E020 76E6                   <1>       jna    short sysmsg4
0000E022 46                     <1>       inc    esi
0000E023 8A25[96030300]         <1>       mov    ah, [u.ttyn]
0000E029 EBE9                   <1>       jmp    short sysmsg6
                                <1>
                                <1> sysmsgk: ; Temporary (01/07/2015)
                                <1>       ; The message has been sent by Kernel (ASCIIZ string)
                                <1>       ; (ECX -character count- will not be considered)
0000E02B 8B35[84030300]         <1>       mov    esi, [u.base]
0000E031 8A25[2E520100]         <1>       mov    ah, [ptty] ; present/current screen (video page)
0000E037 8825[96030300]         <1>       mov    [u.ttyn], ah
0000E03D C605[C6030300]00       <1>       mov    byte [u.kcall], 0
0000E044 EBB8                   <1>       jmp    short sysmsg3
                                <1>
                                <1> print_cmsg:
```

```
                                <1>       ; 13/05/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; 01/07/2015 (Retro UNIX 386 v1)
                                <1>       ;
                                <1>       ; print message (on user's console tty)
                                <1>       ;     with requested color
                                <1>       ;
                                <1>       ; INPUTS:
                                <1>       ;     esi = message address
                                <1>       ;     [u.ttyn] = tty number (0 to 7)
                                <1>       ;     [ccolor] = color attributes (IBM PC BIOS colors)
                                <1>
0000E046 8A3D[96030300]         <1>       mov   bh, [u.ttyn]
                                <1>       ;mov   bh, ah
                                <1>
0000E04C AC                     <1>       lodsb
                                <1> pcmsg1:
0000E04D 56                     <1>       push  esi
0000E04E 8A1D[2F520100]         <1>       mov   bl, [ccolor]
                                <1>       ;mov   bh, [u.ttyn]
0000E054 E8593CFFFF             <1>       call  _write_tty
0000E059 5E                     <1>       pop   esi
0000E05A AC                     <1>       lodsb
0000E05B 20C0                   <1>       and   al, al  ; 0
0000E05D 75EE                   <1>       jnz   short pcmsg1
0000E05F C3                     <1>       retn
                                <1>
                                <1> sysgeterr:
                                <1>       ; 09/12/2015
                                <1>       ; 21/09/2015 - (Retro UNIX 386 v1 feature only!)
                                <1>       ; Get last error number or page fault count
                                <1>       ; (for debugging)
                                <1>       ;
                                <1>       ; Input -> EBX = return type
                                <1>       ;         0 = last error code (which is in 'u.error')
                                <1>       ;         FFFFFFFFh = page fault count for running process
                                <1>       ;         FFFFFFFEh = total page fault count
                                <1>       ;         1 .. FFFFFFFDh = undefined
                                <1>       ;
                                <1>       ; Output -> EAX = last error number or page fault count
                                <1>       ;         (depending on EBX input)
                                <1>       ;
0000E060 21DB                   <1>       and   ebx, ebx
0000E062 750B                   <1>       jnz   short glerr_2
                                <1> glerr_0:
0000E064 A1[C8030300]           <1>       mov   eax, [u.error]
                                <1> glerr_1:
0000E069 A3[64030300]           <1>       mov   [u.r0], eax
0000E06E C3                     <1>       retn
                                <1> glerr_2:
0000E06F 43                     <1>       inc   ebx ; FFFFFFFFh -> 0, FFFFFFFEh -> FFFFFFFFh
0000E070 74FD                   <1>       jz    short glerr_2 ; page fault count for process
0000E072 43                     <1>       inc   ebx ; FFFFFFFFh -> 0
0000E073 75EF                   <1>       jnz   short glerr_0
0000E075 A1[80050300]           <1>       mov   eax, [PF_Count] ; total page fault count
0000E07A EBED                   <1>         jmp     short glerr_1
                                <1> glerr_3:
0000E07C A1[CC030300]           <1>       mov   eax, [u.pfcount]
0000E081 EBE6                   <1>       jmp   short glerr_1
                                <1>
                                <1> load_and_run_file:
                                <1>       ; 22/01/2017
                                <1>       ; 07/01/2017
                                <1>       ; 04/01/2017
                                <1>       ; 24/10/2016
                                <1>       ; 24/04/2016, 02/05/2016, 03/05/2016, 06/05/2016
                                <1>       ; 23/04/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>       ; 23/10/2015 (Retro UNIX 386 v1, 'sysexec')
                                <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
                                <1>       ; 03/06/2013 - 06/12/2013 (Retro UNIX 8086 v1)
                                <1>       ; EAX = First Cluster number
                                <1>       ; EDX = File Size
                                <1>       ; ESI = Argument list address
                                <1>       ; [argc] = argument count
                                <1>       ; [u.nread] = argument list length
                                <1>       ; [esp] = return address to the caller (*)
                                <1>       ;
0000E083 8935[4C040300]         <1>       mov   [argv], esi
0000E089 8915[55040300]         <1>       mov   [i.size], edx
0000E08F A3[51040300]           <1>       mov   [ii], eax
                                <1>
                                <1>       ;sti   ; 07/01/2017
                                <1>       ;mov   eax, [k_page_dir]
                                <1>       ;mov   [u.pgdir], eax
0000E094 31C0                   <1>       xor   eax, eax ; clc ; *** ; 04/01/2017
                                <1>       ;mov   [u.r0], eax ; 0 ; 07/01/2017
                                <1>
                                <1>       ; 06/05/2016
                                <1>       ; Set 'sysexit' return order to MainProg
                                <1>       ;
0000E096 58                     <1>       pop   eax ; * 'loc_load_and_run_file_8:' address
                                <1>       ;; 22/01/2017
                                <1>       ;;cli ; 07/01/2017
0000E097 8B25[9C510100]         <1>       mov   esp, [tss.esp0]
                                <1>       ;
                                <1>       ; 'loc_load_run_file_8' address has
                                <1>       ; 'jmp loc_file_rw_restore_retn' instruction
                                <1>       ; 'loc_file_rw_restore_retn:' will return to
                                <1>       ; [mainprog_return_addr]
                                <1>       ; just after 'call command_interpreter'
                                <1>       ;
0000E09D 68[3B630000]           <1>       push  _end_of_mainprog ; we must not return to here !
0000E0A2 FF35[845F0100]         <1>       push  dword [mainprog_return_addr]
```

```
5770 0000E0A8 89E5                <1>       mov    ebp, esp ; **
5771                              <1>       ;
5772 0000E0AA 9C                  <1>       pushfd  ; EFLAGS      ; IRETD ; ***
5773 0000E0AB 6A08                <1>       push   KCODE ; cs    ; IRETD
5774 0000E0AD 50                  <1>       push   eax ; * (eip) ; IRETD
5775 0000E0AE 8925[5C030300]      <1>       mov    [u.sp], esp
5776                              <1>       ;mov   byte [u.quant], time_count
5777 0000E0B4 1E                  <1>       push   ds
5778 0000E0B5 06                  <1>       push   es
5779 0000E0B6 0FA0                <1>       push   fs
5780 0000E0B8 0FA8                <1>       push   gs
5781                              <1>       ;mov   eax, [u.r0]
5782 0000E0BA 29C0                <1>       sub    eax, eax
5783 0000E0BC 60                  <1>       pushad
5784 0000E0BD 68[B3C40000]        <1>       push   sysret
5785                              <1>       ;push  sysrel1 ; 07/01/2017
5786 0000E0C2 8925[60030300]      <1>       mov    [u.usp], esp
5787                              <1>       ;
5788 0000E0C8 E85E060000          <1>       call   wswap ; Save MainProg (process 1) 'u' structure
5789                              <1>                    ; and registers for return (from program)
5790 0000E0CD 89EC                <1>       mov    esp, ebp ; **
5791                              <1>       ;;22/01/2017
5792                              <1>       ;;sti ; 07/01/2017
5793 0000E0CF 50                  <1>       push   eax  ; * 'loc_load_and_run_file_8:' address
5794                              <1>       ;
5795                              <1>       ;;; 02/05/2016
5796                              <1>       ;;; Create a new process (parent: MainProg)
5797 0000E0D0 31F6                <1>       xor    esi, esi
5798                              <1> cnpm_1: ; search p.stat table for unused process number
5799 0000E0D2 46                  <1>       inc    esi
5800 0000E0D3 80BE[AF000300]00    <1>       cmp    byte [esi+p.stat-1], 0 ; SFREE
5801                              <1>                           ; is process active, unused, dead
5802 0000E0DA 760B                <1>       jna    short cnpm_2 ; it's unused so branch
5803 0000E0DC 6683FE10            <1>       cmp    si, nproc    ; all processes checked
5804 0000E0E0 72F0                <1>       jb     short cnpm_1    ; no, branch back
5805 0000E0E2 E9CA82FFFF          <1>       jmp    panic
5806                              <1> cnpm_2:
5807 0000E0E7 A1[B8030300]        <1>       mov    eax, [u.pgdir] ; page directory of MainProg
5808 0000E0EC A3[BC030300]        <1>       mov    [u.ppgdir], eax ; parent's page directory
5809 0000E0F1 E8836AFFFF          <1>       call   allocate_page
5810 0000E0F6 0F82B582FFFF        <1>       jc     panic
5811                              <1>       ; EAX = UPAGE (user structure page) address
5812 0000E0FC A3[B4030300]        <1>       mov    [u.upage], eax ; memory page for 'user' struct (child)
5813 0000E101 89F7                <1>       mov    edi, esi
5814 0000E103 66C1E702            <1>       shl    di, 2
5815 0000E107 8987[BC000300]      <1>       mov    [edi+p.upage-4], eax ; memory page for 'user' struct
5816 0000E10D E8E16AFFFF          <1>       call   clear_page ; 03/05/2016
5817                              <1>       ;movzx eax, byte [p.ttyc] ; console tty (for MainProg)
5818 0000E112 6629C0              <1>       sub    ax, ax ; 0
5819 0000E115 668986[7F000300]    <1>       mov     [esi+p.ttyc-1], ax ; al - set child's console tty
5820                              <1>                                ; ah - reset child's wait channel
5821 0000E11C 89F0                <1>       mov    eax, esi
5822 0000E11E A2[B3030300]        <1>       mov    [u.uno], al ; child process number
5823 0000E123 FE86[AF000300]      <1>         inc     byte [esi+p.stat-1] ; 1, SRUN
5824 0000E129 66D1E6              <1>       shl    si, 1 ; multiply si by 2 to get index into p.pid table
5825 0000E12C 66FF05[4E030300]    <1>       inc    word [mpid] ; increment m.pid; get a new process name
5826 0000E133 66A1[4E030300]      <1>       mov    ax, [mpid]
5827 0000E139 668986[1E000300]    <1>       mov    [esi+p.pid-2], ax ; put new process name
5828                              <1>                              ; in child process' name slot
5829                              <1>       ;mov   ax, [p.pid]  ; get process name of MainProg
5830 0000E140 66B80100            <1>       mov    ax, 1
5831 0000E144 668986[3E000300]    <1>       mov    [esi+p.ppid-2], ax ; put parent process name
5832                              <1>                                ; in parent process slot for child
5833 0000E14B 6648                <1>       dec    ax ; 0
5834 0000E14D 66A3[94030300]      <1>       mov    [u.ttyp], ax ; 0
5835                              <1>       ;;;
5836 0000E153 A1[51040300]        <1>       mov    eax,  [ii]
5837                              <1>       ; Retro UNIX 386 v1, 'sysexec' (u2.s)
5838 0000E158 E83E100000          <1>       call   iopen
5839                              <1>       ; 06/06/2016
5840 0000E15D C605[A9030300]01    <1>       mov    byte [u.pri], 1 ; normal priority
5841                              <1>       ;
5842 0000E164 EB0A                <1>       jmp    short sysexec_7 ; 02/05/2016
5843                              <1>
5844                              <1> sysexec_6:
5845                              <1>       ; 04/01/2017
5846                              <1>       ; 24/10/2016
5847                              <1>       ;;02/05/2016
5848                              <1>       ; 23/04/2016
5849                              <1>       ; 18/10/2015 ('sysexec_6')
5850                              <1>       ; 23/06/2015
5851 0000E166 A1[B8030300]        <1>       mov    eax, [u.pgdir] ; physical address of page directory
5852                              <1>       ;;cmp  eax, [k_page_dir] ; TRDOS MainProg ?
5853                              <1>       ;;je   short sysexec_7
5854 0000E16B E8426BFFFF          <1>       call   deallocate_page_dir
5855                              <1> sysexec_7:
5856 0000E170 E8726AFFFF          <1>       call   make_page_dir
5857 0000E175 0F823682FFFF        <1>       jc     panic  ; allocation error
5858                              <1>                     ; after a deallocation would be nonsence !?
5859                              <1>       ; 24/07/2015
5860                              <1>       ; map kernel pages (1st 4MB) to PDE 0
5861                              <1>       ;     of the user's page directory
5862                              <1>       ;     (It is needed for interrupts!)
5863                              <1>       ; 18/10/2015
5864 0000E17B 8B15[00520100]      <1>       mov    edx, [k_page_dir] ; Kernel's page directory
5865 0000E181 8B02                <1>       mov    eax, [edx] ; physical address of
5866                              <1>                         ; kernel's first page table (1st 4 MB)
5867                              <1>                         ; (PDE 0 of kernel's page directory)
5868 0000E183 8B15[B8030300]      <1>       mov    edx, [u.pgdir]
5869 0000E189 8902                <1>       mov    [edx], eax ; PDE 0 (1st 4MB)
5870                              <1>       ;
5871                              <1>       ; 20/07/2015
```

```
5872 0000E18B BB00004000          <1>       mov    ebx, CORE ; start address = 0 (virtual) + CORE
5873                              <1>       ; 18/10/2015
5874 0000E190 BE[3C040300]        <1>       mov    esi, pcore ; physical start address
5875                              <1> sysexec_8:
5876 0000E195 B907000000          <1>       mov    ecx, PDE_A_USER + PDE_A_WRITE + PDE_A_PRESENT
5877 0000E19A E8666AFFFF          <1>       call   make_page_table
5878 0000E19F 0F820C82FFFF        <1>       jc     panic
5879                              <1>       ;mov   ecx, PTE_A_USER + PTE_A_WRITE + PTE_A_PRESENT
5880 0000E1A5 E8696AFFFF          <1>       call   make_page ; make new page, clear and set the pte
5881 0000E1AA 0F820182FFFF        <1>       jc     panic
5882                              <1>       ;
5883 0000E1B0 8906                <1>       mov    [esi], eax ; 24/06/2015
5884                              <1>       ; ebx = virtual address (24/07/2015)
5885 0000E1B2 E80170FFFF          <1>       call   add_to_swap_queue
5886                              <1>       ; 18/10/2015
5887 0000E1B7 81FE[40040300]      <1>       cmp    esi, ecore ; user's stack (last) page ?
5888 0000E1BD 740C                <1>       je     short sysexec_9 ; yes
5889 0000E1BF BE[40040300]        <1>       mov    esi, ecore  ; physical address of the last page
5890                              <1>       ; 20/07/2015
5891 0000E1C4 BB00F0FFFF          <1>       mov    ebx, (ECORE - PAGE_SIZE) + CORE
5892                              <1>       ; ebx = virtual end address + segment base address - 4K
5893 0000E1C9 EBCA                <1>         jmp     short sysexec_8
5894                              <1> sysexec_9:
5895                              <1>       ; 24/04/2016
5896                              <1>       ; 18/10/2015
5897                              <1>       ; 26/08/2015
5898                              <1>       ; 25/06/2015
5899                              <1>       ; move arguments from kernel stack to [ecore]
5900                              <1>       ; (argument list/line will be copied from kernel stack
5901                              <1>       ; frame to the last (stack) page of user's core memory)
5902                              <1>       ; 18/10/2015
5903 0000E1CB 8B3D[40040300]      <1>       mov    edi, [ecore]
5904 0000E1D1 81C700100000        <1>       add    edi, PAGE_SIZE
5905 0000E1D7 0FB705[4A040300]    <1>       movzx  eax, word [argc]
5906 0000E1DE 09C0                <1>       or     eax, eax
5907 0000E1E0 7509                <1>       jnz    short sysexec_10
5908 0000E1E2 89FB                <1>       mov    ebx, edi
5909 0000E1E4 83EB04              <1>       sub    ebx, 4
5910 0000E1E7 8903                <1>       mov    [ebx], eax ; 0
5911 0000E1E9 EB44                <1>       jmp    short sysexec_13
5912                              <1> sysexec_10:
5913 0000E1EB 8B0D[8C030300]      <1>       mov    ecx, [u.nread]
5914                              <1>       ;mov   esi, TextBuffer
5915 0000E1F1 8B35[4C040300]      <1>       mov    esi, [argv] ; 24/04/2016 (TRDOS 386  = TRDOS v2.0)
5916 0000E1F7 29CF                <1>       sub    edi, ecx ; page end address - argument list length
5917 0000E1F9 89C2                <1>       mov    edx, eax
5918 0000E1FB FEC2                <1>       inc    dl ; argument count + 1 for argc value
5919 0000E1FD C0E202              <1>       shl    dl, 2  ; 4 * (argument count + 1)
5920 0000E200 89FB                <1>       mov    ebx, edi
5921 0000E202 80E3FC              <1>       and    bl, 0FCh ; 32 bit (dword) alignment
5922 0000E205 29D3                <1>       sub    ebx, edx
5923 0000E207 89FA                <1>       mov    edx, edi
5924 0000E209 F3A4                <1>       rep    movsb
5925 0000E20B 89D6                <1>       mov    esi, edx
5926 0000E20D 89DF                <1>       mov    edi, ebx
5927 0000E20F BA00F0BFFF          <1>       mov    edx, ECORE - PAGE_SIZE ; virtual addr. of the last page
5928 0000E214 2B15[40040300]      <1>       sub    edx, [ecore] ; difference (virtual - physical)
5929 0000E21A AB                  <1>       stosd  ; eax = argument count
5930                              <1> sysexec_11:
5931 0000E21B 89F0                <1>       mov    eax, esi
5932 0000E21D 01D0                <1>       add    eax, edx
5933 0000E21F AB                  <1>       stosd  ; eax = virtual address
5934 0000E220 FE0D[4A040300]      <1>       dec    byte [argc]
5935 0000E226 7407                <1>       jz     short sysexec_13
5936                              <1> sysexec_12:
5937 0000E228 AC                  <1>       lodsb
5938 0000E229 20C0                <1>       and    al, al
5939 0000E22B 75FB                <1>       jnz    short sysexec_12
5940 0000E22D EBEC                <1>       jmp    short sysexec_11
5941                              <1> sysexec_13:
5942                              <1>       ; 24/10/2016
5943                              <1>       ; 24/04/2016 - TRDOS 386 (TRDOS v2.0)
5944                              <1>       ; 23/06/2015 - 19/10/2015 (Retro UNIX 386 v1, 'sysexec_13')
5945                              <1>       ;
5946                              <1>       ; moving arguments to [ecore] is OK here..
5947                              <1>       ;
5948                              <1>       ; ebx = beginning addres of argument list pointers
5949                              <1>             ;      in user's stack
5950 0000E22F 2B1D[40040300]      <1>       sub    ebx, [ecore]
5951 0000E235 81C300F0BFFF        <1>       add     ebx, (ECORE - PAGE_SIZE)
5952                              <1>             ; end of core - 4096 (last page)
5953                              <1>             ; (virtual address)
5954 0000E23B 891D[4C040300]      <1>       mov    [argv], ebx
5955 0000E241 891D[90030300]      <1>       mov    [u.break], ebx ; available user memory
5956                              <1>       ;
5957 0000E247 29C0                <1>       sub    eax, eax
5958 0000E249 C705[88030300]2000- <1>       mov    dword [u.count], 32 ; Executable file header size
5958 0000E251 0000                <1>
5959 0000E253 C705[74030300]-     <1>       mov    dword [u.fofp], u.off
5959 0000E259 [80030300]          <1>
5960 0000E25D A3[80030300]        <1>       mov    [u.off], eax ; 0
5961 0000E262 A3[84030300]        <1>       mov    [u.base], eax ; 0, start of user's core (virtual)
5962                              <1>       ; 24/10/2016
5963 0000E267 A0[C6520100]        <1>       mov    al, [Current_Drv]
5964 0000E26C A2[46030300]        <1>       mov    [cdev], al
5965                              <1>       ;
5966 0000E271 A1[51040300]        <1>       mov    eax, [ii] ; Fist Cluster of the Program (PRG) file
5967                              <1>       ; EAX = First cluster of the executable file
5968 0000E276 E832010000          <1>       call   readi
5969                              <1>
5970 0000E27B 8B0D[90030300]      <1>       mov    ecx, [u.break] ; top of user's stack (physical addr.)
5971 0000E281 890D[88030300]      <1>       mov    [u.count], ecx ; save for overrun check
```

```
5972                                    <1>       ;
5973 0000E287 8B0D[8C030300]           <1>       mov    ecx, [u.nread]
5974 0000E28D 890D[90030300]           <1>       mov    [u.break], ecx ; virtual address (offset from start)
5975 0000E293 80F920                   <1>       cmp    cl, 32
5976 0000E296 7540                     <1>         jne     short sysexec_15
5977                                    <1>       ;:
5978                                    <1>       ; Retro UNIX 386 v1 (32 bit) executable file header format
5979 0000E298 8B35[3C040300]           <1>       mov    esi, [pcore] ; start address of user's core memory
5980                                    <1>                           ; (phys. start addr. of the exec. file)
5981 0000E29E AD                       <1>       lodsd
5982 0000E29F 663DEB1E                 <1>       cmp    ax, 1EEBh ; EBH, 1Eh -> jump to +32
5983 0000E2A3 7533                     <1>       jne    short sysexec_15
5984 0000E2A5 AD                       <1>       lodsd
5985 0000E2A6 89C1                     <1>       mov    ecx, eax ; text (code) section size
5986 0000E2A8 AD                       <1>       lodsd
5987 0000E2A9 01C1                     <1>       add    ecx, eax ; + data section size (initialized data)
5988 0000E2AB 89CB                     <1>       mov    ebx, ecx
5989 0000E2AD AD                       <1>       lodsd
5990 0000E2AE 01C3                     <1>       add    ebx, eax ; + bss section size (for overrun checking)
5991 0000E2B0 3B1D[88030300]           <1>       cmp    ebx, [u.count]
5992 0000E2B6 7711                     <1>       ja     short sysexec_14  ; program overruns stack !
5993                                    <1>       ;
5994                                    <1>       ; add bss section size to [u.break]
5995 0000E2B8 0105[90030300]           <1>       add    [u.break], eax
5996                                    <1>       ;
5997 0000E2BE 83E920                   <1>       sub    ecx, 32  ; header size (already loaded)
5998                                    <1>       ;cmp   ecx, [u.count]
5999                                    <1>       ;jnb   short sysexec_16
6000 0000E2C1 890D[88030300]           <1>       mov    [u.count], ecx ; required read count
6001 0000E2C7 EB29                     <1>       jmp    short sysexec_16
6002                                    <1> sysexec_14:
6003                                    <1>       ; insufficient (out of) memory
6004 0000E2C9 C705[C8030300]0400-      <1>       mov    dword [u.error], ERR_MINOR_IM ; 1
6004 0000E2D1 0000                     <1>
6005 0000E2D3 E9BBE1FFFF               <1>       jmp    error
6006                                    <1> sysexec_15:
6007 0000E2D8 8B15[55040300]           <1>         mov  edx, [i.size] ; file size
6008 0000E2DE 29CA                     <1>       sub    edx, ecx ; file size - loaded bytes
6009 0000E2E0 7626                     <1>       jna    short sysexec_17 ; no need to next read
6010 0000E2E2 01D1                     <1>       add    ecx, edx ; [i.size]
6011 0000E2E4 3B0D[88030300]           <1>       cmp    ecx, [u.count] ; overrun check (!)
6012 0000E2EA 77DD                     <1>       ja     short sysexec_14
6013 0000E2EC 8915[88030300]           <1>       mov    [u.count], edx
6014                                    <1> sysexec_16:
6015 0000E2F2 A1[51040300]             <1>       mov    eax, [ii] ; first cluster
6016 0000E2F7 E8B1000000               <1>       call   readi
6017 0000E2FC 8B0D[8C030300]           <1>       mov    ecx, [u.nread]
6018 0000E302 010D[90030300]           <1>       add    [u.break], ecx
6019                                    <1> sysexec_17:
6020 0000E308 A1[51040300]             <1>       mov    eax, [ii] ; first cluster
6021 0000E30D E88A0E0000               <1>       call   iclose
6022 0000E312 31C0                     <1>       xor     eax, eax
6023 0000E314 FEC0                     <1>       inc    al
6024 0000E316 66A3[AA030300]           <1>       mov    [u.intr], ax ; 1 (interrupt/time-out is enabled)
6025 0000E31C 66A3[AC030300]           <1>       mov    [u.quit], ax ; 1 ('crtl+brk' signal is enabled)
6026 0000E322 833D[BC030300]00         <1>         cmp  dword [u.ppgdir], 0  ; is the caller MainProg (kernel) ?
6027 0000E329 770C                     <1>       ja     short sysexec_18 ; no, the caller is user process
6028                                    <1>       ; If the caller is kernel (MainProg), 'sysexec' will come here
6029 0000E32B 8B15[00520100]           <1>       mov    edx, [k_page_dir] ; kernel's page directory
6030 0000E331 8915[BC030300]           <1>       mov    [u.ppgdir], edx ; next time 'sysexec' must not come here
6031                                    <1> sysexec_18:
6032                                    <1>       ; 02/05/2016
6033                                    <1>       ; 24/04/2016 (TRDOS 386 = TRDOS v2.0)
6034                                    <1>       ; 18/10/2015 (Retro UNIX 386 v1)
6035                                    <1>       ; 05/08/2015
6036                                    <1>       ; 29/07/2015
6037 0000E337 8B2D[4C040300]           <1>       mov    ebp, [argv] ; user's stack pointer must point to argument
6038                                    <1>                          ; list pointers (argument count)
6039 0000E33D FA                       <1>       cli
6040 0000E33E 8B25[9C510100]           <1>         mov     esp, [tss.esp0]  ; ring 0 (kernel) stack pointer
6041                                    <1>       ;mov         esp, [u.sp] ; Restore Kernel stack
6042                                    <1>                            ; for this process
6043                                    <1>       ;add   esp, 20 ; --> EIP, CS, EFLAGS, ESP, SS
6044                                    <1>       ;xor   eax, eax ; 0
6045 0000E344 FEC8                     <1>       dec    al ; eax = 0
6046 0000E346 66BA2300                 <1>       mov    dx, UDATA
6047 0000E34A 6652                     <1>       push   dx  ; user's stack segment
6048 0000E34C 55                       <1>       push   ebp ; user's stack pointer
6049                                    <1>                  ; (points to number of arguments)
6050                                    <1>
6051                                    <1>       ; 04/01/2017
6052                                    <1>       ; MainProg comes here while [sysflg]= 0FFh
6053                                    <1>       ; (but sysexec comes here while [sysflg]= 0)
6054 0000E34D C605[5B030300]00         <1>       mov    byte [sysflg], 0 ; 04/01/2017
6055                                    <1>                           ; (timer_int sysflg control)
6056 0000E354 FB                       <1>       sti
6057 0000E355 9C                       <1>       pushfd ; EFLAGS
6058                                    <1>              ; Set IF for enabling interrupts in user mode
6059                                    <1>       ;or    dword [esp], 200h
6060                                    <1>       ;
6061                                    <1>       ;mov   bx, UCODE
6062                                    <1>       ;push  bx ; user's code segment
6063 0000E356 6A1B                     <1>       push   UCODE
6064                                    <1>       ;push  0
6065 0000E358 50                       <1>       push   eax ; EIP (=0) - start address -
6066 0000E359 8925[5C030300]           <1>       mov    [u.sp], esp ; 29/07/2015
6067                                    <1>       ; 05/08/2015
6068                                    <1>       ; Remedy of a General Protection Fault during 'iretd' is here !
6069                                    <1>       ; ('push dx' would cause to general protection fault,
6070                                    <1>       ; after 'pop ds' etc.)
6071                                    <1>       ;
6072                                    <1>       ;; push dx ; ds (UDATA)
```

```
6073                                    <1>       ;; push dx ; es (UDATA)
6074                                    <1>       ;; push dx ; fs (UDATA)
6075                                    <1>       ;; push dx ; gs (UDATA)
6076                                    <1>       ;
6077                                    <1>       ; This is a trick to prevent general protection fault
6078                                    <1>       ; during 'iretd' intruction at the end of 'sysrele' (in u1.s):
6079 0000E35F 8EC2                     <1>       mov   es, dx ; UDATA
6080 0000E361 06                       <1>       push  es ; ds (UDATA)
6081 0000E362 06                       <1>       push  es ; es (UDATA)
6082 0000E363 06                       <1>       push  es ; fs (UDATA)
6083 0000E364 06                       <1>       push  es ; gs (UDATA)
6084 0000E365 66BA1000                 <1>       mov   dx, KDATA
6085 0000E369 8EC2                     <1>       mov   es, dx
6086                                    <1>       ;
6087                                    <1>       ;; pushad simulation
6088 0000E36B 89E5                     <1>       mov   ebp, esp ; esp before pushad
6089 0000E36D 50                       <1>       push  eax ; eax (0)
6090 0000E36E 50                       <1>       push  eax ; ecx (0)
6091 0000E36F 50                       <1>       push  eax ; edx (0)
6092 0000E370 50                       <1>       push  eax ; ebx (0)
6093 0000E371 55                       <1>       push  ebp ; esp before pushad
6094 0000E372 50                       <1>       push  eax ; ebp (0)
6095 0000E373 50                       <1>       push  eax ; esi (0)
6096 0000E374 50                       <1>       push  eax ; edi (0)
6097                                    <1>       ;
6098 0000E375 A3[64030300]             <1>       mov   [u.r0], eax ; eax = 0
6099 0000E37A 8925[60030300]           <1>       mov   [u.usp], esp
6100                                    <1>
6101                                    <1>       ; 02/05/2016
6102                                    <1>       ;inc   byte [sysflg] ; 0FFh -> 0
6103                                    <1>       ;mov   byte [sysflg], 0 ; 04/01/2017
6104 0000E380 0FB61D[B3030300]         <1>       movzx  ebx, byte [u.uno]
6105 0000E387 6683BB[3E000300]01       <1>       cmp   word [ebx+p.ppid-2], 1 ; MainProg
6106 0000E38F 0F8720E1FFFF             <1>       ja    sysret0 ; 03/05/2016
6107 0000E395 68[B3C40000]             <1>       push  sysret ; *
6108 0000E39A 8925[60030300]           <1>       mov   [u.usp], esp
6109 0000E3A0 E886030000               <1>       call  wswap ; save child process 'u' structure and
6110                                    <1>                   ; registers
6111 0000E3A5 8305[60030300]04         <1>       add   dword [u.usp], 4 ; 03/05/2016
6112                                    <1> sysexec_19: ; 02/05/2016
6113 0000E3AC C3                       <1>       retn ; * 'sysret' ; byte [sysflg] -> 0FFh
6114                                    <1>
6115                                    <1> readi:
6116                                    <1>       ; 01/05/2016
6117                                    <1>       ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)
6118                                    <1>       ; 20/05/2015 - Retro UNIX 386 v1
6119                                    <1>       ; 11/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
6120                                    <1>       ;
6121                                    <1>       ; Reads from a file whose the first cluster number in EAX
6122                                    <1>       ;
6123                                    <1>       ; INPUTS ->
6124                                    <1>       ;    EAX - First cluster number of the file
6125                                    <1>       ;    u.count - byte count user desires
6126                                    <1>       ;    u.base - points to user buffer
6127                                    <1>       ;    u.fofp - points to dword with current file offset
6128                                    <1>       ;    i.size - file size
6129                                    <1>       ;    cdev - logical dos drive number of the file
6130                                    <1>       ; OUTPUTS ->
6131                                    <1>       ;    u.count - cleared
6132                                    <1>       ;    u.nread - accumulates total bytes passed back
6133                                    <1>       ;
6134                                    <1>       ; ((EAX)) input/output
6135                                    <1>       ; (Retro UNIX Prototype : 14/12/2012 - 01/03/2013, UNIXCOPY.ASM)
6136                                    <1>         ; ((Modified registers: edx, ebx, ecx, esi, edi))
6137                                    <1>
6138 0000E3AD 31D2                     <1>       xor   edx, edx ; 0
6139 0000E3AF 8915[8C030300]           <1>       mov   [u.nread], edx ; 0
6140 0000E3B5 668915[C4030300]         <1>       mov   [u.pcount], dx ; 19/05/2015
6141 0000E3BC 3915[88030300]           <1>       cmp   [u.count], edx ; 0
6142 0000E3C2 7701                     <1>       ja    short readi_1
6143 0000E3C4 C3                       <1>       retn
6144                                    <1> readi_1:
6145                                    <1> dskr:
6146                                    <1>       ; 01/05/2016
6147                                    <1>       ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)
6148                                    <1>       ; 24/05/2015 - 12/10/2015 (Retro UNIX 386 v1)
6149                                    <1>       ; 26/04/2013 - 03/08/2013 (Retro UNIX 8086 v1)
6150                                    <1> dskr_0:
6151 0000E3C5 8B15[55040300]           <1>         mov  edx, [i.size]
6152 0000E3CB 8B1D[74030300]           <1>       mov   ebx, [u.fofp]
6153 0000E3D1 2B13                     <1>       sub   edx, [ebx]
6154 0000E3D3 7647                     <1>       jna   short dskr_4
6155                                    <1>       ;
6156 0000E3D5 50                       <1>       push  eax ; 01/05/2016
6157 0000E3D6 3B15[88030300]           <1>       cmp    edx, [u.count]
6158 0000E3DC 7306                     <1>       jnb   short dskr_1
6159 0000E3DE 8915[88030300]           <1>       mov   [u.count], edx
6160                                    <1> dskr_1:
6161                                    <1>       ; EAX = First Cluster
6162                                    <1>       ; [Current_Drv] = Physical drive number
6163 0000E3E4 E83B000000               <1>       call  mget_r
6164                                    <1>       ; NOTE: in 'mget_r', relevant sector will be read in buffer
6165                                    <1>       ; if it is not already in buffer !
6166 0000E3E9 BB[8C050300]             <1>       mov   ebx, readi_buffer
6167 0000E3EE 803D[C6030300]00         <1>       cmp   byte [u.kcall], 0 ; the caller is 'namei' sign (=1)
6168 0000E3F5 770F                     <1>       ja    short dskr_3   ; zf=0 -> the caller is 'namei'
6169 0000E3F7 66833D[C4030300]00       <1>       cmp   word [u.pcount], 0
6170 0000E3FF 7705                     <1>       ja    short dskr_3
6171                                    <1> dskr_2:
6172                                    <1>       ; [u.base] = virtual address to transfer (as destination address)
6173 0000E401 E894010000               <1>       call  trans_addr_w ; translate virtual address to physical (w)
6174                                    <1> dskr_3:
```

```
6175                                    <1>       ; EBX (r5) = system (I/O) buffer address -physical-
6176 0000E406 E8F7010000                <1>       call   sioreg
6177 0000E40B 87F7                      <1>       xchg   esi, edi
6178                                    <1>       ; EDI = file (user data) offset
6179                                    <1>       ; ESI = sector (I/O) buffer offset
6180                                    <1>       ; ECX = byte count
6181 0000E40D F3A4                      <1>       rep    movsb
6182                                    <1>       ; eax = remain bytes in buffer
6183                                    <1>         ;       (check if remain bytes in the buffer > [u.pcount])
6184 0000E40F 09C0                      <1>       or     eax, eax
6185 0000E411 75EE                      <1>       jnz    short dskr_2 ; (page end before system buffer end!)
6186 0000E413 58                        <1>       pop    eax  ; (first cluster number)
6187 0000E414 390D[88030300]            <1>       cmp    [u.count], ecx ; 0
6188 0000E41A 77A9                      <1>       ja     short dskr_0
6189                                    <1> dskr_4:
6190 0000E41C C605[C6030300]00          <1>       mov    byte [u.kcall], 0
6191 0000E423 C3                        <1>       retn
6192                                    <1>
6193                                    <1> mget_r:
6194                                    <1>       ; 24/10/2016
6195                                    <1>       ; 22/10/2016
6196                                    <1>       ; 12/10/2016
6197                                    <1>       ; 29/04/2016
6198                                    <1>       ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)
6199                                    <1>       ; 03/06/2015 (Retro UNIX 386 v1, 'mget', u.5s)
6200                                    <1>       ; 22/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
6201                                    <1>       ;
6202                                    <1>       ; Get existing or (allocate) a new disk block for file
6203                                    <1>       ;
6204                                    <1>       ; INPUTS ->
6205                                    <1>       ;    [u.fofp] = file offset pointer
6206                                    <1>       ;    EAX = First Cluster
6207                                    <1>       ;    [cdev] = Logical dos drive number
6208                                    <1>       ;    ([u.off] = file offset)
6209                                    <1>       ; OUTPUTS ->
6210                                    <1>       ;    EAX = logical sector number
6211                                    <1>       ;    ESI = Logical Dos Drive Description Table address
6212                                    <1>       ;
6213                                    <1>       ; Modified registers: EDX, EBX, ECX, ESI, EDI
6214                                    <1>
6215 0000E424 8B35[74030300]            <1>       mov     esi, [u.fofp]
6216 0000E42A 8B1E                      <1>       mov    ebx, [esi] ; (u.off)
6217                                    <1>
6218 0000E42C 29C9                      <1>       sub    ecx, ecx
6219 0000E42E 8A2D[46030300]            <1>       mov    ch, [cdev]
6220                                    <1>
6221 0000E434 BE00010900                <1>       mov    esi, Logical_DOSDisks
6222 0000E439 01CE                      <1>       add    esi, ecx
6223                                    <1>
6224 0000E43B 380D[385F0100]            <1>       cmp    [readi.valid], cl ; 0
6225 0000E441 7649                      <1>       jna    short mget_r_0
6226                                    <1>
6227 0000E443 3A2D[395F0100]            <1>       cmp    ch, [readi.drv]
6228 0000E449 7541                      <1>       jne    short mget_r_0
6229                                    <1>
6230 0000E44B 3B05[4C5F0100]            <1>       cmp    eax, [readi.fclust]
6231 0000E451 7565                      <1>       jne    short mget_r_3
6232                                    <1>
6233 0000E453 89D8                      <1>       mov    eax, ebx ; file offset
6234 0000E455 668B0D[405F0100]          <1>       mov    cx, [readi.bpc]
6235 0000E45C 41                        <1>       inc    ecx ; <= 65536
6236 0000E45D 29D2                      <1>       sub    edx, edx
6237 0000E45F F7F1                      <1>       div    ecx
6238                                    <1>
6239 0000E461 8B3D[485F0100]            <1>       mov    edi, [readi.c_index] ; cluster index
6240                                    <1>
6241 0000E467 39F8                      <1>       cmp    eax, edi
6242 0000E469 757A                      <1>         jne     short mget_r_4  ; (*)
6243                                    <1>
6244                                    <1>       ; edx = byte offset in cluster (<= 65535)
6245 0000E46B 668915[425F0100]          <1>       mov    [readi.offset], dx
6246 0000E472 66C1EA09                  <1>       shr    dx, 9 ; / 512
6247 0000E476 8815[3B5F0100]            <1>       mov    [readi.s_index], dl ; sector index in cluster (0 to spc -1)
6248                                    <1>
6249 0000E47C A1[445F0100]              <1>       mov    eax, [readi.cluster]  ; > 0 if [readi.valid] = 1
6250 0000E481 8B15[505F0100]            <1>       mov    edx, [readi.fs_index]
6251 0000E487 E99A000000                <1>         jmp     mget_r_7
6252                                    <1>
6253                                    <1> mget_r_0:
6254 0000E48C 882D[395F0100]            <1>       mov    [readi.drv], ch ; physical drive number
6255 0000E492 807E0300                  <1>       cmp    byte [esi+LD_FATType], 0
6256 0000E496 7707                      <1>       ja     short mget_r_1
6257 0000E498 8A4E12                    <1>       mov    cl, [esi+LD_FS_BytesPerSec+1]
6258 0000E49B D0E9                      <1>       shr    cl, 1 ;  ; 1 for 512 bytes, 4 for 2048 bytes
6259 0000E49D EB03                      <1>       jmp    short mget_r_2
6260                                    <1> mget_r_1:
6261 0000E49F 8A4E13                    <1>       mov    cl, [esi+LD_BPB+BPB_SecPerClust]
6262                                    <1> mget_r_2:
6263 0000E4A2 880D[3A5F0100]            <1>       mov    [readi.spc], cl  ; sectors per cluster
6264                                    <1>       ; NOTE: readi bytes per sector value is always 512 !
6265 0000E4A8 66C1E109                  <1>       shl    cx, 9 ; * 512
6266 0000E4AC 6649                      <1>       dec    cx ; bytes per cluster - 1
6267 0000E4AE 66890D[405F0100]          <1>       mov    [readi.bpc], cx
6268 0000E4B5 6629C9                    <1>       sub    cx, cx
6269                                    <1> mget_r_3:
6270 0000E4B8 A3[4C5F0100]              <1>       mov    [readi.fclust], eax ; first cluster (or FDT address)
6271 0000E4BD 880D[385F0100]            <1>       mov    [readi.valid], cl ; 0
6272                                    <1>       ;mov   [readi.s_index], cl ; 0
6273                                    <1>       ;mov   [readi.offset], cx ; 0
6274 0000E4C3 890D[485F0100]            <1>       mov    [readi.c_index], ecx ; 0
6275 0000E4C9 890D[445F0100]            <1>       mov    [readi.cluster], ecx ; 0
6276 0000E4CF 890D[3C5F0100]            <1>       mov    [readi.sector], ecx ; 0
```

```
6277                                <1>
6278 0000E4D5 89D8                 <1>       mov   eax, ebx ; file offset
6279 0000E4D7 668B0D[405F0100]     <1>       mov   cx, [readi.bpc]
6280 0000E4DE 41                   <1>       inc   ecx ; <= 65536
6281 0000E4DF 29D2                 <1>       sub   edx, edx
6282 0000E4E1 F7F1                 <1>       div   ecx
6283                               <1>       ;mov  edi, [readi.c_index] ; previous cluster index
6284 0000E4E3 29FF                 <1>       sub   edi, edi
6285                               <1> mget_r_4:
6286 0000E4E5 A3[485F0100]         <1>       mov   [readi.c_index], eax ; cluster index
6287                               <1>       ; edx = byte offset in cluster (<= 65535)
6288 0000E4EA 668915[425F0100]     <1>       mov   [readi.offset], dx
6289 0000E4F1 66C1EA09             <1>       shr   dx, 9 ; / 512
6290 0000E4F5 8815[3B5F0100]       <1>       mov   [readi.s_index], dl ; sector index in cluster (0 to spc -1)
6291                               <1>
6292 0000E4FB 89C1                 <1>       mov   ecx, eax ; current cluster index
6293 0000E4FD A1[4C5F0100]         <1>       mov   eax, [readi.fclust]
6294 0000E502 09C9                 <1>       or    ecx, ecx ; cluster index
6295 0000E504 741B                 <1>       jz    short mget_r_6
6296                               <1>
6297 0000E506 39CF                 <1>       cmp   edi, ecx
6298 0000E508 7710                 <1>       ja    short mget_r_5 ; old cluster index is higher
6299 0000E50A 8B15[445F0100]       <1>       mov   edx, [readi.cluster]
6300 0000E510 21D2                 <1>       and   edx, edx
6301 0000E512 7406                 <1>       jz    short mget_r_5
6302                               <1>       ; valid 'readi' parameters (*)
6303 0000E514 89D0                 <1>       mov   eax, edx
6304 0000E516 29F9                 <1>       sub   ecx, edi
6305 0000E518 740C                 <1>       jz    short mget_r_7
6306                               <1> mget_r_5:
6307                               <1>       ; EAX = Beginning cluster
6308                               <1>       ; EDX = Sector index in disk/file section
6309                               <1>       ;     (Only for SINGLIX file system!)
6310                               <1>       ; ECX = Cluster sequence number after the beginning cluster
6311                               <1>       ; ESI = Logical DOS Drive Description Table address
6312 0000E51A E80DDEFFFF           <1>       call  get_cluster_by_index
6313 0000E51F 724E                 <1>       jc    short mget_r_err
6314                               <1>       ; EAX = Cluster number
6315                               <1> mget_r_6:
6316 0000E521 A3[445F0100]         <1>       mov   [readi.cluster], eax ; FDT number for Singlix File System
6317                               <1> mget_r_7:
6318 0000E526 807E0300             <1>       cmp   byte [esi+LD_FATType], 0
6319 0000E52A 765F                 <1>       jna   short mget_r_12
6320                               <1>
6321 0000E52C 83E802               <1>       sub   eax, 2
6322 0000E52F 0FB615[3A5F0100]     <1>       movzx edx, byte [readi.spc]
6323 0000E536 F7E2                 <1>       mul   edx
6324                               <1>
6325 0000E538 034668               <1>       add   eax, [esi+LD_DATABegin]
6326 0000E53B 8A15[3B5F0100]       <1>       mov   dl, [readi.s_index]
6327 0000E541 01D0                 <1>       add   eax, edx
6328                               <1> mget_r_8:
6329                               <1>       ; eax = logical sector number
6330 0000E543 803D[385F0100]00     <1>       cmp   byte [readi.valid], 0
6331 0000E54A 7608                 <1>       jna   short mget_r_9
6332 0000E54C 3B05[3C5F0100]       <1>       cmp   eax, [readi.sector]
6333 0000E552 7436                 <1>       je    short mget_r_11 ; sector is already in 'readi' buffer
6334                               <1> mget_r_9:
6335 0000E554 A3[3C5F0100]         <1>       mov   [readi.sector], eax
6336 0000E559 BB[8C050300]         <1>       mov   ebx, readi_buffer ; buffer address
6337 0000E55E B901000000           <1>       mov   ecx, 1
6338                               <1>       ; 29/04/2016
6339                               <1>       ;xor  dl, dl
6340                               <1>
6341                               <1>       ; EAX = Logical sector number
6342                               <1>       ; ECX = Sector count
6343                               <1>       ; EBX = Buffer address
6344                               <1>       ; (EDX = 0)
6345                               <1>       ; ESI = Logical DOS drive description table address
6346                               <1>
6347 0000E563 E84A0C0000           <1>       call  disk_read
6348 0000E568 7314                 <1>       jnc   short mget_r_10
6349                               <1>
6350                               <1>       ; 22/10/2016 (15h -> 17)
6351 0000E56A B811000000           <1>       mov   eax, 17 ; Drive not ready or read error !
6352                               <1> mget_r_err:
6353 0000E56F A3[C8030300]         <1>       mov   [u.error], eax
6354                               <1>       ; 12/10/2016
6355 0000E574 A3[64030300]         <1>       mov   [u.r0], eax
6356 0000E579 E915DFFFFF           <1>       jmp   error
6357                               <1> mget_r_10:
6358 0000E57E C605[385F0100]01     <1>       mov   byte [readi.valid], 1 ; 24/10/2016
6359 0000E585 A1[3C5F0100]         <1>       mov   eax, [readi.sector]
6360                               <1> mget_r_11:
6361 0000E58A C3                   <1>       retn
6362                               <1> mget_r_12:
6363                               <1>       ; EAX = FDT number
6364                               <1>       ; EDX = Sector index from FDT sector (0,1,2,3,4...)
6365 0000E58B 40                   <1>       inc   eax ; the first data sector in FS disk section
6366 0000E58C 8915[505F0100]       <1>       mov   [readi.fs_index], edx
6367 0000E592 01D0                 <1>       add   eax, edx
6368 0000E594 EBAD                 <1>       jmp   short mget_r_8
6369                               <1>
6370                               <1> trans_addr_r:
6371                               <1>       ; 12/10/2016
6372                               <1>       ; 02/05/2016 - TRDOS 386 (TRDOS v2.0)
6373                               <1>       ; Translate virtual address to physical address
6374                               <1>       ; for reading from user's memory space
6375                               <1>       ; 04/06/2015 - 18/10/2015 (Retro UNIX 386 v1)
6376                               <1>
6377 0000E596 31D2                 <1>       xor   edx, edx ; 0 (read access sign)
6378 0000E598 EB04                 <1>       jmp   short trans_addr_rw
```

```
                                <1>
                                <1> trans_addr_w:
                                <1>       ; 12/10/2016
                                <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; Translate virtual address to physical address
                                <1>       ; for writing to user's memory space
                                <1>       ; 04/06/2015 - 18/10/2015 (Retro UNIX 386 v1)
                                <1>
0000E59A 29D2                   <1>       sub    edx, edx
0000E59C FEC2                   <1>       inc    dl ; 1 (write access sign)
                                <1> trans_addr_rw:
0000E59E 50                     <1>       push   eax
0000E59F 53                     <1>       push   ebx
0000E5A0 52                     <1>       push   edx ; r/w sign (in DL)
                                <1>       ;
0000E5A1 8B1D[84030300]         <1>       mov    ebx, [u.base]
0000E5A7 E8E26CFFFF             <1>       call   get_physical_addr ; get physical address
0000E5AC 730F                   <1>       jnc    short passc_0
0000E5AE A3[C8030300]           <1>       mov    [u.error], eax
0000E5B3 A3[64030300]           <1>       mov    [u.r0], eax ; 12/10/2016
                                <1>       ;pop   edx
                                <1>       ;pop   ebx
                                <1>       ;pop   eax
0000E5B8 E9D6DEFFFF             <1>       jmp    error
                                <1> passc_0:
0000E5BD F6C202                 <1>       test   dl, PTE_A_WRITE ; writable page
0000E5C0 5A                     <1>       pop    edx
0000E5C1 751C                   <1>       jnz    short passc_1
                                <1>
0000E5C3 20D2                   <1>       and    dl, dl
0000E5C5 7418                   <1>       jz     short passc_1
                                <1>       ; read only (duplicated) page -must be copied to a new page-
                                <1>       ; EBX = linear address
0000E5C7 51                     <1>       push   ecx
0000E5C8 E85A69FFFF             <1>       call   copy_page
0000E5CD 59                     <1>       pop    ecx
0000E5CE 721E                   <1>       jc     short passc_2
0000E5D0 50                     <1>       push   eax ; physical address of the new/allocated page
0000E5D1 E8E26BFFFF             <1>       call   add_to_swap_queue
0000E5D6 58                     <1>       pop    eax
0000E5D7 81E3FF0F0000           <1>       and    ebx, PAGE_OFF ; 0FFFh
                                <1>       ;mov   ecx, PAGE_SIZE
                                <1>       ;sub   ecx, ebx
0000E5DD 01D8                   <1>       add    eax, ebx
                                <1> passc_1:
0000E5DF A3[C0030300]           <1>       mov    [u.pbase], eax ; physical address
0000E5E4 66890D[C4030300]       <1>       mov    [u.pcount], cx ; remain byte count in page (1-4096)
0000E5EB 5B                     <1>       pop    ebx
0000E5EC 58                     <1>       pop    eax
0000E5ED C3                     <1>       retn
                                <1> passc_2:
0000E5EE B804000000             <1>       mov    eax, ERR_MINOR_IM ; "Insufficient memory !" error
0000E5F3 A3[64030300]           <1>       mov    [u.r0], eax ; 12/10/2016
0000E5F8 A3[C8030300]           <1>       mov    dword [u.error], eax
                                <1>       ;pop   ebx
                                <1>       ;pop   eax
0000E5FD E991DEFFFF             <1>       jmp    error
                                <1>
                                <1> sioreg:
                                <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; 19/05/2015 - 25/07/2015 (Retro UNIX 386 v1)
                                <1>       ; 12/03/2013 - 22/07/2013 (Retro UNIX 8086 v1)
                                <1>       ; INPUTS ->
                                <1>       ;     EBX = system buffer (data) address (r5)
                                <1>       ;     [u.fofp] = pointer to file offset pointer
                                <1>       ;     [u.base] = virtual address of the user buffer
                                <1>       ;     [u.pbase] = physical address of the user buffer
                                <1>       ;     [u.count] = byte count
                                <1>       ;     [u.pcount] = byte count within page frame
                                <1>       ; OUTPUTS ->
                                <1>       ;     ESI = user data offset (r1)
                                <1>       ;     EDI = system (I/O) buffer offset (r2)
                                <1>       ;     ECX = byte count (r3)
                                <1>       ;     EAX = remain bytes after byte count within page frame
                                <1>       ;      (If EAX > 0, transfer will continue from the next page)
                                <1>         ;
                                <1>       ; ((Modified registers:  EDX))
                                <1>
0000E602 8B35[74030300]         <1>         mov     esi, [u.fofp]
0000E608 8B3E                   <1>         mov     edi, [esi]
0000E60A 89F9                   <1>       mov    ecx, edi
0000E60C 81C900FEFFFF           <1>       or     ecx, 0FFFFFE00h
0000E612 81E7FF010000           <1>       and    edi, 1FFh
0000E618 01DF                   <1>       add    edi, ebx ; EBX = system buffer (data) address
0000E61A F7D9                   <1>       neg    ecx
0000E61C 3B0D[88030300]         <1>       cmp    ecx, [u.count]
0000E622 7606                   <1>       jna    short sioreg_0
0000E624 8B0D[88030300]         <1>       mov    ecx, [u.count]
                                <1> sioreg_0:
0000E62A 803D[C6030300]00       <1>       cmp    byte [u.kcall], 0
0000E631 7613                   <1>       jna    short sioreg_1
                                <1>        ; the caller is 'mkdir' or 'namei'
0000E633 A1[84030300]           <1>       mov    eax, [u.base]
0000E638 A3[C0030300]           <1>       mov    [u.pbase], eax ; physical address = virtual address
0000E63D 66890D[C4030300]       <1>       mov    word [u.pcount], cx ; remain bytes in buffer (1 sector)
0000E644 EB0B                   <1>       jmp    short sioreg_2
                                <1> sioreg_1:
0000E646 0FB715[C4030300]       <1>       movzx  edx, word [u.pcount]
0000E64D 39D1                   <1>       cmp    ecx, edx
0000E64F 772A                   <1>       ja     short sioreg_4 ; transfer count > [u.pcount]
                                <1> sioreg_2: ; 2:
0000E651 31C0                   <1>       xor    eax, eax
```

```
6481                                <1> sioreg_3:
6482 0000E653 010D[8C030300]       <1>         add    [u.nread], ecx
6483 0000E659 290D[88030300]       <1>         sub    [u.count], ecx
6484 0000E65F 010D[84030300]       <1>         add    [u.base], ecx
6485 0000E665 010E                 <1>           add  [esi], ecx
6486 0000E667 8B35[C0030300]       <1>         mov    esi, [u.pbase]
6487 0000E66D 66290D[C4030300]     <1>         sub    [u.pcount], cx
6488 0000E674 010D[C0030300]       <1>         add    [u.pbase], ecx
6489 0000E67A C3                   <1>           retn
6490                                <1> sioreg_4:
6491                                <1>         ; transfer count > [u.pcount]
6492                                <1>         ; (ecx > edx)
6493 0000E67B 89C8                 <1>         mov    eax, ecx
6494 0000E67D 29D0                 <1>         sub    eax, edx ; remain bytes for 1 sector (block) transfer
6495 0000E67F 89D1                 <1>         mov    ecx, edx ; current transfer count = [u.pcount]
6496 0000E681 EBD0                 <1>         jmp    short sioreg_3
6497                                <1>
6498                                <1> tswitch: ; Retro UNIX 386 v1
6499                                <1> tswap:
6500                                <1>         ; 16/01/2017
6501                                <1>         ; 21/05/2016 - TRDOS 386 (TRDOS v2.0)
6502                                <1>         ; 10/05/2015 - 01/09/2015 (Retro UNIX 386 v1)
6503                                <1>         ; 14/04/2013 - 14/02/2014 (Retro UNIX 8086 v1)
6504                                <1>         ; time out swap, called when a user times out.
6505                                <1>         ; the user is put on the low priority queue.
6506                                <1>         ; This is done by making a link from the last user
6507                                <1>         ; on the low priority queue to him via a call to 'putlu'.
6508                                <1>         ; then he is swapped out.
6509                                <1>
6510                                <1>         ; TRDOS 386 (TRDOS v2.0) modification ->  ** 21/05/2016 **
6511                                <1>         ;     * when a high priority (event) process will be stopped
6512                                <1>         ;     (swapped out, swithched out/off), 'tswap/tswitch' will
6513                                <1>         ;     not add it to a run queue.
6514                                <1>         ;     /// What for: Process may be already in a run queue,
6515                                <1>         ;     it is unspeficied state because process might be started
6516                                <1>         ;     by a timer event which does not regard previous priority
6517                                <1>         ;     level and run queue of the process (for fast executing!).
6518                                <1>         ;     After the 'run for event', process will be sequenced
6519                                <1>         ;     to run by it's actual run queue. ///
6520                                <1>         ;
6521                                <1>         ; Retro UNIX 386 v1 modification ->
6522                                <1>         ;       swap (software task switch) is performed by changing
6523                                <1>         ;     user's page directory (u.pgdir) instead of segment change
6524                                <1>         ;     as in Retro UNIX 8086 v1.
6525                                <1>         ;
6526                                <1>         ; RETRO UNIX 8086 v1 modification ->
6527                                <1>         ;       'swap to disk' is replaced with 'change running segment'
6528                                <1>         ;     according to 8086 cpu (x86 real mode) architecture.
6529                                <1>         ;     pdp-11 was using 64KB uniform memory while IBM PC
6530                                <1>         ;     compatibles was using 1MB segmented memory
6531                                <1>         ;     in 8086/8088 times.
6532                                <1>         ;
6533                                <1>         ; INPUTS ->
6534                                <1>         ;    u.uno - users process number
6535                                <1>         ;    runq+4 - lowest priority queue
6536                                <1>         ; OUTPUTS ->
6537                                <1>         ;    r0 - users process number
6538                                <1>         ;    r2 - lowest priority queue address
6539                                <1>         ;
6540                                <1>         ; ((AX = R0, BX = R2)) output
6541                                <1>         ; ((Modified registers: EDX, EBX, ECX, ESI, EDI))
6542                                <1>         ;
6543                                <1>
6544                                <1>         NOTE:
6545                                <1>         ;* [u.pri] priority level is specified by run queue which is process
6546                                <1>         ;  comes to run from.
6547                                <1>         ;* Initial [u.pri] is 1 ('normal/regular') for programs
6548                                <1>         ;  (which are launched by MainProg or 'sysexec'), it is changed
6549                                <1>         ;  to 2 ('high') by timer event, if program uses 'systimer' system call.
6550                                <1>         ;* Program (Process) also can change it's running priority
6551                                <1>         ;  from 1 to 0 or up to 2 by using 'syspri' system call; but,
6552                                <1>         ;  if program selects priority level 2 (high) for running, next time
6553                                <1>         ;  it is reduced to 1 (normal/regular) because 'syspri' adds this
6554                                <1>         ;  program to 'run for normal' queue while running duration is a bit
6555                                <1>         ;  protected from swap/switch out immediate, behalf of other high
6556                                <1>         ;  priority process in sequence. Program (with high priority) will not
6557                                <1>         ;  be swapped/switched out (by timer event) before it's time quantum
6558                                <1>         ;  will be elapsed, but, this will be temporary if program is not using
6559                                <1>         ;  timer event function.
6560                                <1>
6561                                <1>         ;For example:
6562                                <1>         ;If a process frequently gets a timer event, it runs at high priority
6563                                <1>         ;level but when it returns from running it returns to actual run queue,
6564                                <1>         ;not to 'run for event' queue again.
6565                                <1>         ;'tswap' will not change the sequence at return/stop(swap out) stage.
6566                                <1>         ;But if priority level not high (=2, 'run for event'), 'tswap/tswitch'
6567                                <1>         ;will add the stopping process to relevant run queue according to
6568                                <1>         ;[u.pri] priority level.
6569                                <1>
6570                                <1>         ; 16/01/2017
6571 0000E683 BB[54030300]         <1>         mov    ebx, runq+2  ; 'runq_normal' ; normal/regular priority
6572                                <1>         ; 21/05/2016
6573                                <1>         ;cmp   byte [u.pri], 2     ; high priority (run for event) ?
6574                                <1>         ;jnb   short swap
6575                                <1>         ; 16/01/2017
6576                                <1>         ; (Normal and also high/event priority processes will be added to
6577                                <1>         ; normal priority run queue for ensuring circular running sequence!)
6578                                <1>         ; (Timer interrupt or 'syspri' system call may change priority and run
6579                                <1>         ; queue to high/event level.)
6580 0000E688 803D[A9030300]00     <1>         cmp    byte [u.pri], 0
6581 0000E68F 7702                 <1>         ja     short tswap_1; normal priority run queue
6582                                <1>         ;
```

```
6583 0000E691 43                <1>       inc    ebx
6584 0000E692 43                <1>       inc    ebx          ; runq+4, 'runq_background', low priority
6585                            <1> tswap_1:
6586 0000E693 A0[B3030300]      <1>       mov    al, [u.uno]
6587                            <1>                    ; movb u.uno,r1 / move users process number to r1
6588                            <1>              ; mov  $runq+4,r2
6589                            <1>                    ; / move lowest priority queue address to r2
6590                            <1>              ; ebx = run queue
6591 0000E698 E8FE000000        <1>       call   putlu
6592                            <1>              ; jsr r0,putlu / create link from last user on Q to
6593                            <1>                          ; / u.uno's user
6594                            <1>
6595                            <1> switch: ; Retro UNIX 386 v1
6596                            <1> swap:
6597                            <1>       ; 02/01/2017
6598                            <1>       ; 21/05/2016
6599                            <1>       ; 20/05/2016
6600                            <1>       ; 02/05/2016
6601                            <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
6602                            <1>       ; 10/05/2015 - 02/09/2015 (Retro UNIX 386 v1)
6603                            <1>       ; 14/04/2013 - 08/03/2014 (Retro UNIX 8086 v1)
6604                            <1>       ;
6605                            <1>       ; 'swap' is routine that controls the swapping of processes
6606                            <1>       ; in and out of core.
6607                            <1>       ;
6608                            <1>       ; TRDOS 386 (TRDOS v2.0) modification ->  ** 20/05/2016 **
6609                            <1>       ;     * 3 different priority level is applied
6610                            <1>       ;     (just as original unix v1)
6611                            <1>       ;     1) high priority (event) run queue, 'runq_event'
6612                            <1>       ;     2) normal priority (regular) run queue, 'runq_normal'
6613                            <1>       ;     3) low priority (background) run queue, 'runq_backgroud'
6614                            <1>       ;     'swap' code will run a process which has max. priority
6615                            <1>       ;       (for earliest event at first)
6616                            <1>      ;
6617                            <1>      ; Retro UNIX 386 v1 modification ->
6618                            <1>      ;       swap (software task switch) is performed by changing
6619                            <1>      ;     user's page directory (u.pgdir) instead of segment change
6620                            <1>      ;     as in Retro UNIX 8086 v1.
6621                            <1>      ;
6622                            <1>      ; RETRO UNIX 8086 v1 modification ->
6623                            <1>      ;       'swap to disk' is replaced with 'change running segment'
6624                            <1>      ;     according to 8086 cpu (x86 real mode) architecture.
6625                            <1>      ;     pdp-11 was using 64KB uniform memory while IBM PC
6626                            <1>      ;     compatibles was using 1MB segmented memory
6627                            <1>      ;     in 8086/8088 times.
6628                            <1>      ;
6629                            <1>      ; INPUTS ->
6630                            <1>      ;    runq table - contains processes to run.
6631                            <1>      ;    p.link - contains next process in line to be run.
6632                            <1>      ;    u.uno - process number of process in core
6633                            <1>      ;    s.stack - swap stack used as an internal stack for swapping.
6634                            <1>      ; OUTPUTS ->
6635                            <1>      ;    (original unix v1 -> present process to its disk block)
6636                            <1>      ;    (original unix v1 -> new process into core ->
6637                            <1>      ;        Retro Unix 8086 v1 -> segment registers changed
6638                            <1>      ;        for new process)
6639                            <1>      ;    u.quant = 3 (Time quantum for a process)
6640                            <1>      ;     ((INT 1Ch count down speed -> 18.2 times per second)
6641                            <1>      ;    RETRO UNIX 8086 v1 will use INT 1Ch (18.2 times per second)
6642                            <1>      ;      for now, it will swap the process if there is not
6643                            <1>      ;      a keyboard event (keystroke) (Int 15h, function 4Fh)
6644                            <1>      ;      or will count down from 3 to 0 even if there is a
6645                            <1>      ;       keyboard event locking due to repetitive key strokes.
6646                            <1>      ;      u.quant will be reset to 3 for RETRO UNIX 8086 v1.
6647                            <1>      ;
6648                            <1>      ; ((Modified registers: EAX, EDX, EBX, ECX, ESI, EDI))
6649                            <1>
6650                            <1>      ;NOTE:
6651                            <1>      ;High priority queue is the first for selecting a process to run.
6652                            <1>      ;If there is not a process in high priority level run queue,
6653                            <1>      ;a process in normal priority run queue will be selected
6654                            <1>      ;or a proces in low priority run queue will be selected if normal
6655                            <1>      ;priority level run queue is empty.
6656                            <1>
6657                            <1>      ; 21/05/2016 -(3 priority levels, 3 run queues)
6658 0000E69D BE[52030300]      <1>      mov    esi, runq ; 'runq_event' ; high priority, 'run for event'
6659 0000E6A2 C605[945F0100]03 <1>      mov    byte [priority], 3 ; high priority + 1
6660 0000E6A9 31DB              <1>      xor    ebx, ebx ; 02/01/2017
6661                            <1> swap_0: ; 1: / search runq table for highest priority process
6662 0000E6AB 66AD              <1>      lodsw  ; mov ax, [esi], add esi+2
6663                            <1>      ;xor   ebx, ebx ; 02/05/2016
6664 0000E6AD 6621C0            <1>      and    ax, ax ; are there any processes to run in this Q entry
6665 0000E6B0 750E              <1>      jnz    short swap_2
6666                            <1>      ; 21/05/2026
6667                            <1>      ; runq_normal = runq+2, runq_background = runq+4
6668 0000E6B2 FE0D[945F0100]    <1>      dec    byte [priority] ; 3 -> 3, 2 -> 1, 1-> 0
6669 0000E6B8 75F1              <1>      jnz    short swap_0
6670                            <1>      ;cmp   esi, runq+6  ; if zero compare address to end of table
6671                            <1>      ;jb    short swap_0 ; if not at end, go back
6672                            <1> swap_1:
6673                            <1>      ; 02/05/2016
6674                            <1>      ; 29/04/2016 (TRDOS 386 = TRDOS v2.0)
6675                            <1>      ; No user process to run...
6676                            <1>      ; Run the kernel process... MainProg: Internal Command Interpreter
6677 0000E6BA FEC0              <1>      inc    al ; mov al, 1  ; process number of MainProg
6678 0000E6BC FEC3              <1>      inc    bl ; mov bl, al ; 1
6679 0000E6BE EB1E              <1>      jmp    short swap_4
6680                            <1> swap_2:
6681                            <1>      ; 21/05/2016
6682 0000E6C0 FE0D[945F0100]    <1>      dec    byte [priority] ; priority level of present user/process
6683                            <1>                             ; 0, 1, 2
6684 0000E6C6 4E                <1>      dec    esi
```

```
6685 0000E6C7 4E                <1>         dec     esi
6686                            <1>       ;
6687 0000E6C8 88C3              <1>       mov   bl, al
6688 0000E6CA 38E0              <1>       cmp   al, ah ; is there only 1 process in the queue to be run
6689 0000E6CC 740A              <1>       je    short swap_3 ; yes
6690 0000E6CE 8AA3[9F000300]    <1>       mov   ah, [ebx+p.link-1]
6691 0000E6D4 8826              <1>             mov   [esi], ah ; move next process in line into run queue
6692 0000E6D6 EB06              <1>       jmp   short swap_4
6693                            <1> swap_3:
6694 0000E6D8 6631D2            <1>       xor   dx, dx
6695 0000E6DB 668916            <1>       mov   [esi], dx ; zero the entry; no processes on the Q
6696                            <1> swap_4:
6697 0000E6DE 8A25[B3030300]    <1>       mov   ah, [u.uno]
6698 0000E6E4 38C4              <1>       cmp   ah, al ;is this process the same as the process in core?
6699 0000E6E6 743B              <1>             je    short swap_8 ; yes, don't have to swap
6700 0000E6E8 08E4              <1>       or    ah, ah  ; is the process # = 0
6701 0000E6EA 740D              <1>             jz    short swap_6 ; 'sysexit'
6702                            <1>       ;cmp  ah, al ;is this process the same as the process in core?
6703                            <1>             ;je   short swap_8 ; yes, don't have to swap
6704 0000E6EC 8925[60030300]    <1>       mov   [u.usp], esp ; return  address for 'syswait' & 'sleep'
6705 0000E6F2 E834000000        <1>       call  wswap   ; write out core to disk
6706 0000E6F7 EB1C              <1>       jmp   short swap_7
6707                            <1> swap_6:
6708                            <1>       ; Deallocate memory pages belong to the process
6709                            <1>       ; which is being terminated.
6710                            <1>       ; (Retro UNIX 386 v1 modification !)
6711                            <1>       ;
6712 0000E6F9 53                <1>       push  ebx
6713 0000E6FA A1[B8030300]      <1>       mov   eax, [u.pgdir]  ; page directory of the process
6714 0000E6FF 8B1D[BC030300]    <1>       mov   ebx, [u.ppgdir] ; page directory of the parent process
6715 0000E705 E8A865FFFF        <1>       call  deallocate_page_dir
6716 0000E70A A1[B4030300]      <1>       mov   eax, [u.upage] ; 'user' structure page of the process
6717 0000E70F E84366FFFF        <1>       call  deallocate_page
6718 0000E714 5B                <1>       pop   ebx
6719                            <1> swap_7:
6720 0000E715 C0E302            <1>       shl   bl, 2 ; * 4
6721 0000E718 8B83[BC000300]    <1>       mov   eax, [ebx+p.upage-4] ; the 'u' page of the new process
6722 0000E71E E840000000        <1>       call  rswap ; read new process into core
6723                            <1> swap_8:
6724                            <1>       ; Retro UNIX  8086 v1 modification !
6725 0000E723 C605[A8030300]04  <1>       mov   byte [u.quant], time_count
6726 0000E72A C3                <1>       retn
6727                            <1>
6728                            <1> wswap:  ; < swap out, swap to disk >
6729                            <1>       ; 28/02/2017 (fnsave)
6730                            <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
6731                            <1>       ; 09/05/2015 (Retro UNIX 386 v1)
6732                            <1>       ; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)
6733                            <1>       ; 'wswap' writes out the process that is in core onto its
6734                            <1>       ; appropriate disk area.
6735                            <1>       ;
6736                            <1>       ; Retro UNIX 386 v1 modification ->
6737                            <1>       ;       User (u) structure content and the user's register content
6738                            <1>       ;     will be copied to the process's/user's UPAGE (a page for
6739                            <1>       ;     saving 'u' structure and user registers for task switching).
6740                            <1>       ;     u.usp - points to kernel stack address which contains
6741                            <1>       ;             user's registers while entering system call.
6742                            <1>       ;     u.sp  - points to kernel stack address
6743                            <1>       ;             to return from system call -for IRET-.
6744                            <1>       ;     [u.usp]+32+16 = [u.sp]
6745                            <1>       ;     [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,
6746                            <1>       ;             edx, ecx, eax, gs, fs, es, ds, -> [u.sp].
6747                            <1>       ;
6748                            <1>       ; Retro UNIX 8086 v1 modification ->
6749                            <1>       ;       'swap to disk' is replaced with 'change running segment'
6750                            <1>       ;     according to 8086 cpu (x86 real mode) architecture.
6751                            <1>       ;     pdp-11 was using 64KB uniform memory while IBM PC
6752                            <1>       ;     compatibles was using 1MB segmented memory
6753                            <1>       ;     in 8086/8088 times.
6754                            <1>       ;
6755                            <1>       ; INPUTS ->
6756                            <1>       ;    u.break - points to end of program
6757                            <1>       ;    u.usp - stack pointer at the moment of swap
6758                            <1>       ;    core - beginning of process program
6759                            <1>       ;    ecore - end of core
6760                            <1>       ;    user - start of user parameter area
6761                            <1>       ;    u.uno - user process number
6762                            <1>       ;    p.dska - holds block number of process
6763                            <1>       ; OUTPUTS ->
6764                            <1>       ;    swp I/O queue
6765                            <1>       ;    p.break - negative word count of process
6766                            <1>       ;    r1 - process disk address
6767                            <1>       ;    r2 - negative word count
6768                            <1>       ;
6769                            <1>       ; RETRO UNIX 8086 v1 input/output:
6770                            <1>       ;
6771                            <1>       ; INPUTS ->
6772                            <1>       ;    u.uno - process number (to be swapped out)
6773                            <1>       ; OUTPUTS ->
6774                            <1>       ;    none
6775                            <1>       ;
6776                            <1>       ;   ((Modified registers: ECX, ESI, EDI))
6777                            <1>       ;
6778                            <1>
6779                            <1>       ; 28/02/2017
6780                            <1>       ;cmp   byte [multi_tasking], 0  ; Musti tasking mode ?
6781                            <1>       ;jna   short wswp
6782 0000E72B 803D[DA030300]00  <1>       cmp   byte [u.fpsave], 0 ; 28/02/2017
6783 0000E732 7606              <1>       jna   short wswp
6784 0000E734 DD35[DC030300]    <1>       fnsave [u.fpregs] ; save floating point registers (94 bytes)
6785                            <1> wswp:
6786 0000E73A 8B3D[B4030300]    <1>       mov   edi, [u.upage] ; process's user (u) structure page addr
```

```
6787 0000E740 B938000000          <1>       mov   ecx, (U_SIZE + 3) / 4
6788 0000E745 BE[5C030300]        <1>       mov   esi, user ; active user (u) structure
6789 0000E74A F3A5                <1>       rep   movsd
6790                              <1>       ;
6791 0000E74C 8B35[60030300]      <1>       mov   esi, [u.usp] ; esp (system stack pointer,
6792                              <1>                          ;     points to user registers)
6793 0000E752 8B0D[5C030300]      <1>       mov   ecx, [u.sp]  ; return address from the system call
6794                              <1>                          ; (for IRET)
6795                              <1>                          ; [u.sp] -> EIP (user)
6796                              <1>                          ; [u.sp+4]-> CS (user)
6797                              <1>                          ; [u.sp+8] -> EFLAGS (user)
6798                              <1>                          ; [u.sp+12] -> ESP (user)
6799                              <1>                          ; [u.sp+16] -> SS (user)
6800 0000E758 29F1                <1>       sub   ecx, esi     ; required space for user registers
6801 0000E75A 83C114              <1>       add   ecx, 20           ; +5 dwords to return from system call
6802                              <1>                          ; (for IRET)
6803 0000E75D C1E902              <1>       shr   ecx, 2
6804 0000E760 F3A5                <1>       rep   movsd
6805 0000E762 C3                  <1>       retn
6806                              <1>
6807                              <1> rswap:  ; < swap in, swap from disk >
6808                              <1>       ; 28/02/2017 (frstor)
6809                              <1>       ; 15/01/2017
6810                              <1>       ; 14/01/2017
6811                              <1>       ; 21/05/2016
6812                              <1>       ; 03/05/2016
6813                              <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
6814                              <1>       ; 09/05/2015 - 15/09/2015 (Retro UNIX 386 v1)
6815                              <1>       ; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)
6816                              <1>       ; 'rswap' reads a process whose number is in r1,
6817                              <1>       ; from disk into core.
6818                              <1>       ;
6819                              <1>       ; Retro UNIX 386 v1 modification ->
6820                              <1>       ;      User (u) structure content and the user's register content
6821                              <1>       ;     will be restored from process's/user's UPAGE (a page for
6822                              <1>       ;     saving 'u' structure and user registers for task switching).
6823                              <1>       ;     u.usp - points to kernel stack address which contains
6824                              <1>       ;             user's registers while entering system call.
6825                              <1>       ;     u.sp  - points to kernel stack address
6826                              <1>       ;             to return from system call -for IRET-.
6827                              <1>       ;     [u.usp]+32+16 = [u.sp]
6828                              <1>       ;     [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,
6829                              <1>       ;             edx, ecx, eax, gs, fs, es, ds, -> [u.sp].
6830                              <1>       ;
6831                              <1>       ; RETRO UNIX 8086 v1 modification ->
6832                              <1>       ;      'swap to disk' is replaced with 'change running segment'
6833                              <1>       ;     according to 8086 cpu (x86 real mode) architecture.
6834                              <1>       ;     pdp-11 was using 64KB uniform memory while IBM PC
6835                              <1>       ;     compatibles was using 1MB segmented memory
6836                              <1>       ;     in 8086/8088 times.
6837                              <1>       ;
6838                              <1>       ; INPUTS ->
6839                              <1>       ;    r1 - process number of process to be read in
6840                              <1>       ;    p.break - negative of word count of process
6841                              <1>       ;    p.dska - disk address of the process
6842                              <1>       ;    u.emt - determines handling of emt's
6843                              <1>       ;    u.ilgins - determines handling of illegal instructions
6844                              <1>       ; OUTPUTS ->
6845                              <1>       ;    8 = (u.ilgins)
6846                              <1>       ;    24 = (u.emt)
6847                              <1>       ;    swp - bit 10 is set to indicate read
6848                              <1>       ;            (bit 15=0 when reading is done)
6849                              <1>       ;    swp+2 - disk block address
6850                              <1>       ;    swp+4 - negative word count
6851                              <1>       ;      ((swp+6 - address of user structure))
6852                              <1>       ;
6853                              <1>       ; RETRO UNIX 8086 v1 input/output:
6854                              <1>       ;
6855                              <1>       ; INPUTS ->
6856                              <1>       ;    AL      - new process number (to be swapped in)
6857                              <1>       ; OUTPUTS ->
6858                              <1>       ;    none
6859                              <1>       ;
6860                              <1>       ;   ((Modified registers: EAX, ECX, ESI, EDI, ESP))
6861                              <1>       ;
6862                              <1>       ; Retro UNIX 386 v1 - modification ! 14/05/2015
6863 0000E763 89C6                <1>       mov   esi, eax  ; process's user (u) structure page addr
6864 0000E765 B938000000          <1>       mov   ecx, (U_SIZE + 3) / 4
6865 0000E76A BF[5C030300]        <1>       mov   edi, user ; active user (u) structure
6866 0000E76F F3A5                <1>       rep   movsd
6867 0000E771 58                  <1>       pop   eax    ; 'rswap' return address
6868                              <1>       ;
6869                              <1>       ;cli
6870 0000E772 8B3D[60030300]      <1>       mov   edi, [u.usp] ; esp (system stack pointer,
6871                              <1>                          ;     points to user registers)
6872 0000E778 89FC                <1>       mov   esp, edi     ; 14/01/2017
6873 0000E77A 8B0D[5C030300]      <1>       mov   ecx, [u.sp]  ; return address from the system call
6874                              <1>                          ; (for IRET)
6875                              <1>                          ; [u.sp] -> EIP (user)
6876                              <1>                          ; [u.sp+4]-> CS (user)
6877                              <1>                          ; [u.sp+8] -> EFLAGS (user)
6878                              <1>                          ; [u.sp+12] -> ESP (user)
6879                              <1>                          ; [u.sp+16] -> SS (user)
6880 0000E780 29F9                <1>       sub   ecx, edi     ; required space for user registers
6881 0000E782 83C114              <1>       add   ecx, 20           ; +5 dwords to return from system call
6882                              <1>                          ; (for IRET)
6883 0000E785 C1E902              <1>       shr   ecx, 2
6884 0000E788 F3A5                <1>       rep   movsd
6885                              <1>       ;mov  esp, [u.usp] ; 15/09/2015
6886                              <1>       ;sti
6887                              <1>       ; 28/02/2017
6888                              <1>       ;cmp  byte [multi_tasking], 0  ; Musti tasking mode ?
```

```
6889                                    <1>       ;jna   short rswp_retn
6890 0000E78A 803D[DA030300]00          <1>       cmp    byte [u.fpsave], 0
6891 0000E791 7606                      <1>       jna    short rswp_retn
6892 0000E793 DD25[DC030300]            <1>       frstor [u.fpregs] ; restore floating point regs (94 bytes)
6893                                    <1> rswp_retn:
6894 0000E799 50                        <1>       push   eax    ; 'rswap' return address
6895 0000E79A C3                        <1>       retn
6896                                    <1>
6897                                    <1> putlu:
6898                                    <1>       ; 20/05/2016
6899                                    <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
6900                                    <1>       ; 10/05/2015 - 12/09/2015 (Retro UNIX 386 v1)
6901                                    <1>       ; 15/04/2013 - 23/02/2014 (Retro UNIX 8086 v1)
6902                                    <1>       ; 'putlu' is called with a process number in r1 and a pointer
6903                                    <1>       ; to lowest priority Q (runq+4) in r2. A link is created from
6904                                    <1>       ; the last process on the queue to process in r1 by putting
6905                                    <1>       ; the process number in r1 into the last process's link.
6906                                    <1>       ;
6907                                    <1>       ; INPUTS ->
6908                                    <1>       ;    r1 - user process number
6909                                    <1>       ;    r2 - points to lowest priority queue
6910                                    <1>       ;    p.dska - disk address of the process
6911                                    <1>       ;    u.emt - determines handling of emt's
6912                                    <1>       ;    u.ilgins - determines handling of illegal instructions
6913                                    <1>       ; OUTPUTS ->
6914                                    <1>       ;    r3 - process number of last process on the queue upon
6915                                    <1>       ;        entering putlu
6916                                    <1>       ;    p.link-1 + r3 - process number in r1
6917                                    <1>       ;    r2 - points to lowest priority queue
6918                                    <1>       ;
6919                                    <1>       ; ((Modified registers: EDX, EBX))
6920                                    <1>       ;
6921                                    <1>       ; / r1 = user process no.; r2 points to lowest priority queue
6922                                    <1>
6923                                    <1>       ; EBX = r2
6924                                    <1>       ; EAX = r1 (AL=r1b)
6925                                    <1>
6926                                    <1>       ; 20/05/2016
6927                                    <1>       ; AL = process number (1 to 16) // Retro UNIX 8086, 386 v1 //
6928                                    <1>       ;     (max. 16 processes available for current kernel version)
6929                                    <1>       ; EBX = run queue address ; 20/05/2016 (TRDOS 386)
6930                                    <1>             ; which is one of following addresses:
6931                                    <1>             ;  1) 'runq_event' high priority run queue
6932                                    <1>             ;  2) 'runq_normal' normal/regular priority run queue
6933                                    <1>             ;  3) 'runq_background' low priority run queue
6934                                    <1>
6935                                    <1>       ;mov   ebx, runq
6936 0000E79B 0FB613                   <1>       movzx        edx, byte [ebx]
6937 0000E79E 43                       <1>       inc    ebx
6938 0000E79F 20D2                     <1>       and    dl, dl
6939                                    <1>             ; tstb (r2)+ / is queue empty?
6940 0000E7A1 740A                     <1>             jz     short putlu_1
6941                                    <1>             ; beq 1f / yes, branch
6942 0000E7A3 8A13                     <1>       mov    dl, [ebx] ; 12/09/2015
6943                                    <1>             ; movb (r2),r3 / no, save the "last user" process number
6944                                    <1>                        ; / in r3
6945 0000E7A5 8882[9F000300]           <1>             mov    [edx+p.link-1], al
6946                                    <1>             ; movb r1,p.link-1(r3) / put pointer to user on
6947                                    <1>                          ; / "last users" link
6948 0000E7AB EB03                     <1>       jmp    short putlu_2
6949                                    <1>             ; br 2f /
6950                                    <1> putlu_1: ; 1:
6951 0000E7AD 8843FF                   <1>       mov    [ebx-1], al
6952                                    <1>                    ; movb r1,-1(r2) / user is only user;
6953                                    <1>                        ; / put process no. at beginning and at end
6954                                    <1> putlu_2: ; 2:
6955 0000E7B0 8803                     <1>       mov    [ebx], al
6956                                    <1>                    ; movb r1,(r2) / user process in r1 is now the last entry
6957                                    <1>                        ; / on the queue
6958 0000E7B2 88C2                     <1>       mov    dl, al
6959 0000E7B4 88B2[9F000300]           <1>         mov     [edx+p.link-1], dh ; 0
6960                                    <1>             ; dec r2 / restore r2
6961 0000E7BA C3                       <1>         retn
6962                                    <1>             ; rts r0
6963                                    <1> sysver:
6964                                    <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
6965 0000E7BB C705[64030300]0002-       <1>       mov    dword [u.r0], 200h ; AH = major version, AL = minor version
6965 0000E7C3 0000                     <1>
6966 0000E7C5 E9E9DCFFFF               <1>       jmp    sysret
6967                                    <1>
6968                                    <1> sysreserved1:
6969                                    <1>       ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
6970                                    <1>       ; // name and content will be changed later //
6971 0000E7CA C705[64030300]E007-       <1>       mov    dword [u.r0], 2016
6971 0000E7D2 0000                     <1>
6972 0000E7D4 E9DADCFFFF               <1>       jmp    sysret
6973                                    <1>
6974                                    <1> syspri: ; change running priority (of the process)
6975                                    <1>       ; 21/05/2016
6976                                    <1>       ; 20/05/2026 - TRDOS 386 (TRDOS v2.0)
6977                                    <1>       ; INPUT ->
6978                                    <1>       ;     BL = priority level
6979                                    <1>       ;        0 = low running priority (running on background)
6980                                    <1>       ;        1 = normal/regular priority (running as regular)
6981                                    <1>       ;        2 = high/event priority (running for event)
6982                                    <1>       ;        >2 = invalid, it will accepted as 2 (event)
6983                                    <1>       ;        0FFh = get/return current running priority only
6984                                    <1>       ; OUTPUT ->
6985                                    <1>       ;     * if current [u.pri] < 2
6986                                    <1>       ;       if BL input < 0FFh ->
6987                                    <1>       ;          [u.pri] is updated as in BL input (0,1,2)
6988                                    <1>       ;       if BL input = 0FFh -> AL = [u.pri] (current)
```

```
                                <1>       ;
                                <1>       ;     * if current [u.pri] = 2
                                <1>       ;       if BL input < 0FFh -> cf = 1 & AL = 2
                                <1>       ;       if BL input = 0FFh -> cf = 0 & AL = 2
                                <1>       ;
                                <1>       ;     NOTE:
                                <1>       ;     If [u.pri] = 2, it can not be changed to 1 or 0;
                                <1>       ;     because, run queue of the running process is unspecified
                                <1>       ;     at this      stage. Process might be started by a timer event
                                <1>       ;     or priority might be changed to high by previous
                                <1>       ;     'syspri' system    call. In both cases, the process is in
                                <1>       ;     'runq_normal' or 'runq_background' queue.
                                <1>       ;     As result of this fact, when the [u.quant] time quantum
                                <1>       ;     of the process is elapsed or 'sysrele' system call is
                                <1>       ;     instructed by the process, 'tswap' ('tswitch') procedure
                                <1>       ;     will be called (to 'swap' or 'switch' out the procedure)
                                <1>       ;     and it will not call 'putlu' to add the (stopping)
                                <1>       ;     process to relevant run queue when [u.pri] = 2.
                                <1>       ;     (Otherwise, it would be possible to add process to
                                <1>       ;     a run queue while it is already in a run queue, wrongly.)
                                <1>       ;
                                <1>       ;     If [u.pri]< 2, 'tswap/tswitch' procedure will call
                                <1>       ;     'putlu' to add process to relevant run queue
                                <1>       ;     according to [u.pri] value. ('runq_normal' for 1,
                                <1>       ;     'runq_background' for 0).
                                <1>       ;
                                <1>       ;     If BL input >= 2 and < 0FFh while [u.pri] < 2,
                                <1>       ;     process will be added to 'runq_normal' queue and
                                <1>       ;     [u.pri] will be set to 2. (in 'syspri' system call)
                                <1>       ;
                                <1>
0000E7D9 29C0                   <1>       sub   eax, eax ; 0
0000E7DB A3[C8030300]           <1>       mov   [u.error], eax
                                <1>
0000E7E0 A0[A9030300]           <1>       mov   al, [u.pri]
0000E7E5 A3[64030300]           <1>       mov   [u.r0], eax
                                <1>
0000E7EA FEC3                   <1>       inc   bl
0000E7EC 0F84C1DCFFFF           <1>       jz    sysret ; 0FFh -> 0, get priority level
                                <1>
0000E7F2 3C02                   <1>       cmp   al, 2
0000E7F4 0F8399DCFFFF           <1>       jnb   error ; CF = 1 & AL = 2 (& last error = 0)
                                <1>
0000E7FA FECB                   <1>       dec   bl
0000E7FC 80FB02                 <1>       cmp   bl, 2
0000E7FF 7602                   <1>       jna   short syspri_1
0000E801 B302                   <1>       mov   bl, 2
                                <1> syspri_1:
0000E803 881D[A9030300]         <1>       mov   [u.pri], bl
0000E809 80FB02                 <1>       cmp   bl, 2
0000E80C 0F82A1DCFFFF           <1>         jb      sysret
                                <1>
                                <1>       ; here...
                                <1>       ; Priority of current process has been changed to high
                                <1>       ; ('run for event') but current process will be added to
                                <1>       ; 'run as normal' queue. ('run for event' high priority
                                <1>       ; queue is under control of timer -& RTC- interrupt only!)
                                <1>       ;
                                <1>       ; (Otherwise, process can fall into black hole!
                                <1>       ; e.g. if it is not in waiting list and it has not got
                                <1>       ; a timer event and it is not in a run queue!
                                <1>       ; Because, when [u.pri] is 2, 'tswap/tswitch' will not
                                <1>       ; add the stopping process to a run queue.)
                                <1>
0000E812 A0[B3030300]           <1>       mov   al, [u.uno]
0000E817 BB[54030300]           <1>       mov   ebx, runq_normal ; normal priority !
                                <1>                              ; [u.pri] is set to high
                                <1>                              ; but 'runq_event' queue is set
                                <1>                              ; only by the kernel's timer
                                <1>                              ; event function (timer interrupt).
0000E81C E87AFFFFFF             <1>       call  putlu
0000E821 E98DDCFFFF             <1>       jmp   sysret
                                <1>
                                <1> cpass: ; / get next character from user area of core and put it in AL (r1)
                                <1>       ; 02/05/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; 19/05/2015 - 18/10/2015 (Retro UNIX 386 v1)
                                <1>       ; 14/08/2013 - 20/09/2013 (Retro UNIX 8086 v1)
                                <1>       ; INPUTS ->
                                <1>       ;     [u.base] = virtual address in user area
                                <1>       ;     [u.count] = byte count (max.)
                                <1>       ;     [u.pcount] = byte count in page (0 = reset)
                                <1>       ; OUTPUTS ->
                                <1>       ;     AL = the character which is pointed by [u.base]
                                <1>       ;     zf = 1 -> transfer count has been completed
                                <1>         ;
                                <1>       ; ((Modified registers:  EAX, EDX, ECX))
                                <1>       ;
0000E826 833D[88030300]00       <1>       cmp   dword [u.count], 0  ; have all the characters been transferred
                                <1>                                 ; i.e., u.count, # of chars. left
0000E82D 763F                   <1>       jna   short cpass_3    ; to be transferred = 0?) yes, branch
0000E82F FF0D[88030300]         <1>       dec   dword [u.count]         ; no, decrement u.count
                                <1>         ; 19/05/2015
                                <1>       ;(Retro UNIX 386 v1 - translation from user's virtual address
                                <1>       ;                  to physical address
0000E835 66833D[C4030300]00     <1>       cmp   word [u.pcount], 0 ; byte count in page = 0 (initial value)
                                <1>                            ; 1-4095 --> use previous physical base address
                                <1>                            ; in [u.pbase]
0000E83D 770E                   <1>       ja    short cpass_1
0000E83F 833D[BC030300]00       <1>       cmp     dword [u.ppgdir], 0  ; is the caller os kernel
0000E846 7427                   <1>         je      short cpass_k       ; (sysexec, '/etc/init') ?  (MainProg)
0000E848 E849FDFFFF             <1>       call  trans_addr_r
                                <1> cpass_1:
```

```
7091 0000E84D 66FF0D[C4030300]    <1>       dec    word [u.pcount]
7092                              <1> cpass_2:
7093 0000E854 8B15[C0030300]      <1>       mov    edx, [u.pbase]
7094 0000E85A 8A02                <1>       mov    al, [edx]    ; take the character pointed to
7095                              <1>                           ; by u.base and put it in r1
7096 0000E85C FF05[8C030300]      <1>       inc    dword [u.nread] ; increment no. of bytes transferred
7097 0000E862 FF05[84030300]      <1>       inc    dword [u.base]  ; increment the buffer address to point to the
7098                              <1>                             ; next byte
7099 0000E868 FF05[C0030300]      <1>       inc    dword [u.pbase]
7100                              <1> cpass_3:
7101 0000E86E C3                  <1>       retn
7102                              <1> cpass_k:
7103                              <1>       ; 02/07/2015
7104                              <1>       ; The caller is os kernel
7105                              <1>       ; (get sysexec arguments from kernel's memory space)
7106 0000E86F 8B1D[84030300]      <1>       mov    ebx, [u.base]
7107 0000E875 66C705[C4030300]00- <1>         mov     word [u.pcount], PAGE_SIZE ; 4096
7107 0000E87D 10                  <1>
7108 0000E87E 891D[C0030300]      <1>       mov    [u.pbase], ebx
7109 0000E884 EBCE                <1>       jmp    short cpass_2
7110                              <1>
7111                              <1> transfer_to_user_buffer: ; fast transfer
7112                              <1>       ; 27/05/2016
7113                              <1>       ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
7114                              <1>       ;
7115                              <1>       ; INPUT ->
7116                              <1>       ;      ESI = source address in system space
7117                              <1>       ;      EDI = user's buffer address
7118                              <1>       ;      ECX = transfer (byte) count
7119                              <1>       ;      [u.pgdir] = user's page directory
7120                              <1>       ; OUTPUT ->
7121                              <1>       ;      ECX = actual transfer count
7122                              <1>       ;      cf = 1 -> error
7123                              <1>       ;      [u.count] = remain byte count
7124                              <1>       ;
7125                              <1>       ; Modified registers: eax, ecx
7126                              <1>       ;
7127                              <1>
7128 0000E886 21C9                <1>       and    ecx, ecx
7129 0000E888 743B                <1>       jz     short ttub_4
7130                              <1>
7131 0000E88A 890D[88030300]      <1>       mov    [u.count], ecx
7132                              <1>
7133 0000E890 57                  <1>       push   edi
7134 0000E891 56                  <1>       push   esi
7135 0000E892 53                  <1>       push   ebx
7136 0000E893 52                  <1>       push   edx
7137 0000E894 51                  <1>       push   ecx
7138                              <1>
7139 0000E895 89FB                <1>       mov    ebx, edi
7140 0000E897 81C300004000        <1>       add    ebx, CORE ; 27/05/2016
7141                              <1> ttub_1:
7142                              <1>       ; ebx = virtual (linear) address
7143                              <1>       ; [u.pgdir] = user's page directory
7144 0000E89D E8F269FFFF          <1>             call   get_physical_addr_x ; get physical address
7145 0000E8A2 7222                <1>       jc     short ttub_5
7146                              <1>       ; eax = physical address
7147                              <1>       ; ecx = remain byte count in page (1-4096)
7148 0000E8A4 89C7                <1>       mov    edi, eax
7149 0000E8A6 A1[88030300]        <1>       mov    eax, [u.count]
7150 0000E8AB 39C1                <1>       cmp    ecx, eax
7151 0000E8AD 7602                <1>       jna    short ttub_2
7152 0000E8AF 89C1                <1>       mov    ecx, eax
7153                              <1> ttub_2:
7154 0000E8B1 29C8                <1>       sub    eax, ecx
7155 0000E8B3 01CB                <1>       add    ebx, ecx
7156 0000E8B5 F3A4                <1>       rep    movsb
7157 0000E8B7 A3[88030300]        <1>       mov    [u.count], eax
7158 0000E8BC 09C0                <1>       or     eax, eax
7159 0000E8BE 75DD                <1>       jnz    short ttub_1
7160                              <1> ttub_retn:
7161                              <1> tfub_retn:
7162 0000E8C0 59                  <1>       pop    ecx ; transfer count = actual transfer count
7163                              <1> ttub_3:
7164 0000E8C1 5A                  <1>       pop    edx
7165 0000E8C2 5B                  <1>       pop    ebx
7166 0000E8C3 5E                  <1>       pop    esi
7167 0000E8C4 5F                  <1>       pop    edi
7168                              <1> ttub_4:
7169 0000E8C5 C3                  <1>       retn
7170                              <1> ttub_5:
7171 0000E8C6 59                  <1>       pop    ecx
7172 0000E8C7 2B0D[88030300]      <1>       sub    ecx, [u.count] ; actual transfer count
7173 0000E8CD F9                  <1>       stc
7174 0000E8CE EBF1                <1>       jmp    short ttub_3
7175                              <1>
7176                              <1> transfer_from_user_buffer: ; fast transfer
7177                              <1>       ; 27/05/2016
7178                              <1>       ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
7179                              <1>       ;
7180                              <1>       ; INPUT ->
7181                              <1>       ;      ESI = user's buffer address
7182                              <1>       ;      EDI = destination address in system space
7183                              <1>       ;      ECX = transfer (byte) count
7184                              <1>       ;      [u.pgdir] = user's page directory
7185                              <1>       ; OUTPUT ->
7186                              <1>       ;      ecx = actual transfer count
7187                              <1>       ;      cf = 1 -> error
7188                              <1>       ;      [u.count] = remain byte count
7189                              <1>       ;
7190                              <1>       ; Modified registers: eax, ecx
7191                              <1>       ;
```

```
7192                                    <1>
7193 0000E8D0 21C9                      <1>       and    ecx, ecx
7194                                    <1>       ;jz    short tfub_4
7195 0000E8D2 74F1                      <1>       jz     short ttub_4
7196                                    <1>
7197 0000E8D4 890D[88030300]            <1>       mov    [u.count], ecx
7198                                    <1>
7199 0000E8DA 57                        <1>       push   edi
7200 0000E8DB 56                        <1>       push   esi
7201 0000E8DC 53                        <1>       push   ebx
7202 0000E8DD 52                        <1>       push   edx
7203 0000E8DE 51                        <1>       push   ecx
7204                                    <1>
7205 0000E8DF 89F3                      <1>       mov    ebx, esi
7206 0000E8E1 81C300004000              <1>       add    ebx, CORE ; 27/05/2016
7207                                    <1> tfub_1:
7208                                    <1>       ; ebx = virtual (linear) address
7209                                    <1>       ; [u.pgdir] = user's page directory
7210 0000E8E7 E8A869FFFF                <1>             call   get_physical_addr_x ; get physical address
7211                                    <1>       ;jc    short tfub_5
7212 0000E8EC 72D8                      <1>       jc     short ttub_5
7213                                    <1>       ; eax = physical address
7214                                    <1>       ; ecx = remain byte count in page (1-4096)
7215 0000E8EE 89C6                      <1>       mov    esi, eax
7216 0000E8F0 A1[88030300]              <1>       mov    eax, [u.count]
7217 0000E8F5 39C1                      <1>       cmp    ecx, eax
7218 0000E8F7 7602                      <1>       jna    short tfub_2
7219 0000E8F9 89C1                      <1>       mov    ecx, eax
7220                                    <1> tfub_2:
7221 0000E8FB 29C8                      <1>       sub    eax, ecx
7222 0000E8FD 01CB                      <1>       add    ebx, ecx
7223 0000E8FF F3A4                      <1>       rep    movsb
7224 0000E901 A3[88030300]              <1>       mov    [u.count], eax
7225 0000E906 09C0                      <1>       or     eax, eax
7226 0000E908 75DD                      <1>       jnz    short tfub_1
7227                                    <1>
7228 0000E90A EBB4                      <1>       jmp    short tfub_retn
7229                                    <1>
7230                                    <1> ;tfub_retn:
7231                                    <1> ;     pop    ecx ; transfer count = actual transfer count
7232                                    <1> ;tfub_3:
7233                                    <1> ;     pop    edx
7234                                    <1> ;     pop    ebx
7235                                    <1> ;     pop    esi
7236                                    <1> ;     pop    edi
7237                                    <1> ;tfub_4:
7238                                    <1> ;     retn
7239                                    <1> ;tfub_5:
7240                                    <1> ;     pop    ecx
7241                                    <1> ;     sub    ecx, [u.count] ; actual transfer count
7242                                    <1> ;     stc
7243                                    <1> ;     jmp    short tfub_3
7244                                    <1>
7245                                    <1> sysfff: ; <Find First File>
7246                                    <1>       ; 17/10/2016
7247                                    <1>       ; 16/10/2016
7248                                    <1>       ; 15/10/2016 TRDOS 386 (TRDOS v2.0) feature only !
7249                                    <1>       ;           -derived from TRDOS v1.0, INT_21H.ASM-
7250                                    <1>       ;            ("loc_INT21h_find_first_file")
7251                                    <1>       ; TRDOS 8086 (v1.0)
7252                                    <1>         ;   07/08/2011
7253                                    <1>         ;   Find First File
7254                                    <1>       ;     INPUT:
7255                                    <1>         ;       CX= Attributes
7256                                    <1>         ;       DS:DX= Pointer to filename
7257                                    <1>       ;     MSDOS OUTPUT:
7258                                    <1>       ;         DTA: (Default address: PSP offset 80h)
7259                                    <1>       ;         Offset  Descrription
7260                                    <1>       ;         0     Reserved for use find next file
7261                                    <1>       ;         21    Attribute of file found
7262                                    <1>       ;         22    Time stamp of file
7263                                    <1>       ;         24    Date stamp of file
7264                                    <1>       ;         26    File size in bytes
7265                                    <1>       ;         30    Filename and extension (zero terminated)
7266                                    <1>       ;     If cf = 1:
7267                                    <1>       ;         Error Codes: (in AX)
7268                                    <1>       ;            2 - File not found
7269                                    <1>       ;            18 - No more files
7270                                    <1>         ;
7271                                    <1>       ; TRDOS 386 (v2.0)
7272                                    <1>       ; 15/10/2016
7273                                    <1>       ;
7274                                    <1>         ; INPUT ->
7275                                    <1>         ;       CL = File attributes
7276                                    <1>       ;            bit 0 (1) - Read only file (R)
7277                                    <1>       ;             bit 1 (1) - Hidden file (H)
7278                                    <1>         ;            bit 2 (1) - System file (R)
7279                                    <1>       ;             bit 3 (1) - Volume label/name (V)
7280                                    <1>         ;            bit 4 (1) - Subdirectory (D)
7281                                    <1>       ;            bit 5 (1) - File has been archived (A)
7282                                    <1>       ;         CH = 0 -> Return basic parameters (24 bytes)
7283                                    <1>       ;         CH > 0 -> Return FindFile structure/table (128 bytes)
7284                                    <1>         ;          EBX = Pointer to filename (ASCIIZ) -path-
7285                                    <1>       ;         EDX = File parameters buffer address
7286                                    <1>       ;            (buffer size = 24 bytes if CH input = 0)
7287                                    <1>       ;            (buffer size = 128 bytes if CH input > 0)
7288                                    <1>       ;
7289                                    <1>       ; OUTPUT ->
7290                                    <1>       ;         EAX = 0 if CH input > 0
7291                                    <1>       ;         EAX = First cluster number of file if CH input = 0
7292                                    <1>       ;         EDX = File parameters table/structure address
7293                                    <1>       ;         Basic Parameters:
```

```
                                <1>       ;            Offset  Description
                                <1>       ;            ------ ---------------
                                <1>       ;            0      File Attributes
                                <1>       ;            1      Ambiguous filename chars are used sign
                                <1>       ;                   (0 = filename fits exactly with request)
                                <1>       ;                   (>0 = ambiguous filename chars are used)
                                <1>       ;            2      Time stamp of file
                                <1>       ;            4      Date stamp of file
                                <1>       ;            6      File size in bytes
                                <1>       ;            10     Short Filename (ASCIIZ, max. 13 bytes)
                                <1>       ;            23     Longname Length (1-255) if existing
                                <1>       ;
                                <1>       ;          cf = 1 -> Error code in AL
                                <1>       ;
                                <1>       ; Modified Registers: EAX (at the return of system call)
                                <1>       ;
                                <1>       ; TR-DOS FindFile (FFF) Structure (128 bytes):
                                <1>       ; 09/10/2011 (DIR.ASM) - 10/02/2016 (trdoskx.s)
                                <1>       ;
                                <1>       ; Offset     Parameter           Size
                                <1>       ; ------     ------------------  --------
                                <1>       ; 0          FindFile_Drv        1 byte
                                <1>       ; 1          FindFile_Directory  65 bytes
                                <1>       ; 66         FindFile_Name       13 bytes
                                <1>       ; 79         FindFile_LongNameEntryLength 1 byte
                                <1>       ;Above 80 bytes form
                                <1>       ;TR-DOS Source/Destination File FullName Format/Structure
                                <1>       ; 80         FindFile_AttributesMask 1 word
                                <1>       ; 82         FindFile_DirEntry   32 bytes (*)
                                <1>       ; 114        FindFile_DirFirstCluster 1 double word
                                <1>       ; 118        FindFile_DirCluster 1 double word
                                <1>       ; 122        FindFile_DirEntryNumber 1 word
                                <1>       ; 124        FindFile_MatchCounter     1 word
                                <1>       ; 126        FindFile_Reserved   1 word
                                <1>       ; (*) MS-DOS, FAT 12-16-32 classic directory entry (32 bytes)
                                <1>
                                <1>       ;mov   [u.namep], ebx
                                <1>       ; 16/10/2016
0000E90C 8915[B45F0100]         <1>       mov    [FFF_UBuffer], edx
0000E912 66890D[B95F0100]       <1>       mov    [FFF_Attrib], cx ; [FFF_RType] = ch
                                <1>                   ; Attributes in CL, return data type in CH
0000E919 89DE                   <1>       mov    esi, ebx
                                <1>       ; file name is forced, change directory as temporary
                                <1>       ;mov   ax, 1
                                <1>       ;mov   [FFF_Valid], ah ; 0 ; reset ; 17/10/2016
                                <1>       ;call  set_working_path
0000E91B E8B50C0000             <1>       call   set_working_path_x ; 17/10/2016
0000E920 731D                   <1>       jnc    short sysfff_0
                                <1>
0000E922 21C0                   <1>       and    eax, eax  ; 0 -> Bad Path!
0000E924 7505                   <1>       jnz    short sysfff_err
                                <1>
                                <1>       ; eax = 0
0000E926 B80C000000             <1>       mov    eax, ERR_DIR_NOT_FOUND ; Directory not found !
                                <1> sysfff_err:
0000E92B A3[64030300]           <1>       mov    [u.r0], eax
0000E930 A3[C8030300]           <1>       mov    [u.error], eax
0000E935 E8700D0000             <1>         call       reset_working_path
0000E93A E954DBFFFF             <1>       jmp    error
                                <1>
                                <1> sysfff_0:
                                <1>       ;sub   ah, ah ; ah = 0
0000E93F 8A0424                 <1>       mov    al, [esp]
0000E942 08C0                   <1>       or     al, al
0000E944 7412                   <1>       jz     short sysfff_2
0000E946 B410                   <1>       mov    ah, 10h
0000E948 A808                   <1>       test   al, 08h
0000E94A 7503                   <1>       jnz    short sysfff_1
0000E94C 80CC08                 <1>       or     ah, 08h
                                <1> sysfff_1:
0000E94F 2410                   <1>       and    al, 10h ; Directory
0000E951 7405                   <1>       jz     short sysfff_2
0000E953 80E408                 <1>       and    ah, 08h
0000E956 30C0                   <1>       xor    al, al ; When a directory is searched,
                                <1>                     ; filename will be returned even if
                                <1>                     ; it is not a directory!
                                <1>                     ; Because: (in order to prevent
                                <1>                     ; creating a dir with existing file name)
                                <1>                     ; Dir and file names must not be same!
                                <1>                     ; (return attribute must be checked)
                                <1> sysfff_2:
                                <1>       ; AX = Attributes mask
                                <1>             ; AL = AND mask (result must be equal to AL)
                                <1>             ; AH = Negative AND mask (result must be ZERO)
                                <1>       ; ESI = FindFile_Name address
                                <1>
0000E958 E89296FFFF             <1>       call   find_first_file
0000E95D 72CC                   <1>       jc     short sysfff_err ; eax = 2 (File not found !)
                                <1>
                                <1>       ; ESI = Directory Entry (FindFile_DirEntry) Location
                                <1>       ; EDI = Directory Buffer Directory Entry Location
                                <1>       ; EAX = File Size
                                <1>       ;  BL = Attributes of The File/Directory
                                <1>       ;  BH = Long Name Yes/No Status (>0 is YES)
                                <1>       ;  DX > 0 : Ambiguous filename chars are used
                                <1>
                                <1> sysfff_3:
                                <1>       ; 16/10/2016
0000E95F 668B0D[B95F0100]       <1>       mov    cx, [FFF_Attrib]
                                <1>       ; Attribs in CL, return data type in CH
                                <1>
                                <1>       ;or    cl, cl
```

```
7396                                    <1>       ;jz    short sysfff_4 ; 0 = No filter
7397 0000E966 80F1FF                    <1>       xor    cl, 0FFh
7398 0000E969 20D9                      <1>       and    cl, bl
7399 0000E96B 7409                      <1>       jz     short sysfff_4
7400                                    <1>
7401                                    <1>       ;mov   eax, 2 ; 'file not found !' error
7402                                    <1>       ;jmp   short sysfff_err_1
7403                                    <1>
7404                                    <1>       ; 16/10/2016
7405 0000E96D E82C97FFFF                <1>       call   find_next_file
7406 0000E972 72B7                      <1>       jc     short sysfff_err ; eax = 12 (no more files !)
7407 0000E974 EBE9                      <1>       jmp    short sysfff_3
7408                                    <1>
7409                                    <1> sysfff_4:
7410 0000E976 20ED                      <1>       and    ch, ch ; [FFF_RType]
7411 0000E978 7412                      <1>       jz     short sysfff_5
7412 0000E97A B980000000                <1>       mov    ecx, 128 ; ; transfer length
7413 0000E97F 880D[B85F0100]            <1>       mov    [FFF_Valid], cl
7414                                    <1> sysfnf_11:
7415 0000E985 BE[6A5C0100]              <1>       mov    esi, FindFile_Drv
7416 0000E98A EB44                      <1>       jmp    short sysfff_6
7417                                    <1> sysfff_5:
7418                                    <1>       ;mov   esi, FindFile_DirEntry
7419 0000E98C B918000000                <1>       mov    ecx, 24  ; transfer length
7420 0000E991 880D[B85F0100]            <1>       mov    [FFF_Valid], cl
7421                                    <1> sysfnf_12:
7422 0000E997 BF[74640100]              <1>       mov    edi, DTA ; FFF data transfer address
7423                                    <1>       ;mov   al, [esi+DirEntry_Attr] ; 11
7424 0000E99C 88D8                      <1>       mov    al, bl ; File/Dir Attributes
7425 0000E99E 887F17                    <1>       mov    [edi+23], bh ; Longname length (0= none)
7426 0000E9A1 AA                        <1>       stosb
7427 0000E9A2 88D0                      <1>       mov    al, dl ; DL is for '?'
7428 0000E9A4 00F0                      <1>       add    al, dh ; DH is for '*'
7429                                    <1>       ; AL > 0 if ambiguous file name wildcards are used
7430 0000E9A6 AA                        <1>       stosb
7431 0000E9A7 8B4616                    <1>       mov    eax, [esi+DirEntry_WrtTime] ; 22
7432 0000E9AA AB                        <1>         stosd      ; DirEntry_WrtTime & DirEntry_WrtDate
7433 0000E9AB 8B461C                    <1>         mov  eax, [esi+DirEntry_FileSize] ; 28
7434 0000E9AE AB                        <1>         stosd
7435 0000E9AF 668B4614                  <1>       mov    ax, [esi+DirEntry_FstClusHI] ; 20
7436 0000E9B3 66C1E010                  <1>       shl    ax, 16
7437 0000E9B7 668B461A                  <1>       mov    ax, [esi+DirEntry_FstClusLO] ; 26
7438 0000E9BB A3[64030300]              <1>       mov    [u.r0], eax ; First Cluster
7439                                    <1>
7440                                    <1>         ;mov esi, FindFile_DirEntry
7441 0000E9C0 E8220D0000                <1>       call   get_file_name
7442                                    <1>
7443 0000E9C5 8A0D[B85F0100]            <1>       mov    cl, [FFF_Valid]
7444 0000E9CB BE[74640100]              <1>              mov    esi, DTA ; FFF data transfer address
7445                                    <1> sysfff_6:
7446 0000E9D0 8B3D[B45F0100]            <1>       mov    edi, [FFF_UBuffer] ; user's buffer address (edx)
7447 0000E9D6 E8ABFEFFFF                <1>       call   transfer_to_user_buffer
7448                                    <1>
7449 0000E9DB 890D[64030300]            <1>       mov    [u.r0], ecx ; actual transfer count
7450 0000E9E1 E8C40C0000                <1>         call       reset_working_path
7451 0000E9E6 E9C8DAFFFF                <1>       jmp    sysret
7452                                    <1>
7453                                    <1> sysfnf: ; <Find Next File>
7454                                    <1>       ; 16/10/2016 TRDOS 386 (TRDOS v2.0) feature only !
7455                                    <1>       ;           -derived from TRDOS v1.0, INT_21H.ASM-
7456                                    <1>       ;            ("loc_INT21h_find_next_file")
7457                                    <1>       ; TRDOS 8086 (v1.0)
7458                                    <1>         ;   07/08/2011
7459                                    <1>         ;   Find First File
7460                                    <1>       ;     INPUT:
7461                                    <1>         ;       none
7462                                    <1>       ;     MSDOS OUTPUT:
7463                                    <1>       ;         DTA: (Default address: PSP offset 80h)
7464                                    <1>       ;         Offset  Descrription
7465                                    <1>       ;         0     Reserved for use find next file
7466                                    <1>       ;         21    Attribute of file found
7467                                    <1>       ;         22    Time stamp of file
7468                                    <1>       ;         24    Date stamp of file
7469                                    <1>       ;         26    File size in bytes
7470                                    <1>       ;         30    Filename and extension (zero terminated)
7471                                    <1>       ;     If cf = 1:
7472                                    <1>       ;         Error Codes: (in AX)
7473                                    <1>       ;            18 - No more files
7474                                    <1>         ;
7475                                    <1>       ; TRDOS 386 (v2.0)
7476                                    <1>       ; 16/10/2016
7477                                    <1>       ;
7478                                    <1>         ; INPUT ->
7479                                    <1>             ;         none
7480                                    <1>       ; OUTPUT ->
7481                                    <1>       ;        EAX = 0 if CH input of 'Find First File' > 0
7482                                    <1>       ;        EAX = First cluster number of file
7483                                    <1>       ;             if CH input of 'Find First File' = 0
7484                                    <1>       ;        EDX = File parameters table/structure address
7485                                    <1>       ;
7486                                    <1>       ;         cf = 1 -> Error code in AL
7487                                    <1>       ;
7488                                    <1>       ; Modified Registers: EAX (at the return of system call)
7489                                    <1>
7490                                    <1>       ;
7491                                    <1>       ; Note: If byte [FFF_Valid] = 0
7492                                    <1>       ;     'sysfnf' will return with 'no more files' error.
7493                                    <1>       ;     If byte [FFF_Valid] = 24
7494                                    <1>       ;     'sysfnf' will return with 32 bytes basic parameters
7495                                    <1>       ;     at the address which is in EDX.
7496                                    <1>       ;     If byte [FFF_Valid] = 128
7497                                    <1>       ;     'sysfnf' will return with 128 bytes Find File
```

```
7498                                <1>       ;     Structure/Table at the address which is in EDX.
7499                                <1>
7500 0000E9EB 803D[B85F0100]00      <1>       cmp   byte [FFF_Valid], 0
7501 0000E9F2 7714                  <1>       ja    short stsfnf_0
7502                                <1>       ; 'no more files !' error
7503 0000E9F4 B80C000000            <1>       mov   eax, ERR_NO_MORE_FILES ; 12
7504 0000E9F9 A3[64030300]          <1>       mov   [u.r0], eax
7505 0000E9FE A3[C8030300]          <1>       mov   [u.error], eax
7506 0000EA03 E98BDAFFFF            <1>       jmp   error
7507                                <1> stsfnf_0:
7508                                <1>       ;cmp  byte [FFF_Valid], 128
7509                                <1>       ;je   short stsfnf_1
7510                                <1>       ;cmp  byte [FFF_Valid], 24
7511                                <1>       ;je   short stsfnf_1
7512                                <1>       ;mov  [FFF_Valid], 24 ; Default
7513                                <1> stsfnf_1:
7514 0000EA08 0FB61D[C6520100]      <1>       movzx ebx, byte [Current_Drv]
7515 0000EA0F 66891D[BE5F0100]      <1>       mov   [SWP_DRV], bx
7516 0000EA16 8A15[6A5C0100]        <1>       mov   dl, [FindFile_Drv]
7517 0000EA1C 38DA                  <1>       cmp   dl, bl
7518 0000EA1E 750B                  <1>       jne   short stsfnf_2
7519 0000EA20 86FB                  <1>       xchg  bh, bl
7520 0000EA22 BE00010900            <1>       mov   esi, Logical_DOSDisks
7521 0000EA27 01DE                  <1>       add   esi, ebx
7522 0000EA29 EB0D                  <1>       jmp   short sysfnf_3
7523                                <1>
7524                                <1> stsfnf_2:
7525 0000EA2B FE05[BF5F0100]        <1>       inc   byte [SWP_DRV_chg]
7526                                <1>
7527 0000EA31 E82782FFFF            <1>       call  change_current_drive
7528 0000EA36 7245                  <1>       jc    short sysfnf_err_1 ; read error !
7529                                <1>                               ; (do not stop, because
7530                                <1>                               ; we don't have a
7531                                <1>                               ; 'no more files'
7532                                <1>                               ; -file not found- error,
7533                                <1>                               ; next sysfnf system call
7534                                <1>                               ; may solve the problem,
7535                                <1>                               ; after re-placing the disk)
7536                                <1> sysfnf_3:
7537 0000EA38 A1[E05C0100]          <1>       mov   eax, [FindFile_DirCluster]
7538 0000EA3D 21C0                  <1>       and   eax, eax
7539 0000EA3F 7550                  <1>       jnz   short sysfnf_6
7540                                <1>
7541 0000EA41 803D[C5520100]02      <1>       cmp   byte [Current_FATType], 2
7542 0000EA48 772C                  <1>       ja    short sysfnf_err_0 ; invalid, we neeed to stop !?
7543 0000EA4A 803D[C5520100]01      <1>       cmp   byte [Current_FATType], 1
7544 0000EA51 7223                  <1>       jb    short sysfnf_err_0 ; invalid, we neeed to stop !?
7545                                <1>
7546 0000EA53 3805[F05A0100]        <1>       cmp   byte [DirBuff_ValidData], al ; 0
7547 0000EA59 7608                  <1>       jna   short sysfnf_4
7548                                <1>
7549 0000EA5B 3B05[F55A0100]        <1>       cmp   eax, [DirBuff_Cluster] ; 0 ?
7550 0000EA61 745E                  <1>       je    short sysfnf_9
7551                                <1>
7552                                <1>       ;cmp  byte [Current_Dir_Level], 0
7553                                <1>         ;ja  short sysfnf_4
7554                                <1>         ;jna short sysfnf_9
7555                                <1>
7556                                <1> sysfnf_4:
7557 0000EA63 FE05[BF5F0100]        <1>       inc   byte [SWP_DRV_chg]
7558 0000EA69 E80AD0FFFF            <1>       call  load_FAT_root_directory
7559 0000EA6E 7351                  <1>       jnc   short sysfnf_9
7560                                <1>       ; eax = error code (17, 'drv not ready or read error')
7561 0000EA70 EB0B                  <1>       jmp   short sysfnf_err_1 ; read error ! (no FNF stop)
7562                                <1>                               ; (if you want, try again,
7563                                <1>                               ;  after re-placing the disk)
7564                                <1> sysfnf_5:
7565 0000EA72 3C0C                  <1>       cmp   al, 12 ; 'no more files' error
7566 0000EA74 7507                  <1>       jne   short sysfnf_err_1 ; (no FNF stop -sysfnf will try
7567                                <1>                               ;  to read the directory again,
7568                                <1>                               ;  if the user calls sysfnf
7569                                <1>                               ;  just after this error return-)
7570                                <1>       ; (FNF stop -sysfnf will not try
7571                                <1>       ;  to read the directory again-)
7572                                <1>
7573                                <1> sysfnf_err_0:
7574 0000EA76 C605[B85F0100]00      <1>       mov   byte [FFF_Valid], 0 ; FNF stop sign
7575                                <1> sysfnf_err_1:
7576 0000EA7D A3[64030300]          <1>       mov   [u.r0], eax
7577 0000EA82 A3[C8030300]          <1>       mov   [u.error], eax
7578 0000EA87 E81E0C0000            <1>       call  reset_working_path
7579 0000EA8C E902DAFFFF            <1>       jmp   error
7580                                <1>
7581                                <1> sysfnf_6:
7582 0000EA91 803D[F05A0100]00      <1>       cmp   byte [DirBuff_ValidData], 0
7583 0000EA98 7608                  <1>       jna   short sysfnf_7
7584                                <1>
7585 0000EA9A 3B05[F55A0100]        <1>       cmp   eax, [DirBuff_Cluster]
7586 0000EAA0 741F                  <1>       je    short sysfnf_9
7587                                <1>
7588                                <1> sysfnf_7:
7589 0000EAA2 FE05[BF5F0100]        <1>       inc   byte [SWP_DRV_chg]
7590 0000EAA8 803D[C5520100]01      <1>       cmp   byte [Current_FATType], 1
7591 0000EAAF 7309                  <1>       jnb   short sysfnf_8
7592                                <1>
7593                                <1>       ; Singlix (TRFS) File System
7594                                <1>       ; (access via compatibility buffer)
7595 0000EAB1 E88AD0FFFF            <1>       call  load_FS_sub_directory
7596 0000EAB6 7309                  <1>       jnc   short sysfnf_9
7597                                <1>
7598 0000EAB8 EBC3                  <1>       jmp   short sysfnf_err_1 ; read error (no FNF stop)
7599                                <1>
```

```
7600                                    <1> sysfnf_8:
7601 0000EABA E844D0FFFF                <1>         call    load_FAT_sub_directory
7602 0000EABF 72BC                      <1>         jc      short sysfnf_err_1 ; read error (no FNF stop)
7603                                    <1>
7604                                    <1> sysfnf_9:
7605 0000EAC1 E8D895FFFF                <1>         call    find_next_file
7606 0000EAC6 72AA                      <1>         jc      short sysfnf_5
7607                                    <1>
7608 0000EAC8 A0[B95F0100]              <1>         mov     al, [FFF_Attrib]
7609                                    <1>         ;or     al, al
7610                                    <1>         ;jz     short sysfnf_10 ; 0 = No filter
7611 0000EACD 34FF                      <1>         xor     al, 0FFh
7612 0000EACF 20D8                      <1>         and     al, bl
7613 0000EAD1 75EE                      <1>         jnz     short sysfnf_9 ; search for next file until
7614                                    <1>                                ; an error return from
7615                                    <1>                                ; find_next_file procedure
7616                                    <1> sysfnf_10:
7617 0000EAD3 0FB60D[B85F0100]          <1>           movzx       ecx, byte [FFF_Valid]
7618 0000EADA 80F980                    <1>         cmp     cl, 128 ; complete FindFile structure/table
7619 0000EADD 0F84A2FEFFFF              <1>         je      sysfnf_11
7620                                    <1>         ;cmp    cl, 24  ; basic parameters
7621                                    <1>         ;je     sysfnf_12
7622 0000EAE3 E9AFFEFFFF                <1>         jmp     sysfnf_12
7623                                    <1>
7624                                    <1> writei:
7625                                    <1>         ; 26/10/2016
7626                                    <1>         ; 25/10/2016
7627                                    <1>         ; 23/10/2016
7628                                    <1>         ; 22/10/2016
7629                                    <1>         ; 19/10/2016 - TRDOS 386 (TRDOS v2.0)
7630                                    <1>         ; 19/05/2015 - 20/05/2015 (Retro UNIX 386 v1)
7631                                    <1>         ; 12/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
7632                                    <1>         ;
7633                                    <1>         ; Write data to file with first cluster number in EAX
7634                                    <1>         ;
7635                                    <1>         ; INPUTS ->
7636                                    <1>         ;    EAX - First cluster number of the file
7637                                    <1>         ;    EBX - File number (Open file index number)
7638                                    <1>         ;    u.count - byte count to be written
7639                                    <1>         ;    u.base - points to user buffer
7640                                    <1>         ;    u.fofp - points to dword with current file offset
7641                                    <1>         ;    i.size - file size
7642                                    <1>         ;    cdev - logical dos drive number of the file
7643                                    <1>         ; OUTPUTS ->
7644                                    <1>         ;    u.count - cleared
7645                                    <1>         ;    u.nread - accumulates total bytes passed back
7646                                    <1>         ;    i.size - new file size (if file byte offset overs file size)
7647                                    <1>         ;    u.fofp - points to u.off (with new offset value)
7648                                    <1>         ;
7649                                    <1>         ; (Retro UNIX Prototype : 11/11/2012 - 18/11/2012, UNIXCOPY.ASM)
7650                                    <1>         ; ((Modified registers: eax, edx, ebx, ecx, esi, edi, ebp))
7651                                    <1>
7652 0000EAE8 31C9                      <1>         xor     ecx, ecx
7653 0000EAEA 890D[8C030300]            <1>         mov     [u.nread], ecx  ; 0
7654 0000EAF0 66890D[C4030300]          <1>         mov     [u.pcount], cx ; 19/05/2015
7655 0000EAF7 390D[88030300]            <1>         cmp     [u.count], ecx
7656 0000EAFD 7701                      <1>         ja      short writei_1
7657 0000EAFF C3                        <1>         retn
7658                                    <1> writei_1:
7659 0000EB00 881D[785F0100]            <1>         mov     [writei.ofn], bl ; Open file number
7660 0000EB06 880D[B35F0100]            <1>         mov     [setfmod], cl ; 0 ; reset 'update lm date&time' sign
7661                                    <1> dskw_0:
7662                                    <1>         ; 26/10/2016
7663                                    <1>         ; 22/10/2016, 23/10/2016, 25/10/2016
7664                                    <1>         ; 19/10/2016 - TRDOS 386 (TRDOS v2.0)
7665                                    <1>         ; 31/05/2015 - 25/07/2015 (Retro UNIX 386 v1)
7666                                    <1>         ; 26/04/2013 - 20/09/2013 (Retro UNIX 8086 v1)
7667                                    <1>         ;
7668                                    <1>         ; 01/08/2013 (mkdir_w check)
7669 0000EB0C E8D7000000                <1>         call    mget_w
7670                                    <1>         ; eax = sector/block number
7671                                    <1>
7672 0000EB11 8B1D[74030300]            <1>         mov     ebx, [u.fofp]
7673 0000EB17 8B13                      <1>         mov     edx, [ebx]
7674 0000EB19 81E2FF010000              <1>         and     edx, 1FFh  ; / test the lower 9 bits of the file offset
7675 0000EB1F 750C                      <1>         jnz     short dskw_1 ; / if its non-zero, branch
7676                                    <1>                              ; if zero, file offset = 0,
7677                                    <1>                                   ; / 512, 1024,...(i.e., start of new block)
7678 0000EB21 813D[88030300]0002-       <1>         cmp     dword [u.count], 512
7678 0000EB29 0000                      <1>
7679                                    <1>                                ; / if zero, is there enough data to fill
7680                                    <1>                                ; / an entire block? (i.e., no. of
7681 0000EB2B 7337                      <1>         jnb     short dskw_2 ; / bytes to be written greater than 512.?
7682                                    <1>                              ; / Yes, branch. Don't have to read block
7683                                    <1> dskw_1: ; in as no past info. is to be saved
7684                                    <1>         ; (the entire block will be overwritten).
7685                                    <1>         ; 23/10/2016
7686                                    <1>
7687 0000EB2D BB[94070300]              <1>         mov     ebx, writei_buffer
7688                                    <1>         ; esi = logical dos drive description table address
7689                                    <1>         ; eax = sector number
7690                                    <1>         ; ebx = buffer address (in kernel's memory space)
7691                                    <1>         ; ecx = sector count
7692 0000EB32 B901000000                <1>         mov     ecx, 1
7693 0000EB37 E876060000                <1>         call    disk_read
7694                                    <1>         ;call   dskrd  ; / no, must retain old info..
7695                                    <1>                                ; / Hence, read block 'r1' into an I/O buffer
7696 0000EB3C 7326                      <1>         jnc     short dskw_2
7697                                    <1>
7698                                    <1>         ; disk read error
7699 0000EB3E B811000000                <1>         mov     eax, 17 ; drive not ready or READ ERROR !
7700                                    <1> dskw_err: ; jump from disk write error
```

```
7701 0000EB43 A3[64030300]        <1>       mov    [u.r0], eax
7702 0000EB48 A3[C8030300]        <1>       mov    [u.error], eax
7703                              <1>
7704 0000EB4D 803D[B35F0100]00    <1>       cmp    byte [setfmod], 0
7705 0000EB54 0F8639D9FFFF        <1>       jna    error
7706                              <1>
7707 0000EB5A E8AF030000          <1>       call   update_file_lmdt ; update last modif. date&time of the file
7708                              <1>       ;mov   byte [setfmod], 0
7709                              <1>
7710 0000EB5F E92FD9FFFF          <1>       jmp    error
7711                              <1>
7712                              <1> dskw_2: ; 3:
7713                              <1>       ; 23/10/2016
7714 0000EB64 C605[545F0100]01    <1>       mov    byte [writei.valid], 1 ; writei buffer contains valid data
7715 0000EB6B 56                  <1>       push   esi ; logical dos drive description table address
7716                              <1>       ; EAX (r1) = block/sector number
7717                              <1>       ;call  wslot
7718                              <1>              ; jsr r0,wslot / set write and inhibit bits in I/O queue,
7719                              <1>                     ; / proc. status=0, r5 points to 1st word of data
7720 0000EB6C 803D[C6030300]00    <1>       cmp    byte [u.kcall], 0
7721 0000EB73 770F                <1>       ja     short dskw_4 ; zf=0 -> the caller is 'mkdir'
7722                              <1>       ;
7723 0000EB75 66833D[C4030300]00  <1>       cmp    word [u.pcount], 0
7724 0000EB7D 7705                <1>       ja     short dskw_4
7725                              <1> dskw_3:
7726                              <1>       ; [u.base] = virtual address to transfer (as source address)
7727 0000EB7F E812FAFFFF          <1>       call   trans_addr_r ; translate virtual address to physical (r)
7728                              <1> dskw_4:
7729 0000EB84 BB[94070300]        <1>       mov    ebx, writei_buffer
7730                              <1>       ; EBX (r5) = system (I/O) buffer address
7731 0000EB89 E874FAFFFF          <1>       call   sioreg
7732                              <1>       ; ESI = file (user data) offset
7733                              <1>       ; EDI = sector (I/O) buffer offset
7734                              <1>       ; ECX = byte count
7735                              <1>       ;
7736 0000EB8E F3A4                <1>       rep    movsb
7737                              <1>       ; 25/07/2015
7738                              <1>       ; eax = remain bytes in buffer
7739                              <1>         ;       (check if remain bytes in the buffer > [u.pcount])
7740 0000EB90 09C0                <1>       or     eax, eax
7741 0000EB92 75EB                <1>       jnz    short dskw_3 ; (page end before system buffer end!)
7742                              <1>
7743                              <1>       ; 23/10/2016
7744 0000EB94 B101                <1>       mov    cl, 1
7745 0000EB96 5E                  <1>       pop    esi
7746 0000EB97 A1[585F0100]        <1>       mov    eax, [writei.sector]
7747                              <1>       ; esi = logical dos drive description table address
7748                              <1>       ; eax = sector number
7749                              <1>       ; ebx = writei buffer address
7750                              <1>       ; ecx = sector count
7751 0000EB9C E802060000          <1>       call   disk_write ; / yes, write the block
7752 0000EBA1 7307                <1>       jnc    short dskw_5
7753                              <1>
7754 0000EBA3 B812000000          <1>       mov    eax, 18 ; drive not ready or WRITE ERROR !
7755 0000EBA8 EB99                <1>       jmp    short dskw_err
7756                              <1>
7757                              <1> dskw_5:
7758                              <1>       ; 26/10/2016
7759 0000EBAA 0FB61D[785F0100]    <1>       movzx  ebx, byte [writei.ofn] ; open file number
7760 0000EBB1 C0E302              <1>       shl    bl, 2 ; *4
7761 0000EBB4 8B83[48630100]      <1>       mov    eax, [ebx+OF_POINTER]
7762 0000EBBA 3B83[70630100]      <1>       cmp    eax, [ebx+OF_SIZE]
7763 0000EBC0 7606                <1>       jna    short dskw_6
7764 0000EBC2 8983[70630100]      <1>       mov    [ebx+OF_SIZE], eax
7765                              <1> dskw_6:
7766                              <1>       ;shr   bl, 2
7767 0000EBC8 833D[88030300]00    <1>         cmp     dword [u.count], 0 ; / any more data to write?
7768 0000EBCF 760A                <1>       jna    short dskw_7
7769 0000EBD1 A1[685F0100]        <1>       mov    eax, [writei.fclust]
7770 0000EBD6 E931FFFFFF          <1>       jmp    dskw_0 ; / yes, branch
7771                              <1> dskw_7:
7772                              <1>       ; update last modif. date&time of the file
7773                              <1>       ; (also updates file size as OF_SIZE)
7774 0000EBDB E82E030000          <1>       call   update_file_lmdt
7775                              <1>       ;mov   byte [setfmod], 0
7776                              <1>
7777                              <1>       ; 03/08/2013
7778 0000EBE0 C605[C6030300]00    <1>       mov    byte [u.kcall], 0
7779                              <1>       ; 23/10/2016
7780                              <1>       ;mov   eax, [writei.fclust]
7781 0000EBE7 C3                  <1>       retn
7782                              <1>
7783                              <1> mget_w:
7784                              <1>       ; 02/11/2016
7785                              <1>       ; 01/11/2016
7786                              <1>       ; 23/10/2016, 31/10/2016
7787                              <1>       ; 22/10/2016 - TRDOS 386 (TRDOS v2.0)
7788                              <1>       ; 03/06/2015 (Retro UNIX 386 v1, 'mget', u.5s)
7789                              <1>       ; 22/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
7790                              <1>       ;
7791                              <1>       ; Get existing or (allocate) a new disk block for file
7792                              <1>       ;
7793                              <1>       ; INPUTS ->
7794                              <1>       ;    [u.fofp] = file offset pointer
7795                              <1>       ;    [i.size] = file size
7796                              <1>       ;    [u.count] = byte count
7797                              <1>       ;    EAX = First cluster
7798                              <1>       ;    [cdev] = Logical dos drive number
7799                              <1>       ;    [writei.ofn] = File Number
7800                              <1>       ;                (Open file index, 0 based)
7801                              <1>       ;    ([u.off] = file offset)
7802                              <1>       ; OUTPUTS ->
```

```
7803                            <1>      ;    EAX = logical sector number
7804                            <1>      ;    ESI = Logical Dos Drive Description Table address
7805                            <1>      ;
7806                            <1>      ; Modified registers: EDX, EBX, ECX, ESI, EDI, EBP
7807                            <1>
7808 0000EBE8 8B35[74030300]    <1>        mov     esi, [u.fofp]
7809 0000EBEE 8B2E              <1>      mov    ebp, [esi] ; u.off (or EBX*4+OF_POINTER)
7810                            <1>
7811 0000EBF0 29C9              <1>      sub    ecx, ecx
7812 0000EBF2 8A2D[46030300]    <1>      mov    ch, [cdev]
7813                            <1>
7814 0000EBF8 BE00010900        <1>      mov    esi, Logical_DOSDisks
7815 0000EBFD 01CE              <1>      add    esi, ecx
7816                            <1>
7817                            <1>      ; 31/10/2016
7818 0000EBFF 89C3              <1>      mov    ebx, eax ; First Cluster or FDT address
7819                            <1>
7820 0000EC01 807E0300          <1>      cmp    byte [esi+LD_FATType], 0
7821 0000EC05 0F86DD010000      <1>      jna    mget_w_14 ; Singlix FS
7822                            <1>
7823 0000EC0B 0FB74611          <1>      movzx  eax, word [esi+LD_BPB+BytesPerSec]
7824 0000EC0F 0FB65613          <1>      movzx  edx, byte [esi+LD_BPB+SecPerClust]
7825 0000EC13 8815[565F0100]    <1>      mov    [writei.spc], dl  ; sectors per cluster
7826 0000EC19 F7E2              <1>      mul    edx
7827                            <1>      ; edx = 0
7828                            <1>      ; eax = bytes per cluster (<= 65536)
7829                            <1>
7830                            <1>      ; 02/11/2016
7831 0000EC1B 89C1              <1>      mov    ecx, eax
7832 0000EC1D 48                <1>      dec    eax
7833 0000EC1E 66A3[5C5F0100]    <1>      mov    [writei.bpc], ax
7834                            <1>
7835 0000EC24 89E8              <1>      mov    eax, ebp
7836 0000EC26 0305[88030300]    <1>      add    eax, [u.count] ; next file position
7837 0000EC2C 3B05[55040300]    <1>      cmp    eax, [i.size] ; <= file size ?
7838 0000EC32 0F86FC000000      <1>      jna    mget_w_4 ; no
7839                            <1>
7840 0000EC38 F7F1              <1>      div    ecx
7841 0000EC3A A3[645F0100]      <1>      mov    [writei.c_index], eax ; cluster index
7842                            <1>      ; edx = byte offset in cluster (<= 65535)
7843                            <1>      ;mov   [writei.offset], dx
7844                            <1>      ;shr   dx, 9 ; / 512
7845                            <1>      ;mov   [writei.s_index], dl ; sector index in cluster (0 to spc -1)
7846                            <1>
7847 0000EC3F 29D2              <1>      sub    edx, edx ; 01/11/2016
7848 0000EC41 8915[585F0100]    <1>      mov    [writei.sector], edx ; 0
7849 0000EC47 668915[5E5F0100]  <1>      mov    [writei.offset], dx  ; byte offset in cluster
7850 0000EC4E 8815[575F0100]    <1>      mov    [writei.s_index], dl ; sector index in cluster (0 to spc -1)
7851                            <1>
7852 0000EC54 89D8              <1>      mov    eax, ebx ; First Cluster
7853                            <1>
7854                            <1>      ; is this the 1st mget_w or a next mget_w call ? (by 'writei')
7855 0000EC56 3815[545F0100]    <1>      cmp    byte [writei.valid], dl ; 0
7856 0000EC5C 7624              <1>      jna    short mget_w_0
7857                            <1>
7858 0000EC5E 8815[545F0100]    <1>      mov    byte [writei.valid], dl ; 0 ; reset ('writei' will set it)
7859                            <1>
7860 0000EC64 3B05[685F0100]    <1>      cmp    eax, [writei.fclust]
7861 0000EC6A 7516              <1>      jne    short mget_w_0
7862                            <1>
7863 0000EC6C 8A0D[46030300]    <1>      mov    cl, [cdev]
7864 0000EC72 3A0D[555F0100]    <1>      cmp    cl, [writei.drv]
7865 0000EC78 7508              <1>      jne    short mget_w_0
7866                            <1>      ; [writei.l_clust] & [writei.l_index] are valid,
7867                            <1>      ;  we don't need to get last cluster & last cluster index
7868 0000EC7A 8B0D[745F0100]    <1>      mov    ecx, [writei.l_index]
7869 0000EC80 EB64              <1>      jmp    short mget_w_2
7870                            <1> mget_w_0:
7871 0000EC82 A3[685F0100]      <1>      mov    [writei.fclust], eax ; first cluster
7872                            <1>      ; edx = 0
7873 0000EC87 A3[605F0100]      <1>      mov    [writei.cluster], eax ; first cluster ; 01/11/2016
7874 0000EC8C 8915[6C5F0100]    <1>      mov    [writei.fs_index], edx ; 0 ; curret cluster index
7875                            <1>
7876                            <1>      ; FAT file system (FAT12, FAT16, FAT32)
7877 0000EC92 E881D4FFFF        <1>      call   get_last_cluster
7878 0000EC97 0F822B010000      <1>      jc     mget_w_err ; eax = error code
7879                            <1>
7880 0000EC9D A3[705F0100]      <1>      mov    [writei.lclust], eax ; last cluster
7881                            <1>
7882 0000ECA2 8B0D[945D0100]    <1>      mov    ecx, [glc_index] ; last cluster index
7883 0000ECA8 890D[745F0100]    <1>      mov    [writei.l_index], ecx
7884                            <1>
7885 0000ECAE A0[785F0100]      <1>      mov    al, [writei.ofn]
7886 0000ECB3 FEC0              <1>      inc    al
7887 0000ECB5 A2[B35F0100]      <1>      mov    [setfmod], al ; update lm date&time sign
7888                            <1>
7889                            <1> mget_w_1:
7890 0000ECBA 3B0D[645F0100]    <1>      cmp    ecx, [writei.c_index]  ; last cluster index
7891 0000ECC0 7324              <1>      jnb    short mget_w_2 ; 01/11/2016
7892                            <1>
7893 0000ECC2 A1[705F0100]      <1>      mov    eax, [writei.lclust]
7894                            <1>      ; EAX = Last cluster
7895 0000ECC7 E85AD5FFFF        <1>      call   add_new_cluster
7896 0000ECCC 0F82F6000000      <1>      jc     mget_w_err ; eax = error code
7897                            <1>      ; edx = 0
7898 0000ECD2 A3[705F0100]      <1>      mov    [writei.lclust], eax ; (new) last cluster
7899 0000ECD7 8B0D[745F0100]    <1>      mov    ecx, [writei.l_index]
7900 0000ECDD 41                <1>      inc    ecx ; add 1 to last cluster index
7901 0000ECDE 890D[745F0100]    <1>      mov    [writei.l_index], ecx ; current last cluster index
7902                            <1>
7903 0000ECE4 EBD4              <1>      jmp    short mget_w_1
7904                            <1>
```

```
7905                                <1> mget_w_2:
7906 0000ECE6 89E9                  <1>         mov    ecx, ebp
7907 0000ECE8 030D[88030300]        <1>         add    ecx, [u.count]
7908 0000ECEE 890D[55040300]        <1>         mov    [i.size], ecx ; save new file size
7909                                <1>         ;sub   edx, edx ; 0
7910                                <1>
7911 0000ECF4 A0[46030300]          <1>         mov    al, [cdev]
7912 0000ECF9 A2[555F0100]          <1>         mov    [writei.drv], al ; physical drive number
7913                                <1>         ; edx = 0
7914 0000ECFE 89E8                  <1>         mov    eax, ebp ; file offset
7915 0000ED00 0FB70D[5C5F0100]      <1>         movzx  ecx, word [writei.bpc] ; bytes per cluster - 1
7916 0000ED07 41                    <1>         inc    ecx ; bytes per cluster
7917 0000ED08 F7F1                  <1>         div    ecx
7918                                <1>         ; edx = byte offset in cluster (<= 65535)
7919                                <1>         ; eax = cluster index
7920 0000ED0A A3[645F0100]          <1>         mov    [writei.c_index], eax
7921 0000ED0F 668915[5E5F0100]      <1>         mov    [writei.offset], dx
7922 0000ED16 66C1EA09              <1>         shr    dx, 9 ; / 512
7923 0000ED1A 8815[575F0100]        <1>         mov    [writei.s_index], dl ; sector index in cluster (0 to spc -1)
7924                                <1>
7925                                <1> mget_w_3:
7926 0000ED20 3B05[745F0100]        <1>         cmp    eax, [writei.l_index] ; last cluster index
7927 0000ED26 752A                  <1>         jne    short mget_w_5
7928                                <1>
7929 0000ED28 A3[6C5F0100]          <1>         mov    [writei.fs_index], eax ; cluster index (for next check)
7930 0000ED2D A1[705F0100]          <1>         mov    eax, [writei.lclust] ; last cluster
7931 0000ED32 EB60                  <1>         jmp    short mget_w_10
7932                                <1>
7933                                <1> mget_w_4: ; 02/11/2016
7934                                <1>         ; eax = next file position
7935 0000ED34 2B05[88030300]        <1>         sub    eax, [u.count] ; current file position
7936                                <1>         ; edx = 0
7937                                <1>         ; ecx = bytes per cluster
7938 0000ED3A F7F1                  <1>         div    ecx
7939 0000ED3C A3[645F0100]          <1>         mov    [writei.c_index], eax ; cluster index
7940 0000ED41 668915[5E5F0100]      <1>         mov    [writei.offset], dx
7941 0000ED48 66C1EA09              <1>         shr    dx, 9 ; / 512
7942 0000ED4C 8815[575F0100]        <1>         mov    [writei.s_index], dl ; sector index in cluster (0 to spc -1)
7943                                <1>
7944                                <1> mget_w_5:
7945 0000ED52 21C0                  <1>         and    eax, eax ; 0 = First Cluster's index number
7946 0000ED54 750C                  <1>         jnz    short mget_w_6
7947                                <1>
7948 0000ED56 A3[6C5F0100]          <1>         mov    [writei.fs_index], eax ; cluster index (for next check)
7949 0000ED5B A1[685F0100]          <1>         mov    eax, [writei.fclust] ; first cluster
7950 0000ED60 EB32                  <1>         jmp    short mget_w_10
7951                                <1>
7952                                <1> mget_w_6:
7953 0000ED62 3B05[6C5F0100]        <1>         cmp    eax, [writei.fs_index] ; current cluster index (>0)
7954 0000ED68 7507                  <1>         jne    short mget_w_7
7955 0000ED6A A1[605F0100]          <1>         mov    eax, [writei.cluster] ; current cluster
7956 0000ED6F EB3A                  <1>         jmp    short mget_w_11
7957                                <1>
7958                                <1> mget_w_7:
7959 0000ED71 89C1                  <1>         mov    ecx, eax
7960 0000ED73 2B0D[6C5F0100]        <1>         sub    ecx, [writei.fs_index]
7961 0000ED79 730D                  <1>         jnc    short mget_w_8
7962                                <1>         ; get cluster by index from the first cluster
7963 0000ED7B A1[685F0100]          <1>         mov    eax, [writei.fclust]
7964 0000ED80 8B0D[645F0100]        <1>         mov    ecx, [writei.c_index]
7965 0000ED86 EB05                  <1>         jmp    short mget_w_9
7966                                <1>
7967                                <1> mget_w_8:
7968 0000ED88 A1[605F0100]          <1>         mov    eax, [writei.cluster] ; beginning cluster
7969                                <1>         ; ecx = cluster sequence number after the beginning cluster
7970                                <1>         ; sub  edx, edx ; 0
7971                                <1>
7972                                <1> mget_w_9:
7973                                <1>         ; EAX = Beginning cluster
7974                                <1>         ; EDX = Sector index in disk/file section
7975                                <1>         ;     (Only for SINGLIX file system!)
7976                                <1>         ; ECX = Cluster sequence number after the beginning cluster
7977                                <1>         ; ESI = Logical DOS Drive Description Table address
7978 0000ED8D E89AD5FFFF            <1>         call   get_cluster_by_index
7979 0000ED92 7234                  <1>         jc     short mget_w_err ; error code in EAX
7980                                <1>         ; EAX = Cluster number
7981                                <1> mget_w_10:
7982 0000ED94 A3[605F0100]          <1>         mov    [writei.cluster], eax ; FDT number for Singlix File System
7983                                <1>
7984 0000ED99 807E0300              <1>         cmp    byte [esi+LD_FATType], 0
7985 0000ED9D 7638                  <1>         jna    short mget_w_13
7986                                <1>         ; 01/11/2016
7987 0000ED9F 8B15[645F0100]        <1>         mov    edx, [writei.c_index]
7988 0000EDA5 8915[6C5F0100]        <1>         mov    [writei.fs_index], edx
7989                                <1> mget_w_11:
7990 0000EDAB 83E802                <1>         sub    eax, 2
7991 0000EDAE 0FB615[565F0100]      <1>         movzx  edx, byte [writei.spc]
7992 0000EDB5 F7E2                  <1>         mul    edx
7993                                <1>
7994 0000EDB7 034668                <1>         add    eax, [esi+LD_DATABegin]
7995 0000EDBA 8A15[575F0100]        <1>         mov    dl, [writei.s_index]
7996 0000EDC0 01D0                  <1>         add    eax, edx
7997                                <1> mget_w_12:
7998 0000EDC2 A3[585F0100]          <1>         mov    [writei.sector], eax
7999                                <1>         ;; buffer validation must be done in writei
8000                                <1>         ;;mov  byte [writei.valid], 1
8001 0000EDC7 C3                    <1>         retn
8002                                <1>
8003                                <1> mget_w_err:
8004 0000EDC8 A3[C8030300]          <1>         mov    [u.error], eax
8005 0000EDCD A3[64030300]          <1>         mov    [u.r0], eax
8006 0000EDD2 E9BCD6FFFF            <1>         jmp    error
```

```
8007                                    <1>
8008                                    <1> mget_w_13:
8009                                    <1>       ; EAX = FDT number (Current Section)
8010                                    <1>       ; EDX = Sector index from the first section (0,1,2,3,4...)
8011 0000EDD7 2B15[6C5F0100]            <1>       sub    edx, [writei.fs_index]
8012                                    <1>       ; EDX = Sector index from current section
8013 0000EDDD 8915[6C5F0100]            <1>       mov    [writei.fs_index], edx
8014 0000EDE3 40                        <1>       inc    eax ; the first data sector in FS disk section
8015 0000EDE4 01D0                      <1>       add    eax, edx
8016 0000EDE6 EBDA                      <1>       jmp    short mget_w_12
8017                                    <1>
8018                                    <1> mget_w_14:
8019 0000EDE8 8A4E12                    <1>       mov    cl, [esi+LD_FS_BytesPerSec+1]
8020 0000EDEB D0E9                      <1>       shr    cl, 1 ;  ; 1 for 512 bytes, 4 for 2048 bytes
8021 0000EDED 880D[565F0100]            <1>       mov    [writei.spc], cl  ; sectors per cluster
8022                                    <1>       ; NOTE: writei bytes per sector value is always 512 !
8023 0000EDF3 66C705[5C5F0100]00-       <1>       mov    word [writei.bpc], 512
8023 0000EDFB 02                        <1>
8024                                    <1>
8025 0000EDFC 89E9                      <1>       mov    ecx, ebp
8026 0000EDFE 030D[88030300]            <1>       add    ecx, [u.count] ; next file position
8027 0000EE04 3B0D[55040300]            <1>       cmp    ecx, [i.size] ; <= file size ?
8028 0000EE0A 0F86C8000000              <1>       jna    mget_w_19 ; no
8029                                    <1>
8030 0000EE10 29D2                      <1>       sub    edx, edx ; 0
8031 0000EE12 8915[585F0100]            <1>       mov    [writei.sector], edx ; 0
8032 0000EE18 668915[5E5F0100]          <1>       mov    [writei.offset], dx  ; byte offset in cluster
8033 0000EE1F 8815[575F0100]            <1>       mov    [writei.s_index], dl ; sector index in cluster (0 to spc -1)
8034                                    <1>
8035 0000EE25 C1E909                    <1>       shr    ecx, 9 ; 1 cluster = 512 bytes
8036 0000EE28 890D[645F0100]            <1>       mov    [writei.c_index], ecx ; section/cluster index
8037                                    <1>
8038 0000EE2E 89D8                      <1>       mov    eax, ebx ; FDT number (First FDT address)
8039                                    <1>
8040                                    <1>       ; is this the 1st mget_w or a next mget_w call ? (by 'writei')
8041 0000EE30 3815[545F0100]            <1>       cmp    byte [writei.valid], dl ; 0
8042 0000EE36 7624                      <1>       jna    short mget_w_15
8043                                    <1>
8044 0000EE38 8815[545F0100]            <1>       mov    byte [writei.valid], dl ; 0 ; reset ('writei' will set it)
8045                                    <1>
8046 0000EE3E 3B05[685F0100]            <1>       cmp    eax, [writei.fclust]
8047 0000EE44 7516                      <1>       jne    short mget_w_15
8048                                    <1>
8049 0000EE46 8A0D[46030300]            <1>       mov    cl, [cdev]
8050 0000EE4C 3A0D[555F0100]            <1>       cmp    cl, [writei.drv]
8051 0000EE52 7508                      <1>       jne    short mget_w_15
8052                                    <1>       ; [writei.l_clust] & [writei.l_index] are valid,
8053                                    <1>       ;  we don't need to get last cluster & last cluster index
8054 0000EE54 8B0D[745F0100]            <1>       mov    ecx, [writei.l_index]
8055 0000EE5A EB49                      <1>       jmp    short mget_w_17
8056                                    <1> mget_w_15:
8057 0000EE5C A3[685F0100]              <1>       mov    [writei.fclust], eax ; first section (FDT number)
8058                                    <1>       ; edx = 0
8059 0000EE61 8915[605F0100]            <1>       mov    [writei.cluster], edx ; 0 ; current section
8060 0000EE67 8915[6C5F0100]            <1>       mov    [writei.fs_index], edx ; 0 ; curret section index
8061                                    <1>
8062                                    <1>       ; eax = FDT number (section 0 header address)
8063 0000EE6D E8E4D4FFFF                <1>       call   get_last_section
8064 0000EE72 0F8250FFFFFF              <1>       jc     mget_w_err ; eax = error code
8065                                    <1>
8066 0000EE78 8915[6C5F0100]            <1>       mov    [writei.fs_index], edx ; sector index in last section
8067                                    <1>
8068 0000EE7E A3[705F0100]              <1>       mov    [writei.lclust], eax ; last section address
8069                                    <1>
8070 0000EE83 8B0D[945D0100]            <1>       mov    ecx, [glc_index] ; last section index
8071 0000EE89 890D[745F0100]            <1>       mov    [writei.l_index], ecx
8072                                    <1>
8073 0000EE8F A0[785F0100]              <1>       mov    al, [writei.ofn]
8074 0000EE94 FEC0                      <1>       inc    al
8075 0000EE96 A2[B35F0100]              <1>       mov    [setfmod], al ; update lm date&time sign
8076                                    <1>
8077                                    <1> mget_w_16:
8078                                    <1>       ; edx = (existing) last section (sector) index
8079 0000EE9B 8B0D[645F0100]            <1>       mov    ecx, [writei.c_index] ; final section (sector) index
8080 0000EEA1 29D1                      <1>       sub    ecx, edx
8081 0000EEA3 7633                      <1>       jna    short mget_w_19
8082                                    <1>        ; ecx = sector count
8083                                    <1> mget_w_17:
8084 0000EEA5 A1[705F0100]              <1>       mov    eax, [writei.lclust]
8085                                    <1>       ; ESI = Logical dos drv desc. table address
8086                                    <1>         ; EAX = Last section
8087                                    <1>         ; (ECX = 0 for directory)
8088                                    <1>         ; ECX = sector count (except FDT)
8089 0000EEAA E86ACAFFFF                <1>       call   add_new_fs_section
8090 0000EEAF 7312                      <1>       jnc    short mget_w_18
8091                                    <1>
8092                                    <1>       ; If error number = 27h (insufficient disk space)
8093                                    <1>       ; it is needed to check free consequent sectors
8094                                    <1>       ; (1 data sector at least and +1 section header sector)
8095                                    <1>
8096 0000EEB1 83F827                    <1>       cmp    eax, 27h
8097 0000EEB4 0F850EFFFFFF              <1>       jne    mget_w_err ; eax = error code
8098                                    <1>
8099                                    <1>       ; ecx = count of free consequent sectors
8100                                    <1>       ; ecx must be > 1 (1 data + 1 header sector)
8101 0000EEBA 49                        <1>       dec    ecx
8102 0000EEBB 0F8407FFFFFF              <1>       jz     mget_w_err
8103 0000EEC1 EBE2                      <1>       jmp    short mget_w_17
8104                                    <1>
8105                                    <1> mget_w_18:
8106 0000EEC3 A3[705F0100]              <1>       mov    [writei.lclust], eax ; (new) last section
8107                                    <1>       ; ecx = sector count (except section header)
```

```
8108 0000EEC8 8B15[745F0100]      <1>       mov    edx, [writei.l_index]
8109 0000EECE 01CA                <1>       add    edx, ecx ; add sector count to index
8110 0000EED0 8915[745F0100]      <1>       mov    [writei.l_index], edx
8111 0000EED6 EBC3                <1>       jmp    short mget_w_16
8112                              <1>
8113                              <1> mget_w_19:
8114 0000EED8 89E9                <1>       mov    ecx, ebp
8115 0000EEDA 030D[88030300]      <1>       add    ecx, [u.count]
8116 0000EEE0 890D[55040300]      <1>       mov    [i.size], ecx ; save new file size
8117                              <1>       ;sub   edx, edx ; 0
8118                              <1>
8119 0000EEE6 A0[46030300]        <1>       mov    al, [cdev]
8120 0000EEEB A2[555F0100]        <1>       mov    [writei.drv], al ; physical drive number
8121                              <1>       ; edx = 0
8122 0000EEF0 89E8                <1>       mov    eax, ebp ; file offset
8123 0000EEF2 89C2                <1>       mov    edx, eax
8124                              <1>       ; 1 cluster = 512 bytes (for Singlix FS)
8125 0000EEF4 C1E809              <1>       shr    eax, 9  ; / 512
8126 0000EEF7 81E2FF010000        <1>       and    edx, 1FFh
8127                              <1>       ; edx = byte offset in cluster/sector (<= 511)
8128                              <1>       ; eax = section (sector/cluster) index
8129 0000EEFD A3[645F0100]        <1>       mov    [writei.c_index], eax
8130 0000EF02 668915[5E5F0100]    <1>       mov    [writei.offset], dx
8131                              <1>       ;mov   byte [writei.s_index], 0 ; sector index in cluster
8132 0000EF09 E912FEFFFF          <1>       jmp    mget_w_3
8133                              <1>
8134                              <1> update_file_lmdt: ; & update file size
8135                              <1>       ; 26/10/2016
8136                              <1>       ; 24/10/2016
8137                              <1>       ; 23/10/2016
8138                              <1>       ; 22/10/2016 - TRDOS 386 (TRDOS v2.0)
8139                              <1>       ;
8140                              <1>       ; Update last modification date&time of file
8141                              <1>       ; (call from syswrite -> writei)
8142                              <1>       ; ((also updates file size)) // 26/10/2016
8143                              <1>       ;
8144                              <1>       ; INPUT:
8145                              <1>       ;      byte [setfmod] = open file number
8146                              <1>       ; OUTPUT:
8147                              <1>       ;      cf = 0 -> success !
8148                              <1>       ;      cf = 1 -> lmdt update has been failed!
8149                              <1>       ;
8150                              <1>       ; Modified registers: eax, ebx, ecx, edx, esi, edi
8151                              <1>       ;
8152                              <1>
8153                              <1>       ;cmp   byte [setfmod], 0
8154                              <1>       ;jna   short uflmdt_2 ; nothing to do
8155                              <1>
8156 0000EF0E 31C0                <1>       xor    eax, eax
8157                              <1>
8158 0000EF10 0FB61D[B35F0100]    <1>       movzx  ebx, byte [setfmod]
8159 0000EF17 FECB                <1>       dec    bl ; open file index number (0 based)
8160                              <1>
8161 0000EF19 8AA3[20630100]      <1>       mov    ah, [ebx+OF_DRIVE]
8162 0000EF1F BE00010900          <1>       mov    esi, Logical_DOSDisks
8163 0000EF24 01C6                <1>       add    esi, eax
8164 0000EF26 C0E302              <1>       shl    bl, 2 ; *4
8165 0000EF29 8B8B[F8620100]      <1>       mov    ecx, [ebx+OF_FCLUSTER] ; first cluster
8166 0000EF2F 8B93[C0630100]      <1>       mov    edx, [ebx+OF_DIRCLUSTER] ; dir cluster
8167                              <1>
8168 0000EF35 D0EB                <1>       shr    bl, 1 ; /2
8169 0000EF37 0FB7BB[60640100]    <1>       movzx  edi, word [ebx+OF_DIRENTRY]
8170                              <1>
8171 0000EF3E 803D[F05A0100]01    <1>       cmp    byte [DirBuff_ValidData], 1
8172 0000EF45 726E                <1>       jb     short uflmdt_4
8173                              <1>
8174 0000EF47 A0[EE5A0100]        <1>       mov    al, [DirBuff_DRV]
8175 0000EF4C 2C41                <1>       sub    al, 'A'
8176 0000EF4E 38E0                <1>       cmp    al, ah
8177 0000EF50 7563                <1>       jne    short uflmdt_4 ; different drive
8178 0000EF52 8A4603              <1>       mov    al, [esi+LD_FATType]
8179 0000EF55 3A05[EF5A0100]      <1>       cmp    al, [DirBuff_FATType]
8180 0000EF5B 755B                <1>       jne    short uflmdt_5 ; different FS type
8181 0000EF5D 3B15[F55A0100]      <1>       cmp    edx, [DirBuff_Cluster]
8182 0000EF63 7553                <1>       jne    short uflmdt_5 ; different cluster
8183                              <1>
8184                              <1> uflmdt_1:
8185                              <1>       ; Directory buffer is ready here!
8186                              <1>       ; OF_FCLUSTER must be compared/verified
8187 0000EF65 BE00000800          <1>       mov    esi, Directory_Buffer
8188 0000EF6A 66C1E705            <1>       shl    di, 5 ; dir entry index * 32
8189 0000EF6E 01FE                <1>       add    esi, edi ; offset
8190                              <1>       ;
8191 0000EF70 F6460B18            <1>       test   byte [esi+DirEntry_Attr], 18h ; Vol & Dir
8192 0000EF74 750F                <1>       jnz    short uflmdt_2 ; not a valid file !
8193 0000EF76 668B4614            <1>       mov    ax, [esi+DirEntry_FstClusHI]
8194 0000EF7A C1E010              <1>       shl    eax, 16
8195 0000EF7D 668B461A            <1>       mov    ax, [esi+DirEntry_FstClusLO]
8196 0000EF81 39C8                <1>       cmp    eax, ecx ; same first cluster ?
8197 0000EF83 7407                <1>       je     short uflmdt_3 ; yes, it is OK !!!
8198                              <1>
8199                              <1> uflmdt_2:
8200                              <1>       ; save directory buffer if has modified/changed sign
8201                              <1>       ; (It is good to save dir buff even if the searched
8202                              <1>       ; directory entry is not found !?)
8203 0000EF85 E8E0B6FFFF          <1>       call   save_directory_buffer
8204 0000EF8A F9                  <1>       stc    ; update failed
8205 0000EF8B C3                  <1>       retn
8206                              <1>
8207                              <1> uflmdt_3:
8208                              <1>       ; Update directory entry
8209                              <1>       ; 26/10/2016
```

```
8210 0000EF8C D0E3                <1>       shl    bl, 1 ; *2
8211 0000EF8E 8B83[70630100]      <1>       mov    eax, [ebx+OF_SIZE] ; file size
8212 0000EF94 89461C              <1>       mov    [esi+DirEntry_FileSize], eax
8213                              <1>       ;
8214 0000EF97 E830B6FFFF          <1>       call   convert_current_date_time
8215                              <1>       ; OUTPUT -> DX = Date in dos dir entry format
8216                              <1>         ;       AX = Time in dos dir entry format
8217 0000EF9C 66894616            <1>       mov    [esi+DirEntry_WrtTime], ax
8218 0000EFA0 66895618            <1>       mov    [esi+DirEntry_WrtDate], dx
8219 0000EFA4 66895612            <1>       mov    [esi+DirEntry_LastAccDate], dx
8220 0000EFA8 C605[F05A0100]02    <1>       mov    byte [DirBuff_ValidData], 2
8221 0000EFAF E8B6B6FFFF          <1>       call   save_directory_buffer
8222 0000EFB4 C3                  <1>       retn
8223                              <1>
8224                              <1> uflmdt_4:
8225                              <1>       ; Directory buffer sector read&write
8226                              <1>       ; 23/10/2016
8227                              <1>       ;
8228 0000EFB5 8A4603              <1>       mov    al, [esi+LD_FATType]
8229                              <1> uflmdt_5:
8230 0000EFB8 BB[9C090300]        <1>       mov    ebx, rw_buffer ; Common r/w sector buffer addr
8231                              <1>
8232 0000EFBD 20C0                <1>       and    al, al ; 0 = Singlix FS
8233 0000EFBF 0F8492000000        <1>       jz     uflmdt_11
8234                              <1>
8235 0000EFC5 21D2                <1>       and    edx, edx
8236 0000EFC7 7521                <1>       jnz    short uflmdt_9
8237                              <1>
8238 0000EFC9 3C02                <1>       cmp    al, 2  ; 3 = FAT32
8239 0000EFCB 771A                <1>       ja     short uflmdt_8
8240                              <1>
8241 0000EFCD 89F8                <1>       mov    eax, edi ; directory entry index number
8242 0000EFCF 66C1E804            <1>       shr    ax, 4 ; 16 entries per sector
8243 0000EFD3 034664              <1>       add    eax, [esi+LD_ROOTBegin]
8244                              <1>       ; eax = root directory sector
8245                              <1> uflmdt_6:
8246 0000EFD6 50                  <1>       push   eax ; * ; disk sector address
8247 0000EFD7 51                  <1>       push   ecx ; first cluster
8248 0000EFD8 B901000000          <1>       mov    ecx, 1
8249                              <1>       ; ecx = sector count
8250 0000EFDD E8D0010000          <1>       call   disk_read
8251 0000EFE2 59                  <1>       pop    ecx
8252 0000EFE3 731A                <1>       jnc    short uflmdt_10
8253 0000EFE5 58                  <1>       pop    eax ; *
8254                              <1> uflmdt_7:
8255 0000EFE6 C3                  <1>       retn
8256                              <1>
8257                              <1> uflmdt_8:
8258 0000EFE7 8B5632              <1>       mov    edx, [esi+LD_BPB+FAT32_RootFClust]
8259                              <1> uflmdt_9:
8260 0000EFEA 83FA02              <1>       cmp    edx, 2
8261 0000EFED 72F7                <1>       jb     short uflmdt_7 ; invalid, nothing to do
8262                              <1>
8263 0000EFEF 83EA02              <1>       sub    edx, 2
8264 0000EFF2 89D0                <1>       mov    eax, edx
8265 0000EFF4 0FB65613            <1>       movzx  edx, byte [esi+LD_BPB+SecPerClust]
8266 0000EFF8 F7E2                <1>       mul    edx
8267 0000EFFA 034668              <1>       add    eax, [esi+LD_DATABegin]
8268                              <1>       ; eax = sub directory (data) sector
8269 0000EFFD EBD7                <1>       jmp    short uflmdt_6
8270                              <1>
8271                              <1> uflmdt_10:
8272                              <1>       ; Directory sector buffer is ready here!
8273                              <1>       ; OF_FCLUSTER must be compared/verified
8274                              <1>       ; edi = dir entry index number (<= 2047)
8275 0000EFFF 6683E70F            <1>       and    di, 0Fh ; 16 entries per sector
8276 0000F003 66C1E705            <1>       shl    di, 5 ; dir entry index * 32
8277 0000F007 81C7[9C090300]      <1>       add    edi, rw_buffer
8278                              <1>       ;
8279 0000F00D F6470B18            <1>       test   byte [edi+DirEntry_Attr], 18h ; Vol & Dir
8280 0000F011 0F856EFFFFFF        <1>       jnz    uflmdt_2 ; not a valid file !
8281 0000F017 668B5714            <1>       mov    dx, [edi+DirEntry_FstClusHI]
8282 0000F01B C1E210              <1>       shl    edx, 16
8283 0000F01E 668B571A            <1>       mov    dx, [edi+DirEntry_FstClusLO]
8284 0000F022 39CA                <1>       cmp    edx, ecx ; same first cluster ?
8285 0000F024 0F855BFFFFFF        <1>       jne    uflmdt_2 ; no !?
8286                              <1>
8287                              <1>       ; Update directory entry
8288 0000F02A E89DB5FFFF          <1>       call   convert_current_date_time
8289                              <1>       ; OUTPUT -> DX = Date in dos dir entry format
8290                              <1>         ;       AX = Time in dos dir entry format
8291 0000F02F 66894716            <1>       mov    [edi+DirEntry_WrtTime], ax
8292 0000F033 66895718            <1>       mov    [edi+DirEntry_WrtDate], dx
8293 0000F037 66895712            <1>       mov    [edi+DirEntry_LastAccDate], dx
8294                              <1>
8295 0000F03B 58                  <1>       pop    eax ; *
8296                              <1>
8297 0000F03C BB[9C090300]        <1>       mov    ebx, rw_buffer ; Common r/w sector buffer addr
8298 0000F041 B901000000          <1>       mov    ecx, 1
8299                              <1>       ; esi = logical dos description table address
8300                              <1>       ; eax = disk sector number/address (LBA)
8301                              <1>       ; ecx = sector count
8302                              <1>       ; ebx = buffer address
8303 0000F046 E858010000          <1>       call   disk_write
8304 0000F04B 0F8234FFFFFF        <1>       jc     uflmdt_2
8305                              <1>
8306                              <1>       ; save directory buffer if has modified/changed sign
8307 0000F051 E814B6FFFF          <1>       call   save_directory_buffer
8308 0000F056 C3                  <1>       retn
8309                              <1>
8310                              <1> uflmdt_11:
8311                              <1>       ; 24/10/2016
```

```
8312                                    <1>       ; Update last modification date & time of a file
8313                                    <1>       ; on a disk with Singlix File System.
8314                                    <1>       ;
8315                                    <1>       ; (Method: Read the FDT -File Description Table-
8316                                    <1>       ; sector of the file and update the lmdt data fields,
8317                                    <1>       ; then write FDT sector to the disk.
8318                                    <1>       ; /// It is easy but there is compatibility buffer
8319                                    <1>       ; method also for changing directory entry data and
8320                                    <1>       ; also there are some programming issues for Singlix
8321                                    <1>       ; file system (TRFS), which are not completed yet!)
8322                                    <1>       ;
8323                                    <1>       ; Not ready yet ! (24/10/2016)
8324                                    <1>       ; /// Temporary code for error return ! ///
8325 0000F057 31C0                     <1>       xor    eax, eax
8326 0000F059 F9                       <1>       stc
8327 0000F05A C3                       <1>       retn
8328                                    <1>
8329                                    <1> sysalloc:
8330                                    <1>       ; 14/10/2017
8331                                    <1>       ; 20/08/2017, 01/09/2017
8332                                    <1>       ; 20/02/2017, 04/03/2017, 15/05/2017
8333                                    <1>       ; 19/02/2017 - TRDOS 386 (TRDOS v2.0)
8334                                    <1>       ; (TRDOS 386 feature only!)
8335                                    <1>       ;
8336                                    <1>       ; Allocate Contiguous Memory Block/Pages (for user)
8337                                    <1>       ; (System call for DMA Buffer allocation etc.)
8338                                    <1>       ;
8339                                    <1>       ; INPUT ->
8340                                    <1>       ;     EBX = Virtual address (for user)
8341                                    <1>       ;          (Physical memory block/aperture
8342                                    <1>       ;          will be mapped to this virtual address)
8343                                    <1>       ;     ECX = Byte Count
8344                                    <1>       ;          (will be rounded up to page border)
8345                                    <1>       ;     If ECX = 0
8346                                    <1>       ;         System call will return with an error (cf=1)
8347                                    <1>       ;         but ECX will contain maximum size of
8348                                    <1>       ;         available memory aperture and physical
8349                                    <1>       ;         (beginning) address of that aperture
8350                                    <1>       ;         (which have maximum size) will be in EAX.
8351                                    <1>       ;     EDX = Upper limit of the requested physical memory
8352                                    <1>       ;           block/pages.
8353                                    <1>       ;          (The last byte address of the memory aperture
8354                                    <1>       ;           must not be equal to or above this limit.)
8355                                    <1>       ;     If EDX = 0
8356                                    <1>       ;        there is NOLIMIT !
8357                                    <1>       ;     If EDX = 0FFFFFFFFh (-1)
8358                                    <1>       ;        ESI = Lower Limit !
8359                                    <1>       ;            (Beginning of the block must not be 'less'
8360                                    <1>       ;            than this.) (Must be equal to or above...)
8361                                    <1>       ;        EDI = Upper Limit !
8362                                    <1>       ;            (End of the block must be !less! than this)
8363                                    <1>       ;            (The last byte addr of the memory aperture
8364                                    <1>       ;            must not be equal to or above this limit.)
8365                                    <1>       ;
8366                                    <1>       ; OUTPUT ->
8367                                    <1>       ;     If CF = 0
8368                                    <1>       ;     EAX = Physical address of the allocated memory block
8369                                    <1>       ;     ECX = Allocated bytes (as rounded up to page borders)
8370                                    <1>       ;     EBX = Virtual address (as rounded up)
8371                                    <1>       ;     IF CF = 1
8372                                    <1>       ;         Requested (size of) Memory block could not be
8373                                    <1>       ;         allocated to the user!
8374                                    <1>       ;     IF CF = 1 & EAX = 0 (Insufficient memory error!)
8375                                    <1>       ;        ECX = Total number of free bytes
8376                                    <1>       ;              (not size of available contiguous bytes!)
8377                                    <1>       ;     If CF = 1 & EAX > 0
8378                                    <1>       ;        there is not a memory aperture with requested size
8379                                    <1>       ;        but total free mem is not less than requested size.
8380                                    <1>       ;        EAX = Physical addr of available memory aperture
8381                                    <1>       ;             with max size
8382                                    <1>       ;             (but it doesn't fit to the conditions!)
8383                                    <1>       ;        ECX = Size of available memory aperture in bytes.
8384                                    <1>       ;     If CF = 1 -> EAX = 0FFFFFFFFh
8385                                    <1>       ;        Conditions/Parameters are wrong !
8386                                    <1>       ;        ECX is same with input value.
8387                                    <1>       ;
8388                                    <1>       ; Note:      Previously allocated pages will be deallocated if
8389                                    <1>       ;       new allocation conditions are met.
8390                                    <1>       ;
8391                                    <1>       ; Note: u.break control may be included in future versions
8392                                    <1>       ;
8393                                    <1>
8394 0000F05B 31C0                     <1>       xor    eax, eax ; 0
8395                                    <1>       ; 14/10/2017
8396 0000F05D 4A                       <1>       dec    edx ; is there a limit ?
8397 0000F05E 7810                     <1>       js     short sysalloc_1 ;  0 -> 0FFFFFFFFh -> NO LIMIT
8398 0000F060 42                       <1>       inc    edx ; > 0
8399                                    <1>       ; Check upper address limit
8400                                    <1>       ;(round up to page borders)
8401 0000F061 81C1FF0F0000             <1>       add    ecx, PAGE_SIZE-1 ; 4095
8402 0000F067 6681E100F0               <1>       and    cx, ~PAGE_OFF ; not 4095
8403 0000F06C 39CA                     <1>       cmp    edx, ecx ; upper limit - block size
8404 0000F06E 7224                     <1>       jb     short sysalloc_err
8405                                    <1> sysalloc_1:
8406                                    <1>       ; EAX = Beginning address (physical)
8407                                    <1>       ; EAX = 0 -> Allocate mem block from the 1st proper aperture
8408                                    <1>       ; ECX = Number of bytes to be allocated
8409 0000F070 E8AE63FFFF               <1>       call   allocate_memory_block
8410 0000F075 721D                     <1>       jc     short sysalloc_err
8411                                    <1>       ; 01/09/2017
8412 0000F077 29C2                     <1>       sub    edx, eax ; upper limit address - beginning address
8413 0000F079 760F                     <1>       jna    short sysalloc_3 ; begin addr not less than the limit
```

```
8414 0000F07B 39CA              <1>       cmp    edx, ecx
8415 0000F07D 720B              <1>       jb     short sysalloc_3 ; end address overs the limit
8416                            <1> sysalloc_2:
8417                            <1>       ; EAX = Beginning (physical) addr of the allocated mem block
8418                            <1>       ; ECX = Num of allocated bytes (rounded up to page borders)
8419 0000F07F 50                <1>       push   eax ; * ; 04/03/2017
8420                            <1>       ; Here, requested contiquous memory pages have been allocated
8421                            <1>       ; on Memory Allocation Table but user's page directory
8422                            <1>       ; and page tables have not been updated yet!
8423 0000F080 51                <1>       push   ecx ; **
8424                            <1>       ; ebx = virtual address (will be rounded up to page border)
8425                            <1>       ; ecx = number of bytes to be deallocated
8426                            <1>       ;      will be adjusted to ebx+ecx round down - ebx round up
8427 0000F081 E8F866FFFF        <1>       call   deallocate_user_pages
8428 0000F086 731F              <1>       jnc    short sysalloc_4 ; EAX = Deallocated memory bytes
8429 0000F088 59                <1>       pop    ecx ; **
8430 0000F089 58                <1>       pop    eax ; *
8431                            <1> sysalloc_3:
8432                            <1>       ; error !
8433                            <1>       ; restore Memory Allocation Table Content
8434 0000F08A E8A165FFFF        <1>       call   deallocate_memory_block
8435 0000F08F 31C0              <1>       xor    eax, eax ; 0
8436 0000F091 48                <1>       dec    eax ; 0FFFFFFFFh ; 15/05/2017
8437 0000F092 EB09              <1>       jmp    short sysalloc_wrong
8438                            <1> sysalloc_err:
8439 0000F094 8B2D[60030300]    <1>       mov    ebp, [u.usp]  ; ebp points to user's registers
8440 0000F09A 894D18            <1>       mov    [ebp+24], ecx ; return to user with ecx value
8441                            <1> sysalloc_wrong:
8442                            <1>       ; eax = 0FFFFFFFFh
8443 0000F09D A3[64030300]      <1>       mov    [u.r0], eax
8444 0000F0A2 E9ECD3FFFF        <1>       jmp    error
8445                            <1> sysalloc_4:
8446 0000F0A7 8B2D[60030300]    <1>       mov    ebp, [u.usp]  ; ebp points to user's registers
8447 0000F0AD 894518            <1>       mov    [ebp+24], eax ; return to user with ecx value
8448 0000F0B0 895D10            <1>       mov    [ebp+16], ebx ; new value of ebx (rounded up)
8449 0000F0B3 89C1              <1>       mov    ecx, eax ; byte count (from 'deallocate_user_pages')
8450 0000F0B5 5A                <1>       pop    edx ; ** ; discard (another) byte count
8451 0000F0B6 58                <1>       pop    eax ; *
8452 0000F0B7 A3[64030300]      <1>       mov    [u.r0], eax ; physical address
8453                            <1>
8454 0000F0BC 51                <1>       push   ecx ; 20/08/2017
8455                            <1>       ;
8456                            <1>       ; Write newly allocated contiguous (physical) pages
8457                            <1>       ; on page dir and page tables of current user/process
8458                            <1>       ; as PRESENT, USER, WRITABLE
8459                            <1>       ; (then clear allocated pages)
8460 0000F0BD E8B167FFFF        <1>       call   allocate_user_pages
8461                            <1>       ;jnc   sysret ; OK! return to process with success...
8462                            <1>
8463                            <1>       ; 20/08/2017 ('sysdma' modification)
8464 0000F0C2 59                <1>       pop    ecx
8465 0000F0C3 A1[64030300]      <1>       mov    eax, [u.r0]   ; physical address (of the block)
8466                            <1>
8467 0000F0C8 721D              <1>       jc     short sysalloc_6
8468                            <1>
8469 0000F0CA 833D[C8690100]FF  <1>       cmp    dword [dma_addr], 0FFFFFFFFh ; -1
8470 0000F0D1 0F82DCD3FFFF      <1>       jb     sysret
8471                            <1>
8472 0000F0D7 A3[C8690100]      <1>       mov    [dma_addr], eax ; save dma address for sysdma
8473 0000F0DC 890D[CC690100]    <1>       mov    [dma_size], ecx ; save dma buff size for sysdma
8474                            <1>
8475 0000F0E2 E9CCD3FFFF        <1>       jmp    sysret
8476                            <1>
8477                            <1> sysalloc_6:
8478                            <1>       ;
8479                            <1>       ; unexpected error ! insufficient memory !? conflict !?
8480                            <1>       ; (!!?there is not a free page for a new page table?!!)
8481                            <1>       ; We need to terminate process with error message !!!
8482                            <1>       ;
8483 0000F0E7 8B2D[60030300]    <1>       mov    ebp, [u.usp]  ; ebp points to user's registers
8484 0000F0ED 8B4D18            <1>       mov    ecx, [ebp+24] ; byte count
8485                            <1>
8486                            <1>       ; 20/08/2017
8487                            <1>       ;mov   eax, [u.r0]   ; physical address (of the block)
8488                            <1>
8489                            <1>       ;
8490                            <1>       ; restore Memory Allocation Table Content
8491 0000F0F0 E83B65FFFF        <1>       call   deallocate_memory_block
8492                            <1>       ;
8493 0000F0F5 803D[C25E0000]03  <1>       cmp    byte [CRT_MODE], 3 ; 80x25 text mode?
8494 0000F0FC 7407              <1>       je     short sysalloc_7 ; yes
8495                            <1>       ; Current mode is VGA (or CGA graphics) mode,
8496                            <1>       ; We need to return to text mode for displaying
8497                            <1>       ; error message just before 'sysexit'.
8498 0000F0FE B003              <1>       mov    al, 3
8499 0000F100 E86024FFFF        <1>       call   _set_mode
8500                            <1> sysalloc_7:
8501 0000F105 BE[0E0A0100]      <1>       mov    esi, beep_Insufficient_Memory ; error message
8502 0000F10A E84E72FFFF        <1>       call   print_msg ; print/display the message
8503 0000F10F B801000000        <1>       mov    eax, 1 ; ax=1 is needed for 'sysexit' procedure
8504 0000F114 E921D5FFFF        <1>       jmp    sysexit ; and terminate the process !
8505                            <1>
8506                            <1> sysdalloc:
8507                            <1>       ; 19/02/2017 - TRDOS 386 (TRDOS v2.0)
8508                            <1>       ; (TRDOS 386 feature only!)
8509                            <1>       ;
8510                            <1>       ; Deallocate Memory Block/Pages (for user)
8511                            <1>       ; (Complementary call for sysalloc.)
8512                            <1>       ;
8513                            <1>       ; INPUT ->
8514                            <1>       ;      EBX = Virtual address (for user)
8515                            <1>       ;            (will be rounded up to page border)
```

```
                                <1>       ;     ECX = Byte Count
                                <1>       ;          (will be adjusted to page borders)
                                <1>       ;     If ICX = 0
                                <1>       ;        nothing to do
                                <1>       ;     If EBX + ECX > User's ESP
                                <1>       ;        nothing to do
                                <1>       ;
                                <1>       ; Note: u.break control may be included in future versions
                                <1>       ;
                                <1>       ; OUTPUT ->
                                <1>       ;     If CF = 0
                                <1>       ;        EAX = Deallocated memory bytes
                                <1>       ;        EBX = Virtual address (as rounded up)
                                <1>       ;     IF CF = 1
                                <1>       ;        EAX = 0
                                <1>       ;
                                <1>       ; Note:      Main purpose of this call is to deallocate/release
                                <1>       ;     previously allocated (physically) contiguous memory
                                <1>       ;     pages but beginning (virtual) address may not be
                                <1>       ;     followed by physically contiguous pages. So, this
                                <1>       ;     system call will deallocate user's virtually
                                <1>       ;     contiguous memory pages. Also, there is not any
                                <1>       ;     objections to use this system call without sysalloc
                                <1>       ;     system call; only possible objection is to lost data
                                <1>       ;     within user's memory space, if the beginning address
                                <1>       ;     and size is not proper.
                                <1>       ;
                                <1>       ; Note: Empty page tables will not be deallocated!!!
                                <1>       ;       (they will be deallocated at process termination)
                                <1>       ;
                                <1>       ; Note: When the program terminates itself or when it is
                                <1>       ;     terminated by operating system kernel, all allocated
0000F119 8B15[5C030300]         <1>       ;     memory pages will be deallocated during termination
0000F11F 8B420C                 <1>       ;     stage. So, 'sysdalloc' is not necessary except
0000F122 29C8                   <1>       ;     forgiving memory block to other programs/processes.
0000F124 24FC                   <1>       ;
0000F126 39D8                   <1>       mov   edx, [u.sp]
0000F128 7220                   <1>       mov   eax, [edx+12] ; user's stack pointer
                                <1>       sub   eax, ecx ; esp - byte count
0000F12A 31C0                   <1>       and   al, 0FCh ; dword alignment
0000F12C 21C9                   <1>       cmp   eax, ebx
0000F12E 7407                   <1>       jb    short sysdalloc_err ; deallocation overlaps with stack
                                <1>
0000F130 E84966FFFF             <1>       xor   eax, eax
0000F135 7213                   <1>       and   ecx, ecx
                                <1>       jz    short sysdalloc_2
                                <1>
0000F137 A3[64030300]           <1>       call  deallocate_user_pages
0000F13C 8B2D[60030300]         <1>       jc    short sysdalloc_err
0000F142 895D10                 <1>
0000F145 E969D3FFFF             <1>
                                <1> sysdalloc_2:
0000F14A A3[64030300]           <1>       mov   [u.r0], eax
0000F14F E93FD3FFFF             <1>       mov   ebp, [u.usp]
                                <1>       mov   [ebp+16], ebx ; new value of ebx
                                <1>       jmp   sysret
                                <1>
                                <1> sysdalloc_err:
                                <1>       mov   [u.r0], eax ; 0
                                <1>       jmp   error
                                <1>
                                <1> syscalbac:
                                <1>       ; SYS CALLBACK
                                <1>       ; 16/04/2017
                                <1>       ; 14/04/2017
                                <1>       ; 13/04/2017
                                <1>       ; 28/02/2017
                                <1>       ; 26/02/2017
                                <1>       ; 24/02/2017
                                <1>       ; 21/02/2017 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; (TRDOS 386 feature only!)
                                <1>       ;
                                <1>       ; Link or unlink IRQ callback service to/from user (ring 3)
                                <1>       ;
                                <1>       ; INPUT ->
                                <1>       ;     BL = IRQ number (Hardware interrupt request number)
                                <1>       ;          (0 t0 15 but IRQ 0,1,2,6,8,14,15 are prohibited)
                                <1>       ;          IRQ numbers 3,4,5,7,9,10,11,12,13 are valid
                                <1>       ;          (numbers >15 are invalid)
                                <1>       ;
                                <1>       ;     BH = 0 = Unlink IRQ (in BL) from user (ring 3) service
                                <1>       ;          1 = Link IRQ by using Signal Response Byte method
                                <1>       ;          2 = Link IRQ by using Callback service method
                                <1>       ;          3 = Link IRQ by using Auto Increment S.R.B. method
                                <1>       ;         >3 = invalid
                                <1>       ;
                                <1>       ;     CL = Signal Return/Response Byte value
                                <1>       ;
                                <1>       ;     If BH = 2, kernel will put a counter value
                                <1>       ;              (into the S.R.B. addr)
                                <1>       ;            between 0 to 255. (start value = CL+1)
                                <1>       ;
                                <1>       ;     NOTE: counter value, for example: even and odd numbers
                                <1>       ;           may be used for -audio- DMA buffer switch
                                <1>       ;           within double buffer method, etc.
                                <1>       ;
                                <1>       ;     EDX = Signal return (Response) byte address
                                <1>       ;                         - or -
                                <1>       ;           Interrupt/Callback service/routine address
                                <1>       ;
                                <1>       ;           (virtual address in user's memory space)
                                <1>       ;
                                <1>       ; OUTPUT ->
```

```
8618                                    <1>     ;     CF = 0 & EAX = 0 -> Successful setting
8619                                    <1>     ;     CF = 1 & EAX > 0 -> IRQ is prohibited or locked
8620                                    <1>     ;                        by another process
8621                                    <1>     ;                eax = ERR_PERM_DENIED -> prohibited or locked
8622                                    <1>     ;                eax = ERR_INV_PARAMETER ->
8623                                    <1>     ;                      invalid parameter/option or bad address
8624                                    <1>     ;
8625                                    <1>     ;     NOTE: Timer callbacks are set by using 'systimer'
8626                                    <1>     ;           system call (IRQ 0, PIT and IRQ 8, RTC)
8627                                    <1>     ;
8628                                    <1>     ;           Direct keyboard access is performed by using
8629                                    <1>     ;           Keyboard Interrupt (INT 32h)
8630                                    <1>     ;
8631                                    <1>     ;           It is prohibited here because:
8632                                    <1>     ;            1) Signal Response Byte method has not advantage
8633                                    <1>     ;               against INT 32h, function AH = 1. Also,
8634                                    <1>     ;               keyboard service interrupt will return with
8635                                    <1>     ;               ascii and scan codes (AL, AH) while
8636                                    <1>     ;               SRB method has only 1 byte space for ascii code
8637                                    <1>     ;               or scan code. One byte signal response is used
8638                                    <1>     ;               for ensuring very simple and very fast
8639                                    <1>     ;               virtual to physical memory address conversion
8640                                    <1>     ;               without any memory page crossover risk.
8641                                    <1>     ;               (Otherwise double page conversion or word
8642                                    <1>     ;               alignment would be needed.)
8643                                    <1>     ;            2) Badly written user code (callback code)
8644                                    <1>     ;               can prevent keyboard and timesharing functions
8645                                    <1>     ;               of the operating system via continuous and long
8646                                    <1>     ;               keyboard event handling by callback service.
8647                                    <1>     ;               (It can cause to lose immediate keystroke
8648                                    <1>     ;               response from hardware to user.)
8649                                    <1>     ;            3) If user will check any keyboard events, 'getkey'
8650                                    <1>     ;               (or 'getchar') must have more priority than other
8651                                    <1>     ;               (video etc.) events because only control ability
8652                                    <1>     ;               on a procedural infinite loop is a keyboard or
8653                                    <1>     ;               mouse event. So user can use keyboard function
8654                                    <1>     ;               at the end or at the beginning of a loop.
8655                                    <1>     ;               In this case, INT 32h is used for that purpose
8656                                    <1>     ;               and timer interrupt etc. callbacks can be used
8657                                    <1>     ;               for dynamic and synchronized data refresh/transfer
8658                                    <1>     ;               while cpu is in a static loop (without polling).
8659                                    <1>     ;               Keyboard Int callback is not more useful because
8660                                    <1>     ;               already a manual check (a key is pressed or not)
8661                                    <1>     ;               can be performed (via INT 32h, AH = 1) efficiently
8662                                    <1>     ;               in a loop to prevent a locked infinitive loop.
8663                                    <1>     ;
8664                                    <1>     ;            Disk IRQs (6,14,15) have been phohibited from ring 3
8665                                    <1>     ;            callback because, disk operations (file system services
8666                                    <1>     ;            etc.) are independent from user program, for fast disk r/w.
8667                                    <1>     ;            They are not more useful at ring 3 while they are in use
8668                                    <1>     ;            by standard diskio functions which are mandatory part of
8669                                    <1>     ;            (monolithic) OS kernel and mainprog command interpreter.
8670                                    <1>     ;            INT 33h diskio functions are enough for user level disk
8671                                    <1>     ;            r/w.
8672                                    <1>     ;
8673                                    <1>     ; TRDOS 386 - IRQ CALLBACK structures (parameters):
8674                                    <1>     ;
8675                                    <1>     ;           [u.irqlock] = 1 word, IRQ flags (0-15) that indicates
8676                                    <1>     ;                     which IRQs are locked by (that) user.
8677                                    <1>     ;                       Lock and unlock (by user) will change
8678                                    <1>     ;                     these flags or 'terminate process' (sysexit)
8679                                    <1>     ;                     will clear these flags and unlock those IRQs.
8680                                    <1>     ;
8681                                    <1>     ;                     Bit 0 is for IRQ 0 and Bit 15 is for IRQ 15
8682                                    <1>     ;
8683                                    <1>     ;           IRQ(x).owner      : 1 byte, user, [u.uno], 0 = free (unlocked)
8684                                    <1>     ;
8685                                    <1>     ;           IRQ(x).method : 1 byte for callback method & status
8686                                    <1>     ;                           0 = Signal Response Byte method
8687                                    <1>     ;                           1 = Callback service method
8688                                    <1>     ;                           >1 = invalid for current 'syscalback'.
8689                                    <1>     ;                         or(+) 80h = IRQ is in use by system (ring 0)
8690                                    <1>     ;                                     function (audio etc.) or
8691                                    <1>     ;                                   a device driver.
8692                                    <1>     ;                         (system function will ignore the lock/owner)
8693                                    <1>     ;
8694                                    <1>     ;           IRQ(x).srb: 1 byte, Signal Return/Response byte value
8695                                    <1>     ;                        (a fixed value by user or a counter value
8696                                    <1>     ;                       from 0 to 255, which is increased by every
8697                                    <1>     ;                       interrupt just before putting it into
8698                                    <1>     ;                       the Signal Response byte address
8699                                    <1>     ;                       (This is not used in callback serv method)
8700                                    <1>     ;
8701                                    <1>     ;           IRQ(x).addr       : 1 dword
8702                                    <1>     ;                        Signal Response Byte address (physical)
8703                                    <1>     ;                                   -or-
8704                                    <1>     ;                        Callback service address (virtual)
8705                                    <1>     ;
8706                                    <1>     ;           IRQ(x).dev: 1 byte
8707                                    <1>     ;                        0 = Default device or kernel function
8708                                    <1>     ;                                   -or-
8709                                    <1>     ;                        1-255 = Assigned device driver number
8710                                    <1>     ;
8711                                    <1>     ;           (x) = 3,4,5,7,9,10,11,12,13
8712                                    <1>     ;
8713                                    <1>     ;
8714                                    <1>     ;     NOTE: If user's process/program calls the kernel (INT 40h)
8715                                    <1>     ;           while it is already running in a (ring 3) callback
8716                                    <1>     ;           service, kernel will force (convert) system call to
8717                                    <1>     ;           'sysrele' (sys release). So, this feature provides
8718                                    <1>     ;           easy and simple usage of callback services without
8719                                    <1>     ;           falling into deepless <please 'callback me' then
```

```
8720                                    <1>       ;           let me 'callback you'> cycles! (User must return
8721                                    <1>       ;           from callback service by using 'sysrele' system
8722                                    <1>       ;           call, without a significant delay. Otherwise user
8723                                    <1>       ;           process/program may be late to catch the next event
8724                                    <1>       ;           within same callback purpose.
8725                                    <1>       ;
8726                                    <1>
8727 0000F154 30C0                     <1>       xor    al, al ; the caller is 'syscalbac' sign/flag
8728 0000F156 E827110000               <1>       call   set_irq_callback_service
8729                                    <1>       ; 16/04/2017
8730 0000F15B A3[64030300]             <1>       mov    [u.r0], eax
8731 0000F160 0F834DD3FFFF             <1>       jnc    sysret
8732 0000F166 A3[C8030300]             <1>       mov    dword [u.error], eax
8733 0000F16B E923D3FFFF               <1>       jmp    error
8734                                    <1>
8735                                    <1> sysfpstat:
8736                                    <1>       ; 28/02/2017 - TRDOS 386 (TRDOS v2.0)
8737                                    <1>       ; (TRDOS 386 feature only!)
8738                                    <1>       ;
8739                                    <1>       ; Set or reset FPU registers save/restore option (for user)
8740                                    <1>       ;            (during software task switching, wswap-rswap)
8741                                    <1>       ;
8742                                    <1>       ; INPUT ->
8743                                    <1>       ;     BL = 0 -> reset
8744                                    <1>       ;     BL = 1 -> set (FPU register will be saved and restored)
8745                                    <1>       ;
8746                                    <1>       ; OUTPUT ->
8747                                    <1>       ;     cf = 0 -> no error, FPU is ready...
8748                                    <1>       ;              (EAX = 0)
8749                                    <1>       ;     Cf = 1 -> error, 80387 FPU is not ready !
8750                                    <1>       ;              (EAX = 0FFFFFFFFh)
8751                                    <1>
8752 0000F170 31C0                     <1>       xor    eax, eax
8753 0000F172 803D[C05F0100]00         <1>       cmp    byte [fpready], 0
8754 0000F179 7613                     <1>       jna    short sysfpstat_err
8755                                    <1>
8756 0000F17B 80E301                   <1>       and    bl, 1 ; use BIT 0 only !
8757 0000F17E 881D[DA030300]           <1>       mov    [u.fpsave], bl
8758 0000F184 A3[64030300]             <1>       mov    [u.r0], eax ; 0
8759 0000F189 E925D3FFFF               <1>       jmp    sysret
8760                                    <1>
8761                                    <1> sysfpstat_err:
8762 0000F18E 48                       <1>       dec    eax ; 0FFFFFFFFh
8763 0000F18F A3[64030300]             <1>       mov    [u.r0], eax ; -1
8764 0000F194 E9FAD2FFFF               <1>       jmp    error
8765                                    <1>
8766                                    <1> ;maknod:
8767                                    <1>       ; 26/10/2016
8768                                    <1>       ; temporary
8769                                    <1> ;     retn
8770                                    <1>
8771                                    <1> ; temporary - 24/01/2016
8772                                    <1>
8773                                    <1> iget:
8774 0000F199 C3                       <1>       retn
8775                                    <1> isintr:
8776 0000F19A C3                       <1>       retn
8777                                    <1> iopen:
8778 0000F19B C3                       <1>       retn
8779                                    <1> iclose:
8780 0000F19C C3                       <1>       retn
8781                                    <1> setimod:
8782 0000F19D C3                       <1>       retn
8783                                    <1> sndc:
8784 0000F19E C3                       <1>       retn
8785                                    <1> access:
8786 0000F19F C3                       <1>       retn
8787                                    <1> epoch:
8788 0000F1A0 C3                       <1>       retn
8789                                    <1> sleep:
8790 0000F1A1 C3                       <1>       retn
8791                                    <1> set_date_time:
8792 0000F1A2 C3                       <1>       retn
2310                                        %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 0000F1A3 807E0500                  <1>       cmp    byte [esi+LD_LBAYes], 0
  20 0000F1A7 777B                      <1>         ja      short lba_write
  21                                    <1>
  22                                    <1> chs_write:
  23                                    <1>       ; 25/02/2016
  24                                    <1>       ; 23/02/2016
  25 0000F1A9 C605[B95B0100]03          <1>       mov    byte [disk_rw_op], 3 ; CHS write
  26 0000F1B0 EB0D                      <1>       jmp    short chs_rw
  27                                    <1>
  28                                    <1> disk_read:
```

```
                            <1>       ; 25/02/2016
                            <1>       ; 24/02/2016
                            <1>       ; 23/02/2016
                            <1>       ; 17/02/2016
                            <1>       ; 14/02/2016
                            <1>       ; 31/01/2016 (TRDOS 386 =  TRDOS v2.0)
                            <1>       ; 17/10/2010
                            <1>       ; 18/04/2010
                            <1>       ;
                            <1>       ; INPUT -> EAX = Logical Block Address
                            <1>       ;         ESI = Logical Dos Disk Table Offset (DRV)
                            <1>       ;         ECX = Sector Count
                            <1>       ;         EBX = Destination Buffer
                            <1>       ; OUTPUT ->
                            <1>       ;         cf = 0 or cf = 1
                            <1>       ; (Modified registers: EAX, EBX, ECX, EDX)
                            <1>
0000F1B2 807E0500           <1>       cmp    byte [esi+LD_LBAYes], 0
0000F1B6 7775               <1>         ja      short lba_read
                            <1>
                            <1> chs_read:
                            <1>       ; 25/02/2016
                            <1>       ; 24/02/2016
                            <1>       ; 23/02/2016
                            <1>       ; 31/01/2016 (TRDOS 386 =  TRDOS v2.0)
                            <1>       ; 20/07/2011
                            <1>       ; 04/07/2009
                            <1>       ;
                            <1>       ; INPUT -> EAX = Logical Block Address
                            <1>       ;         ECX = Number of sectors to read
                            <1>       ;         ESI = Logical Dos Disk Table Offset (DRV)
                            <1>       ;         EBX = Destination Buffer
                            <1>       ; OUTPUT ->
                            <1>       ;         cf = 0 or cf = 1
                            <1>       ; (Modified registers: EAX; EBX, ECX, EDX)
                            <1>
                            <1>       ; 23/02/2016
0000F1B8 C605[B95B0100]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:
0000F1BF C605[BA5B0100]04   <1>       mov    byte [retry_count], 4
                            <1>
                            <1> chs_read_retry:
                            <1>       ;mov   [sector_count], ecx ; 23/02/2016
                            <1>
0000F1C6 50                 <1>       push   eax                ; Linear sector #
0000F1C7 51                 <1>       push   ecx                ; # of FAT/FILE/DIR sectors
                            <1>
0000F1C8 0FB74E1E           <1>       movzx  ecx, word [esi+LD_BPB+SecPerTrack]
                            <1>       ;movzx ecx, byte [disk_rw_spt] ; 23/02/2016
0000F1CC 29D2               <1>       sub    edx, edx
0000F1CE F7F1               <1>       div    ecx
                            <1>       ; eax = track, dx (dl ) = sector (on track)
                            <1>       ;sub   cl, dl ; 24/02/2016 (spt - sec)
                            <1>       ;push  ecx ; *
0000F1D0 6689D1             <1>       mov    cx, dx             ; Sector (zero based)
0000F1D3 6641               <1>       inc    cx                 ; To make it 1 based
0000F1D5 6651               <1>       push   cx
0000F1D7 668B4E20           <1>       mov    cx, [esi+LD_BPB+Heads]
0000F1DB 6629D2             <1>       sub    dx, dx
0000F1DE F7F1               <1>       div    ecx                ; Convert track to head & cyl
                            <1>       ; eax (ax) = cylinder, dx (dl) = head (max. FFh)
0000F1E0 88D6               <1>       mov    dh, dl
0000F1E2 6659               <1>       pop    cx                 ; AX=Cyl, DH=Head, CX=Sector
0000F1E4 8A5602             <1>       mov    dl, [esi+LD_PhyDrvNo]
                            <1>
0000F1E7 88C5               <1>       mov    ch, al             ; NOTE: max. 1023 cylinders !
0000F1E9 C0CC02             <1>       ror    ah, 2              ; Rotate 2 bits right
0000F1EC 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 !
0000F1EE B001               <1>       mov    al, 1 ; 25/02/2016
0000F1F0 8A25[B95B0100]     <1>       mov    ah, [disk_rw_op]  ; 02h = chs read, 03h = chs write
                            <1>       ;
0000F1F6 E80B50FFFF         <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)
0000F1FB 8825[BB5B0100]     <1>       mov    [disk_rw_err], ah
                            <1>
0000F201 59                 <1>       pop    ecx
0000F202 58                 <1>       pop    eax
0000F203 7314               <1>       jnc    short chs_read_ok
                            <1>
0000F205 803D[BB5B0100]09   <1>       cmp    byte [disk_rw_err], 09h ; DMA crossed 64K segment boundary
0000F20C 7408               <1>       je     short chs_read_error_retn
                            <1>
0000F20E FE0D[BA5B0100]     <1>       dec    byte [retry_count]
```

```
                                <1>       jnz    short chs_read_retry
                                <1>
                                <1> chs_read_error_retn:
0000F216 F9                     <1>       stc
                                <1>       ;retn
0000F217 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
0000F219 40                     <1>       inc    eax ; next sector
0000F21A 81C300020000           <1>       add    ebx, 512
0000F220 E29D                   <1>       loop   chs_read_next_sector
0000F222 EB5E                   <1>       jmp    short update_drv_error_byte
                                <1>
                                <1> lba_write:
                                <1>       ; 23/02/2016
0000F224 C605[B95B0100]1C       <1>       mov    byte [disk_rw_op], 1Ch ; LBA write
0000F22B 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
0000F22D C605[B95B0100]1B       <1>       mov    byte [disk_rw_op], 1Bh ; LBA read
                                <1>
                                <1> lba_rw:
                                <1>       ; 17/02/2016
0000F234 57                     <1>       push   edi
                                <1>
0000F235 890D[BC5B0100]         <1>       mov    [sector_count], ecx ; total sector (read) count
                                <1>
0000F23B 8A5602                 <1>       mov    dl, [esi+LD_PhyDrvNo]
                                <1>       ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)
                                <1>
                                <1> lba_read_next:
0000F23E 81F900010000           <1>       cmp    ecx, 256
0000F244 7605                   <1>       jna    short lba_read_rsc
0000F246 B900010000             <1>       mov    ecx, 256 ; 17/02/2016
                                <1> lba_read_rsc:
0000F24B 290D[BC5B0100]         <1>       sub    [sector_count], ecx ; remain sectors
                                <1>
0000F251 89CF                   <1>       mov    edi, ecx
0000F253 89C1                   <1>       mov    ecx, eax ; sector number/address
                                <1>
0000F255 C605[BA5B0100]04       <1>       mov    byte [retry_count], 4
                                <1> lba_read_retry:
0000F25C 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
0000F25E 8A25[B95B0100]         <1>       mov    ah, [disk_rw_op] ; 1Bh = LBA read, 1Ch = LBA write
0000F264 E89D4FFFFF             <1>       call   int13h
                                <1>       ; al = ? (changed)
                                <1>       ; ah = error code
0000F269 8825[BB5B0100]         <1>       mov    [disk_rw_err], ah
0000F26F 7334                   <1>       jnc    short lba_read_ok
0000F271 80FC80                 <1>       cmp    ah, 80h ; time out?
0000F274 740A                   <1>         je       short lba_read_stc_retn
0000F276 FE0D[BA5B0100]         <1>       dec    byte [retry_count]
0000F27C 7FDE                   <1>       jg     short lba_read_retry
0000F27E 743A                   <1>       jz     short lba_read_reset
                                <1>       ; sf =  1
                                <1>
                                <1> lba_read_stc_retn:
0000F280 F9                     <1>       stc
                                <1> lba_read_retn:
0000F281 5F                     <1>       pop    edi
```

```
                                <1>
                                <1> update_drv_error_byte:
0000F282 9C                     <1>       pushf
0000F283 53                     <1>       push   ebx
0000F284 6651                   <1>       push   cx
                                <1>       ;or    ecx, ecx
                                <1>       ;jz    short udrv_errb0
0000F286 8A0D[BB5B0100]         <1>       mov    cl, [disk_rw_err]
                                <1> udrv_errb0:
0000F28C 0FB65E02               <1>       movzx  ebx, byte [esi+LD_PhyDrvNo]
0000F290 80FB02                 <1>       cmp    bl, 2
0000F293 7203                   <1>         jb     short udrv_errb1
0000F295 80EB7E                 <1>       sub    bl, 7Eh
                                <1>       ;cmp   bl, 5
                                <1>       ;ja    short udrv_errb2
                                <1> udrv_errb1:
0000F298 81C3[495D0000]         <1>         add    ebx, drv.error ; 13/02/2016
0000F29E 880B                   <1>       mov    [ebx], cl ; error code
                                <1> udrv_errb2:
0000F2A0 6659                   <1>       pop    cx
0000F2A2 5B                     <1>       pop    ebx
0000F2A3 9D                     <1>       popf
0000F2A4 C3                     <1>       retn
                                <1>
                                <1> lba_read_ok:
0000F2A5 89C8                   <1>       mov    eax, ecx ; sector number
0000F2A7 01F8                   <1>       add    eax, edi ; sector number (next)
0000F2A9 C1E709                 <1>       shl    edi, 9 ; sector count * 512
0000F2AC 01FB                   <1>       add    ebx, edi ; next buffer offset
                                <1>
0000F2AE 8B0D[BC5B0100]         <1>       mov    ecx, [sector_count] ; remaining sectors
0000F2B4 09C9                   <1>       or     ecx, ecx
0000F2B6 7586                   <1>       jnz    short lba_read_next
0000F2B8 EBC7                   <1>       jmp    short lba_read_retn
                                <1>
                                <1> lba_read_reset:
0000F2BA B40D                   <1>       mov    ah, 0Dh ; Alternate reset
0000F2BC E8454FFFFF             <1>         call int13h
                                <1>       ; al = ? (changed)
                                <1>       ; ah = error code
0000F2C1 7399                   <1>       jnc    short lba_read_retry
0000F2C3 EBBC                   <1>       jmp    short lba_read_retn
                                    %include 'trdosk8.s' ; 24/01/2016
                                <1> ; ****************************************************************************
                                <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - MAIN PROGRAM : trdosk8.s
                                <1> ; ----------------------------------------------------------------------------
                                <1> ; Last Update: 12/10/2017
                                <1> ; ----------------------------------------------------------------------------
                                <1> ; Beginning: 24/01/2016
                                <1> ; ----------------------------------------------------------------------------
                                <1> ; Assembler: NASM version 2.11 (trdos386.s)
                                <1> ; ----------------------------------------------------------------------------
                                <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
                                <1> ; u0.s (20/11/2015), u4.s (14/10/2015)
                                <1> ; ****************************************************************************
                                <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
                                <1> ; TRDOS2.ASM (09/11/2011)
                                <1> ; ----------------------------------------------------------------------------
                                <1> ; DIR.ASM (c) 2004-2011 Erdogan TAN  [07/01/2004] Last Update: 09/10/2011
                                <1>
                                <1> set_run_sequence:
                                <1>       ; 23/12/2016
                                <1>       ; 10/06/2016
                                <1>       ; 22/05/2016
                                <1>       ; 20/05/2016
                                <1>       ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>       ; TRDOS 386 feature only !
                                <1>       ;
                                <1>       ; INPUT ->
                                <1>       ;     AL = process number (next process)
                                <1>       ;
                                <1>       ;     this process must be added to run sequence
                                <1>       ;
                                <1>       ;     [u.pri] = priority of present process
                                <1>       ;
                                <1>       ;     DL = priority (queue)
                                <1>       ;          0 = background (low) ; run on background
                                <1>       ;          1 = regular (normal) ; run as regular
                                <1>       ;          2 = event (high)     ; run for event
                                <1>       ;
                                <1>       ;     1) If the requested process is already running:
                                <1>       ;        a) If present priority is high ([u.pri]=2)
                                <1>       ;           and requested priority is also high,
                                <1>       ;           there is nothing to do! Because it has been
                                <1>       ;           done already (before this attempt).
                                <1>       ;        b) If present priority is high ([u.pri]=2)
                                <1>       ;           and requested priority is not high, there is
                                <1>       ;           nothing to do! Because, it's current
                                <1>       ;           run queue is unspecified, here. (It may be in
                                <1>       ;           a waiting list or in a run queue; if the new
                                <1>       ;           priority would be used to add it to relavant
                                <1>         ;             run queue, this would be wrong, unnecessary
                                <1>       ;           and destabilizing duplication!)
                                <1>       ;        c) If present priority is not high ([u.pri]<2)
                                <1>         ;             and requested priority is high (event),
                                <1>       ;           process will be added to present priority's
                                <1>       ;           run queue and then, priority will be changed
                                <1>       ;           to high ([u.pri]=2).
                                <1>       ;        d) If present priority is not high ([u.pri]<2)
                                <1>       ;           and requested priority is not high, [u.pri]
                                <1>       ;           value will be changed. There is nothing to do
                                <1>       ;           in addition. (The new priority value will be
```

```
                                        <1>       ;           used by 'tswap/tswitch' procedure at 'sysret'
                                        <1>       ;           or 'sysrele' stage.)
                                        <1>       ;
                                        <1>       ;     2) If the requested process is not running:
                                        <1>       ;        a) If requested priority of the requested
                                        <1>       ;           (next) process is high (event) and priority
                                        <1>       ;           of present process is not high, the requested
                                        <1>       ;           process will be added to ('runq_event') high
                                        <1>       ;           priority run queue and then present (running)
                                        <1>       ;           process will be stopped (swapped/switched out)
                                        <1>       ;           immediately if it is in user mode, or it's
                                        <1>       ;           [u.quant] value will be reset to 0 and (then)
                                        <1>       ;           it will be stopped at 'sysret' stage.
                                        <1>       ;        b) If requested priority of the requested
                                        <1>       ;           (next) process is high (event) and priority
                                        <1>       ;           of present process is also high, the requested
                                        <1>       ;           process will be added to ('runq_event') high
                                        <1>       ;           priority run queue and present (running)
                                        <1>       ;           process will be allowed to run until it's
                                        <1>       ;           time quantum will be elapsed ([u.quant]=0).
                                        <1>       ;        c) If requested priority of the requested
                                        <1>       ;           (next) process is not high ('run for event'),
                                        <1>       ;           there is nothing to do. Because, it's current
                                        <1>       ;           run queue is unspecified, here. (It may be in
                                        <1>       ;           a waiting list or in a run queue; if the new
                                        <1>       ;           priority would be used to add it to relavant
                                        <1>         ;             run queue, this would be wrong, unnecessary
                                        <1>       ;           and destabilizing duplication!)
                                        <1>       ;
                                        <1>       ; OUTPUT ->
                                        <1>       ;     none
                                        <1>       ;
                                        <1>       ;     [u.pri] = priority of present process
                                        <1>       ;
                                        <1>       ;     cf = 1, if the request could not be fulfilled.
                                        <1>       ;
                                        <1>       ;     NOTE:
                                        <1>         ;          * Processes in 'run as regular' queue can run
                                        <1>       ;           if there is no process in 'run for event' queue
                                        <1>       ;           ('run for event' processes have higher priority)
                                        <1>       ;        * When [u.quant] time quantum of a process is
                                        <1>       ;           elapsed, it's high priority ('run for event')
                                        <1>       ;           status will be disabled, it can be run in sequence
                                        <1>       ;           of it's actual run queue.
                                        <1>       ;        * A 'run on background' process will always be
                                        <1>       ;           sequenced in 'run on background' (low priority)
                                        <1>       ;           queue, it can run only when other priority queues
                                        <1>       ;           are empty. (idle time processes, e.g. printing)
                                        <1>       ;
                                        <1>       ; Modified registers: eax, ebx, edx
                                        <1>       ;
                                        <1>
                                        <1> srunseq_0:
0000F2C5 3A05[B3030300]                 <1>         cmp     al, [u.uno]     ; same process ?
0000F2CB 750C                           <1>       jne   short srunseq_2 ; no
                                        <1>
0000F2CD 8A25[A9030300]                 <1>       mov   ah, [u.pri]  ; present/current priority
0000F2D3 80FC02                         <1>       cmp   ah, 2        ; 'run for event' priority level
0000F2D6 7221                           <1>       jb    short srunseq_6 ; no
                                        <1>
                                        <1> srunseq_1:
                                        <1>       ; there is nothing to do!
0000F2D8 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
0000F2D9 80FA02                         <1>       cmp   dl, 2        ; event queue
0000F2DC 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
0000F2DE 3A15[A9030300]                 <1>       cmp   dl, [u.pri]  ; priority of present process
0000F2E4 7606                           <1>       jna   short srunseq_3 ; is high, also
                                        <1>       ;
                                        <1>       ; present process will be swapped/switched out
0000F2E6 FE05[955F0100]                 <1>       inc   byte [p_change] ; 1
                                        <1>
                                        <1> srunseq_3:
                                        <1>       ; add process to 'runq_event' queue for new event
0000F2EC BB[52030300]                   <1>       mov   ebx, runq_event ; high priority run queue
                                        <1>
                                        <1> srunseq_4:
                                        <1>       ; al = process number
                                        <1>       ; ebx = run queue
0000F2F1 E8A5F4FFFF                     <1>       call  putlu
0000F2F6 C3                             <1>       retn
                                        <1>
                                        <1> srunseq_5:
0000F2F7 F5                             <1>       cmc
0000F2F8 C3                             <1>       retn
                                        <1>
                                        <1> srunseq_6:
                                        <1>       ; present priority of the process is not high
                                        <1>
0000F2F9 8815[A9030300]                 <1>       mov   [u.pri], dl ; new priority
```

```
                                <1>                     ; (will be used by 'tswap')
                                <1>
0000F2FF 80FA02                 <1>     cmp     dl, 2           ; high priority ?
0000F302 72F3                   <1>     jb      short srunseq_5 ; no, there is nothing to do
                                <1>                             ; in addition
                                <1>
                                <1>     ; process must be added to relevant run queue, here!
                                <1>     ; (new priority is high/event priority and process
                                <1>     ; will not be added to a run queue by 'tswap')
                                <1>
0000F304 BB[54030300]           <1>     mov     ebx, runq_normal ; 'run as regular' queue
                                <1>
0000F309 20E4                   <1>     and     ah, ah  ;  previous value of [u.pri]
0000F30B 75E4                   <1>     jnz     short srunseq_4
                                <1>
0000F30D 43                     <1>     inc     ebx
0000F30E 43                     <1>     inc     ebx
                                <1>     ; ebx = runq_background ; 'run on backgroud' queue
                                <1>
0000F30F 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>
0000F311 803D[A8030300]00       <1>     cmp     byte [u.quant], 0
0000F318 772C                   <1>     ja      short clk_1
                                <1>     ;
0000F31A 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.
0000F321 7623                   <1>     jna     short clk_1 ; yes, do not swap out
                                <1>     ;
0000F323 803D[5B030300]FF       <1>     cmp     byte [sysflg], 0FFh ; user or system space ?
0000F32A 7520                   <1>     jne     short clk_2       ; system space (sysflg <> 0FFh)
                                <1>     ;
0000F32C 66833D[AA030300]00     <1>     cmp     word [u.intr], 0
0000F334 7616                   <1>     jna     short clk_2
                                <1>     ;
                                <1>     ; 23/05/2016
0000F336 803D[965F0100]00       <1>     cmp     byte [multi_tasking], 0
0000F33D 760D                   <1>     jna     short clk_2
                                <1>     ;
0000F33F FE05[955F0100]         <1>     inc     byte [p_change] ; it is time to change running process
0000F345 C3                     <1>     retn
                                <1> clk_1:
0000F346 FE0D[A8030300]         <1>     dec     byte [u.quant]
                                <1> clk_2:
0000F34C 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
0000F34D 80FC07                 <1>     cmp    ah, 7
0000F350 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
0000F352 FB             <1>       sti
                        <1>
                        <1>       ;sti   ; enable interrupts
0000F353 80642408FE     <1>       and    byte [esp+8], 11111110b   ; clear carry bit of eflags register
                        <1>
0000F358 80FC01         <1>       cmp    ah, 1
0000F35B 7205           <1>       jb     short int34h_0
0000F35D 7705           <1>       ja     short int34h_1
                        <1>
0000F35F EE             <1>       out    dx, al
                        <1>       ;iretd
0000F360 EB01           <1>       jmp short int34h_iret
                        <1>
                        <1> int34h_0:
0000F362 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
0000F363 CF             <1>       iretd
                        <1>
                        <1> int34h_1:
0000F364 F6C401         <1>       test   ah, 1
0000F367 7516           <1>       jnz    short int34h_3 ; out
                        <1>
                        <1>       ; in
0000F369 80FC02         <1>       cmp    ah, 2
0000F36C 7707           <1>       ja     short int34h_2
                        <1>
0000F36E 6689D8         <1>       mov    ax, bx
0000F371 66ED           <1>       in     ax, dx
                        <1>       ;iretd
0000F373 EBEE           <1>       jmp    short int34h_iret
                        <1>
                        <1> int34h_2:
0000F375 80FC04         <1>       cmp    ah, 4
0000F378 772C           <1>       ja     short int34h_7      ; 12/10/2017
                        <1>       ; ah = 4
0000F37A 89D8           <1>       mov    eax, ebx
0000F37C ED             <1>       in     eax, dx
                        <1>       ;iretd
0000F37D EBE4           <1>       jmp    short int34h_iret
                        <1>
                        <1> int34h_3:
0000F37F 80FC03         <1>       cmp    ah, 3
0000F382 7707           <1>       ja     short int34h_4
                        <1>
0000F384 6689D8         <1>       mov    ax, bx
0000F387 66EF           <1>       out    dx, ax
                        <1>       ;iretd
0000F389 EBD8           <1>       jmp    short int34h_iret
                        <1>
                        <1> int34h_4:
0000F38B 80FC05         <1>       cmp    ah, 5
0000F38E 770B           <1>       ja     short int34h_6      ; 12/10/2017
                        <1>       ; ah = 5
0000F390 89D8           <1>       mov    eax, ebx
0000F392 EF             <1>       out    dx, eax
                        <1>       ;iretd
0000F393 EBCE           <1>       jmp    short int34h_iret
                        <1>
                        <1> int34h_5:
0000F395 804C240801     <1>       or     byte [esp+8], 1     ; set carry bit of eflags register
0000F39A CF             <1>       iretd
                        <1>
                        <1>       ; 12/10/2017
                        <1> int34h_6:
0000F39B 6689D8         <1>       mov    ax, bx
0000F39E EE             <1>       out    dx, al
0000F39F EB00           <1>       jmp    short $+2
0000F3A1 86E0           <1>       xchg   ah, al
0000F3A3 EE             <1>       out    dx, al
                        <1>       ;xchg  al, ah
                        <1>       ;iretd
0000F3A4 EB06           <1>       jmp    short int34h_8
                        <1> int34h_7:
0000F3A6 EC             <1>       in     al, dx
0000F3A7 EB00           <1>       jmp    short $+2
0000F3A9 88C4           <1>       mov    ah, al
0000F3AB EC             <1>       in     al, dx
                        <1> int34h_8:
0000F3AC 86C4           <1>       xchg   al, ah
0000F3AE CF             <1>       iretd
                        <1>
                        <1>
                        <1> INT4Ah:
                        <1>       ; 24/01/2016
                        <1>       ; this procedure will be called by 'RTC_INT' (in 'timer.s')
0000F3AF 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>       ;
0000F3B0 890424         <1>       mov   [esp], eax ; overwrite call return address
                        <1>       ;push  eax
0000F3B3 66B80900       <1>       mov   ax, 9
0000F3B7 EB07           <1>       jmp   short comm_int
                        <1> com1_int:
                        <1>       ; 07/11/2015
                        <1>       ; 24/10/2015
0000F3B9 890424         <1>       mov   [esp], eax ; overwrite call return address
                        <1>       ; 23/10/2015
                        <1>       ;push  eax
0000F3BC 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
0000F3C0 53             <1>       push  ebx
0000F3C1 56             <1>       push  esi
0000F3C2 57             <1>       push  edi
0000F3C3 1E             <1>       push  ds
0000F3C4 06             <1>       push  es
                        <1>       ; 18/11/2015
0000F3C5 0F20DB         <1>       mov   ebx, cr3
0000F3C8 53             <1>       push  ebx ; ****
                        <1>       ;
0000F3C9 51             <1>       push  ecx ; ***
0000F3CA 52             <1>       push  edx ; **
                        <1>       ;
0000F3CB BB10000000     <1>       mov   ebx, KDATA
0000F3D0 8EDB           <1>       mov   ds, bx
0000F3D2 8EC3           <1>       mov   es, bx
                        <1>       ;
0000F3D4 8B0D[00520100] <1>       mov   ecx, [k_page_dir]
0000F3DA 0F22D9         <1>       mov   cr3, ecx
                        <1>       ; 20/11/2015
                        <1>       ; Interrupt identification register
0000F3DD 66BAFA02       <1>       mov   dx, 2FAh ; COM2
                        <1>       ;
0000F3E1 3C08           <1>       cmp   al, 8
0000F3E3 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>       ;
0000F3E5 FEC6           <1>       inc   dh ; 3FAh ; COM1 Interrupt id. register
                        <1> com_i0:
                        <1>       ;push  eax ; *
                        <1>       ; 07/11/2015
0000F3E7 A2[6A520100]   <1>       mov   byte [ccomport], al
                        <1>       ; 09/11/2015
0000F3EC 0FB7D8         <1>       movzx  ebx, ax ; 8 or 9
                        <1>       ; 17/11/2015
                        <1>       ; reset request for response status
0000F3EF 88A3[60520100] <1>       mov   [ebx+req_resp-8], ah ; 0
                        <1>       ;
                        <1>       ; 20/11/2015
0000F3F5 EC             <1>       in    al, dx       ; read interrupt id. register
0000F3F6 EB00           <1>       JMP   $+2          ; I/O DELAY
0000F3F8 2404           <1>       and   al, 4        ; received data available?
0000F3FA 7470           <1>       jz    short com_eoi; (transmit. holding reg. empty)
                        <1>       ;
                        <1>       ; 20/11/2015
0000F3FC 80EA02         <1>       sub   dl, 3FAh-3F8h; data register (3F8h, 2F8h)
0000F3FF EC             <1>       in    al, dx       ; read character
                        <1>       ;JMP  $+2          ; I/O DELAY
                        <1>       ; 08/11/2015
                        <1>       ; 07/11/2015
0000F400 89DE           <1>       mov   esi, ebx
0000F402 89DF           <1>       mov   edi, ebx
0000F404 81C6[64520100] <1>       add   esi, rchar - 8 ; points to last received char
0000F40A 81C7[66520100] <1>       add   edi, schar - 8 ; points to last sent char
0000F410 8806           <1>       mov   [esi], al ; received char (current char)
                        <1>       ; query
0000F412 20C0           <1>       and   al, al
0000F414 7527           <1>       jnz   short com_i2
                        <1>       ; response
                        <1>       ; 17/11/2015
                        <1>       ; set request for response status
0000F416 FE83[60520100] <1>         inc     byte [ebx+req_resp-8] ; 1
                        <1>       ;
0000F41C 6683C205       <1>       add   dx, 3FDh-3F8h; (3FDh, 2FDh)
0000F420 EC             <1>       in    al, dx       ; read line status register
0000F421 EB00           <1>       JMP   $+2          ; I/O DELAY
0000F423 2420           <1>       and   al, 20h            ; transmitter holding reg. empty?
0000F425 7445           <1>       jz    short com_eoi      ; no
0000F427 B0FF           <1>       mov   al, 0FFh     ; response
```

```
0000F429 6683EA05           <1>       sub    dx, 3FDh-3F8h       ; data port (3F8h, 2F8h)
0000F42D EE                 <1>       out    dx, al       ; send on serial port
                            <1>       ; 17/11/2015
0000F42E 803F00             <1>       cmp    byte [edi], 0   ; query ? (schar)
0000F431 7502               <1>       jne    short com_i1    ; no
0000F433 8807               <1>       mov    [edi], al    ; 0FFh (responded)
                            <1> com_i1:
                            <1>       ; 17/11/2015
                            <1>       ; reset request for response status (again)
0000F435 FE8B[60520100]     <1>         dec     byte [ebx+req_resp-8] ; 0
0000F43B EB2F               <1>       jmp    short com_eoi
                            <1> com_i2:
                            <1>       ; 08/11/2015
0000F43D 3CFF               <1>       cmp    al, 0FFh     ; (response ?)
0000F43F 7417               <1>       je     short com_i3 ; (check for response signal)
                            <1>       ; 07/11/2015
0000F441 3C04               <1>       cmp    al, 04h      ; EOT
0000F443 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)
0000F445 861D[2E520100]     <1>       xchg   bl, [ptty]  ; tty number (8 or 9)
0000F44B E8866FFFFF         <1>       call   ctrlbrk
0000F450 861D[2E520100]     <1>       xchg   [ptty], bl ; (restore ptty value and BL value)
                            <1>       ;mov   al, 04h ; EOT
                            <1>       ; 08/11/2015
0000F456 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
0000F458 803F00             <1>         cmp     byte [edi], 0 ; query ? (schar)
0000F45B 7704               <1>       ja     short com_i4 ; no
                            <1>       ; reset query status (schar)
0000F45D 8807               <1>       mov    [edi], al ; 0FFh
0000F45F FEC0               <1>       inc    al ; 0
                            <1> com_i4:
                            <1>       ; 27/07/2014
                            <1>       ; 09/07/2014
0000F461 D0E3               <1>       shl    bl, 1
0000F463 81C3[30520100]     <1>       add    ebx, ttychr
                            <1>       ; 23/07/2014 (always overwrite)
                            <1>       ;;cmp  word [ebx], 0
                            <1>       ;;ja   short com_eoi
                            <1>       ;
0000F469 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 ; *
0000F46C A0[6A520100]       <1>       mov    al, byte [ccomport] ; current COM port
                            <1>        ; al = tty number (8 or 9)
0000F471 E85E010000         <1>         call wakeup
                            <1> com_iret:
                            <1>       ; 23/10/2015
0000F476 5A                 <1>       pop    edx ; **
0000F477 59                 <1>       pop    ecx ; ***
                            <1>       ; 18/11/2015
                            <1>       ;pop   eax ; ****
                            <1>       ;mov   cr3, eax
                            <1>       ;jmp   iiret
0000F478 E96316FFFF         <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>       ;
0000F47D BB[66520100]   <1>       mov    ebx, com1p          ; COM1 parameters
0000F482 66BAF803       <1>       mov    dx, 3F8h            ; COM1
                        <1>        ; 29/10/2015
0000F486 66B90103       <1>       mov    cx, 301h  ; divisor = 1 (115200 baud)
0000F48A E86F000000     <1>       call   sp_i3  ; call A4
0000F48F A880           <1>       test   al, 80h
0000F491 7410           <1>       jz     short sp_i0 ; OK..
                        <1>              ; Error !
                        <1>       ;mov   dx, 3F8h
0000F493 80EA05         <1>       sub    dl, 5 ; 3FDh -> 3F8h
0000F496 66B90E03       <1>       mov    cx, 30Eh  ; divisor = 12 (9600 baud)
0000F49A E85F000000     <1>       call   sp_i3  ; call A4
0000F49F A880           <1>       test   al, 80h
0000F4A1 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>       ;
0000F4A3 C603E3         <1>       mov    byte [ebx], 0E3h ; 11100011b
0000F4A6 E8DC000000     <1>       call   sp_i5 ; 29/06/2015
                        <1> sp_i1:
0000F4AB 43             <1>       inc    ebx
0000F4AC 66BAF802       <1>       mov    dx, 2F8h            ; COM2
                        <1>        ; 29/10/2015
0000F4B0 66B90103       <1>       mov    cx, 301h  ; divisor = 1 (115200 baud)
0000F4B4 E845000000     <1>       call   sp_i3  ; call A4
0000F4B9 A880           <1>       test   al, 80h
0000F4BB 7410           <1>       jz     short sp_i2 ; OK..
                        <1>              ; Error !
                        <1>       ;mov   dx, 2F8h
0000F4BD 80EA05         <1>       sub    dl, 5 ; 2FDh -> 2F8h
0000F4C0 66B90E03       <1>       mov    cx, 30Eh  ; divisor = 12 (9600 baud)
0000F4C4 E835000000     <1>       call   sp_i3  ; call A4
0000F4C9 A880           <1>       test   al, 80h
0000F4CB 7530           <1>       jnz    short sp_i7
                        <1> sp_i2:
0000F4CD C603E3         <1>       mov    byte [ebx], 0E3h ; 11100011b
                        <1> sp_i6:
                        <1>       ;; COM2 - enabling IRQ 3
                        <1>       ; 07/11/2015
                        <1>       ; 26/10/2015
0000F4D0 9C             <1>       pushf
0000F4D1 FA             <1>       cli
                        <1>       ;
0000F4D2 66BAFC02       <1>       mov    dx, 2FCh            ; modem control register
0000F4D6 EC             <1>       in     al, dx                   ; read register
0000F4D7 EB00           <1>       JMP    $+2                 ; I/O DELAY
0000F4D9 0C08           <1>       or     al, 8               ; enable bit 3 (OUT2)
0000F4DB EE             <1>       out    dx, al              ; write back to register
0000F4DC EB00           <1>       JMP    $+2                 ; I/O DELAY
0000F4DE 66BAF902       <1>       mov    dx, 2F9h            ; interrupt enable register
0000F4E2 EC             <1>       in     al, dx              ; read register
0000F4E3 EB00           <1>       JMP    $+2                 ; I/O DELAY
                        <1>       ;or    al, 1               ; receiver data interrupt enable and
0000F4E5 0C03           <1>       or     al, 3               ; transmitter empty interrupt enable
0000F4E7 EE             <1>       out    dx, al                   ; write back to register
0000F4E8 EB00           <1>       JMP    $+2                 ; I/O DELAY
0000F4EA E421           <1>       in     al, 21h             ; read interrupt mask register
0000F4EC EB00           <1>       JMP    $+2                 ; I/O DELAY
0000F4EE 24F7           <1>       and    al, 0F7h            ; enable IRQ 3 (COM2)
0000F4F0 E621           <1>       out    21h, al             ; write back to register
                        <1>       ;
                        <1>       ; 23/10/2015
0000F4F2 B8[B0F30000]   <1>       mov    eax, com2_int
0000F4F7 A3[CFF50000]   <1>       mov    [com2_irq3], eax
                        <1>       ; 26/10/2015
0000F4FC 9D             <1>       popf
                        <1> sp_i7:
0000F4FD C3             <1>       retn
                        <1>
                        <1> sp_i3:
                        <1> ;A4:        ;----- INITIALIZE THE COMMUNICATIONS PORT
                        <1>       ; 28/10/2015
```

```
0000F4FE FEC2              <1>       inc    dl     ; 3F9h (2F9h); 3F9h, COM1 Interrupt enable register
0000F500 B000              <1>       mov    al, 0
0000F502 EE                <1>       out    dx, al              ; disable serial port interrupt
0000F503 EB00              <1>       JMP    $+2                 ; I/O DELAY
0000F505 80C202            <1>       add    dl, 2  ; 3FBh (2FBh); COM1 Line control register (3FBh)
0000F508 B080              <1>       mov    al, 80h
0000F50A EE                <1>       out    dx, al              ; SET DLAB=1 ; divisor latch access bit
                           <1>       ;----- SET BAUD RATE DIVISOR
                           <1>       ; 26/10/2015
0000F50B 80EA03            <1>       sub    dl, 3   ; 3F8h (2F8h)     ; register for least significant byte
                           <1>                                  ; of the divisor value
0000F50E 88C8              <1>       mov    al, cl ; 1
0000F510 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)
0000F511 EB00              <1>       JMP    $+2                 ; I/O DELAY
0000F513 28C0              <1>       sub    al, al
0000F515 FEC2              <1>       inc    dl      ; 3F9h (2F9h)     ; register for most significant byte
                           <1>                                  ; of the divisor value
0000F517 EE                <1>       out    dx, al ; 0
0000F518 EB00              <1>       JMP    $+2                 ; I/O DELAY
                           <1>       ;
0000F51A 88E8              <1>       mov    al, ch ; 3          ; 8 data bits, 1 stop bit, no parity
                           <1>       ;and   al, 1Fh ; Bits 0,1,2,3,4
0000F51C 80C202            <1>       add    dl, 2  ; 3FBh (2FBh); Line control register
0000F51F EE                <1>       out    dx, al
0000F520 EB00              <1>       JMP    $+2                 ; I/O DELAY
                           <1>       ; 29/10/2015
0000F522 FECA              <1>       dec    dl     ; 3FAh (2FAh); FIFO Control register (16550/16750)
0000F524 30C0              <1>       xor    al, al              ; 0
0000F526 EE                <1>       out    dx, al              ; Disable FIFOs (reset to 8250 mode)
0000F527 EB00              <1>       JMP    $+2
                           <1> sp_i4:
                           <1> ;A18: ;----- COMM PORT STATUS ROUTINE
                           <1>       ; 29/06/2015 (line status after modem status)
0000F529 80C204            <1>       add    dl, 4  ; 3FEh (2FEh); Modem status register
                           <1> sp_i4s:
0000F52C EC                <1>       in     al, dx              ; GET MODEM CONTROL STATUS
0000F52D EB00              <1>       JMP    $+2                 ; I/O DELAY
0000F52F 88C4              <1>       mov    ah, al              ; PUT IN (AH) FOR RETURN
0000F531 FECA              <1>       dec    dl     ; 3FDh (2FDh); POINT TO LINE STATUS REGISTER
                           <1>                                  ; dx = 3FDh for COM1, 2FDh for COM2
0000F533 EC                <1>       in     al, dx              ; GET LINE CONTROL STATUS
                           <1>       ; AL = Line status, AH = Modem status
0000F534 C3                <1>       retn
                           <1>
                           <1> sp_status:
                           <1>       ; 29/06/2015
                           <1>       ; 27/06/2015 (Retro UNIX 386 v1)
                           <1>       ; Get serial port status
0000F535 66BAFE03          <1>       mov    dx, 3FEh            ; Modem status register (COM1)
0000F539 28C6              <1>       sub    dh, al              ; dh = 2 for COM2 (al = 1)
                           <1>                                  ; dx = 2FEh for COM2
0000F53B 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>       ;
0000F53D 66BAF803          <1>       mov    dx, 3F8h
0000F541 BB[66520100]      <1>       mov    ebx, com1p ; COM1 control byte offset
0000F546 3C01              <1>       cmp    al, 1
```

```
774 0000F548 776B                <1>       ja     short sp_invp_err
775 0000F54A 7203                <1>       jb     short sp_setp1 ;  COM1 (AL = 0)
776 0000F54C FECE                <1>       dec    dh ; 2F8h
777 0000F54E 43                  <1>       inc    ebx ; COM2 control byte offset
778                              <1> sp_setp1:
779                              <1>       ; 29/10/2015
780 0000F54F 8823                <1>       mov    [ebx], ah
781 0000F551 0FB6CC              <1>       movzx  ecx, ah
782 0000F554 C0E905              <1>       shr    cl, 5 ; -> baud rate index
783 0000F557 80E41F              <1>       and    ah, 1Fh ; communication parameters except baud rate
784 0000F55A 8A81[C4F50000]      <1>       mov    al, [ecx+b_div_tbl]
785 0000F560 6689C1              <1>       mov    cx, ax
786 0000F563 E896FFFFFF          <1>       call   sp_i3
787 0000F568 6689C1              <1>       mov    cx, ax ; CL = Line status, CH = Modem status
788 0000F56B A880                <1>       test   al, 80h
789 0000F56D 740F                <1>       jz     short sp_setp2
790 0000F56F C603E3              <1>         mov     byte [ebx], 0E3h ; Reset to initial value (11100011b)
791                              <1> stp_dnr_err:
792 0000F572 C705[C8030300]0F00- <1>       mov    dword [u.error], ERR_DEV_NOT_RDY ; 'device not ready !'
792 0000F57A 0000                <1>
793                              <1>       ; CL = Line status, CH = Modem status
794 0000F57C F9                  <1>       stc
795 0000F57D C3                  <1>       retn
796                              <1> sp_setp2:
797 0000F57E 80FE02              <1>       cmp    dh, 2 ; COM2 (2F?h)
798 0000F581 0F8649FFFFFF        <1>         jna     sp_i6
799                              <1>                   ; COM1 (3F?h)
800                              <1> sp_i5:
801                              <1>       ; 07/11/2015
802                              <1>       ; 26/10/2015
803                              <1>       ; 29/06/2015
804                              <1>       ;
805                              <1>       ;; COM1 - enabling IRQ 4
806 0000F587 9C                  <1>       pushf
807 0000F588 FA                  <1>       cli
808 0000F589 66BAFC03            <1>       mov    dx, 3FCh            ; modem control register
809 0000F58D EC                  <1>       in     al, dx                   ; read register
810 0000F58E EB00                <1>       JMP    $+2                 ; I/O DELAY
811 0000F590 0C08                <1>       or     al, 8               ; enable bit 3 (OUT2)
812 0000F592 EE                  <1>       out    dx, al              ; write back to register
813 0000F593 EB00                <1>       JMP    $+2                 ; I/O DELAY
814 0000F595 66BAF903            <1>       mov    dx, 3F9h            ; interrupt enable register
815 0000F599 EC                  <1>       in     al, dx              ; read register
816 0000F59A EB00                <1>       JMP    $+2                 ; I/O DELAY
817                              <1>       ;or    al, 1               ; receiver data interrupt enable and
818 0000F59C 0C03                <1>       or     al, 3               ; transmitter empty interrupt enable
819 0000F59E EE                  <1>       out    dx, al                   ; write back to register
820 0000F59F EB00                <1>       JMP    $+2                 ; I/O DELAY
821 0000F5A1 E421                <1>       in     al, 21h             ; read interrupt mask register
822 0000F5A3 EB00                <1>       JMP    $+2                 ; I/O DELAY
823 0000F5A5 24EF                <1>       and    al, 0EFh            ; enable IRQ 4 (COM1)
824 0000F5A7 E621                <1>       out    21h, al             ; write back to register
825                              <1>       ;
826                              <1>       ; 23/10/2015
827 0000F5A9 B8[B9F30000]        <1>       mov    eax, com1_int
828 0000F5AE A3[CBF50000]        <1>       mov    [com1_irq4], eax
829                              <1>       ; 26/10/2015
830 0000F5B3 9D                  <1>       popf
831 0000F5B4 C3                  <1>       retn
832                              <1>
833                              <1> sp_invp_err:
834 0000F5B5 C705[C8030300]1700- <1>       mov    dword [u.error], ERR_INV_PARAMETER ; 'invalid parameter !'
834 0000F5BD 0000                <1>
835 0000F5BF 31C9                <1>       xor    ecx, ecx
836 0000F5C1 49                  <1>       dec    ecx ; 0FFFFh
837 0000F5C2 F9                  <1>       stc
838 0000F5C3 C3                  <1>       retn
839                              <1>
840                              <1> ; 29/10/2015
841                              <1> b_div_tbl: ; Baud rate divisor table (115200/divisor)
842 0000F5C4 010C0603080401      <1>       db 1, 12, 6, 3, 8, 4, 1
843                              <1>
844                              <1>
845                              <1> ; 23/10/2015
846                              <1> com1_irq4:
847 0000F5CB [D3F50000]          <1>       dd dummy_retn
848                              <1> com2_irq3:
849 0000F5CF [D3F50000]          <1>       dd dummy_retn
850                              <1>
851                              <1> dummy_retn:
852 0000F5D3 C3                  <1>       retn
853                              <1>
854                              <1> wakeup:
855                              <1>       ; 24/01/2016
856 0000F5D4 C3                  <1>       retn
857                              <1>
858                              <1> set_working_path_x:
859                              <1>             ; 17/10/2016 (TRDOS 386 - FFF & FNF)
860 0000F5D5 66B80100            <1>             mov    ax, 1
861                              <1>                    ; File name is needed/forced (AL=1)
862                              <1>                    ; Change directory as temprorary (AH=0)
863                              <1>
864                              <1>             ; This is needed for preventing wrong Find Next File
865                              <1>             ; system call after sysopen, syscreate, sysmkdir etc.
866                              <1>             ; Find Next File must immmediate follow Find First file)
867                              <1>
868 0000F5D9 8825[B85F0100]      <1>             mov    [FFF_Valid], ah ; 0 ; reset ; 17/10/2016
869                              <1>
870                              <1> set_working_path:
871                              <1>             ; 16/10/2016
872                              <1>             ; 12/10/2016
873                              <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>
0000F5DF 66A3[BC5F0100]         <1>             mov    [SWP_Mode], ax
0000F5E5 A0[C6520100]           <1>             mov    al, [Current_Drv]
0000F5EA 30E4                   <1>             xor    ah, ah
0000F5EC 66A3[BE5F0100]         <1>             mov    word [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>
0000F5F2 55                     <1>             push   ebp
0000F5F3 89E5                   <1>             mov    ebp, esp
                                <1>
0000F5F5 B980000000             <1>             mov    ecx, 128 ; maximum path length = 128 bytes
0000F5FA 29CC                   <1>             sub    esp, ecx ; reserve 128 bytes (buffer) on stack
0000F5FC 89E7                   <1>             mov    edi, esp ; destination address (kernel space)
                                <1>             ; esi = source address (virtual, in user's memory space)
0000F5FE E8CDF2FFFF             <1>             call   transfer_from_user_buffer
0000F603 720A                   <1>             jc     short loc_swp_xor_retn
                                <1>
0000F605 89E6                   <1>             mov    esi, esp ; temporary buffer (the path) on stack
                                <1> loc_swp_fchar:
0000F607 8A06                   <1>             mov    al, [esi]
0000F609 3C20                   <1>             cmp    al, 20h
0000F60B 7711                   <1>             ja     short loc_swp_parse_path_name
0000F60D 740C                   <1>             je     short loc_swp_fchar_next
                                <1>
                                <1> loc_swp_xor_retn:
0000F60F 31C0                   <1>             xor    eax, eax
0000F611 F9                     <1>             stc
                                <1> loc_swp_retn:
0000F612 89EC                   <1>             mov    esp, ebp
0000F614 5D                     <1>             pop    ebp
                                <1>
                                <1>             ;mov   esi, FindFile_Drv
0000F615 BE[AC5C0100]           <1>             mov    esi, FindFile_Name ; 12/10/2016
0000F61A C3                     <1>             retn
                                <1>
                                <1> loc_swp_fchar_next:
0000F61B 46                     <1>             inc    esi
0000F61C EBE9                   <1>             jmp    short loc_swp_fchar
                                <1>
                                <1> loc_swp_parse_path_name:
0000F61E BF[6A5C0100]           <1>             mov    edi, FindFile_Drv
0000F623 E8DEABFFFF             <1>             call   parse_path_name
0000F628 72E8                   <1>             jc     short loc_swp_retn
                                <1>
                                <1> loc_swp_checkfile_name:
0000F62A 803D[BC5F0100]00       <1>             cmp    byte [SWP_Mode], 0
0000F631 761E                   <1>             jna    short loc_swp_drv
                                <1>
                                <1>             ; 10/10/2016 (valid file name checking)
0000F633 BE[AC5C0100]           <1>             mov    esi, FindFile_Name
0000F638 803E20                 <1>             cmp    byte [esi], 20h
0000F63B 76D2                   <1>             jna    short loc_swp_xor_retn
                                <1>
                                <1>             ; 16/10/2016
0000F63D C605[BB5F0100]00       <1>             mov    byte [SWP_inv_fname], 0 ; reset
```

```
                                <1>             ; esi = file name address (ASCIIZ)
0000F644 E8668DFFFF             <1>             call   check_filename
0000F649 7306                   <1>             jnc    short loc_swp_drv
                                <1>
0000F64B FE05[BB5F0100]         <1>             inc    byte [SWP_inv_fname] ; set
                                <1> loc_swp_drv:
0000F651 8A35[C6520100]         <1>             mov    dh, [Current_Drv]
                                <1>                ;mov       [RUN_CDRV], dh
                                <1>
0000F657 8A15[6A5C0100]         <1>             mov    dl, [FindFile_Drv]
                                <1>                ;cmp       dl, dh
0000F65D 3A15[C6520100]         <1>             cmp    dl, [Current_Drv]
0000F663 740D                   <1>             je     short loc_swp_change_directory
                                <1>
0000F665 FE05[BF5F0100]         <1>             inc    byte [SWP_DRV_chg]
0000F66B E8ED75FFFF             <1>             call   change_current_drive
0000F670 72A0                   <1>             jc     short loc_swp_retn ; eax = error code
                                <1>             ; eax = 0
                                <1>
                                <1> loc_swp_change_directory:
0000F672 803D[6B5C0100]21       <1>             cmp    byte [FindFile_Directory], 21h
0000F679 F5                     <1>             cmc
0000F67A 7396                   <1>             jnc    short loc_swp_retn
                                <1>
0000F67C FE05[BF5F0100]         <1>             inc    byte [SWP_DRV_chg]
0000F682 FE05[AD060100]         <1>             inc    byte [Restore_CDIR]
0000F688 BE[6B5C0100]           <1>             mov    esi, FindFile_Directory
0000F68D 8A25[BD5F0100]         <1>             mov    ah, [SWP_Mode+1]
0000F693 E85AA5FFFF             <1>             call   change_current_directory
0000F698 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
0000F69E E874A4FFFF             <1>             call   change_prompt_dir_str
0000F6A3 29C0                   <1>             sub    eax, eax ; 0
0000F6A5 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>
0000F6AA 31C0                   <1>             xor    eax, eax
0000F6AC 48                     <1>             dec    eax
                                <1>
0000F6AD 668B15[BE5F0100]       <1>             mov    dx, [SWP_DRV]
0000F6B4 08F6                   <1>             or     dh, dh
0000F6B6 742E                   <1>             jz     short loc_rwp_return
                                <1>
0000F6B8 3A15[C6520100]         <1>             cmp    dl, [Current_Drv]
0000F6BE 7407                   <1>             je     short loc_rwp_restore_cdir
                                <1> loc_rwp_restore_cdrv:
0000F6C0 E89875FFFF             <1>             call   change_current_drive
0000F6C5 EB10                   <1>             jmp    short loc_rwp_restore_ok
                                <1> loc_rwp_restore_cdir:
0000F6C7 31DB                   <1>             xor    ebx, ebx
0000F6C9 88D7                   <1>             mov    bh, dl
0000F6CB BE00010900             <1>             mov    esi, Logical_DOSDisks
0000F6D0 01DE                   <1>             add    esi, ebx
                                <1>
0000F6D2 E83D76FFFF             <1>             call   restore_current_directory
                                <1>
                                <1> loc_rwp_restore_ok:
0000F6D7 668B15[BE5F0100]       <1>             mov    dx, [SWP_DRV]
0000F6DE 31C0                   <1>             xor    eax, eax
0000F6E0 66A3[BF5F0100]         <1>             mov    [SWP_DRV_chg], ax
                                <1> loc_rwp_return:
0000F6E6 C3                     <1>             retn
                                <1>
                                <1> get_file_name:
                                <1>             ; 15/10/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>             ; Convert file name
                                <1>             ;      from directory entry format
                                <1>                ;  to (8.3) dot file name format
                                <1>             ;
                                <1>             ; TRDOS v1.0 (DIR.ASM, "get_file_name")
                                <1>                ; 2005 - 09/10/2011
                                <1>             ; INPUT:
                                <1>             ;      DS:SI -> Directory Entry Format File Name
                                <1>             ;       ES:DI -> DOS Dot File Name Address
                                <1>             ; OUTPUT:
                                <1>             ;      DS:SI -> DOS Dot File Name Address
                                <1>                ;  ES:DI -> Directory Entry Format File Name
                                <1>             ;
                                <1>             ; TRDOS 386 (15/10/2016)
                                <1>             ; INPUT:
                                <1>             ;      ESI = File name addr in dir entry format
                                <1>             ;      EDI = Dot file name address (destination)
```

```
                                <1>             ; OUTPUT:
                                <1>             ;     File name is converted and moved
                                <1>             ;     to destination (as 8.3 dot filename)
                                <1>             ;
                                <1>             ; Modified registers: EAX, ECX
                                <1>
                                <1>                ; 2005 (TRDOS 8086) - 2016 (TRDOS 386)
                                <1>
0000F6E7 57                     <1>             push   edi
0000F6E8 56                     <1>             push   esi
0000F6E9 AC                     <1>             lodsb
0000F6EA 3C20                   <1>             cmp    al, 20h
0000F6EC 762A                   <1>             jna    short pass_gfn_ext
0000F6EE 56                     <1>             push   esi
0000F6EF AA                     <1>             stosb
0000F6F0 B907000000             <1>             mov    ecx, 7
                                <1> loc_gfn_next_char:
0000F6F5 AC                     <1>             lodsb
0000F6F6 3C20                   <1>             cmp    al, 20h
0000F6F8 7603                   <1>             jna    short pass_gfn_fn
0000F6FA AA                     <1>             stosb
0000F6FB E2F8                   <1>             loop   loc_gfn_next_char
                                <1> pass_gfn_fn:
0000F6FD 5E                     <1>             pop    esi
0000F6FE 83C607                 <1>             add    esi, 7
0000F701 AC                     <1>             lodsb
0000F702 3C20                   <1>             cmp    al, 20h
0000F704 7612                   <1>             jna    short pass_gfn_ext
0000F706 B42E                   <1>             mov    ah, '.'
0000F708 86E0                   <1>             xchg   ah, al
0000F70A 66AB                   <1>             stosw
0000F70C AC                     <1>             lodsb
0000F70D 3C20                   <1>             cmp    al, 20h
0000F70F 7607                   <1>             jna    short pass_gfn_ext
0000F711 AA                     <1>             stosb
0000F712 AC                     <1>             lodsb
0000F713 3C20                   <1>             cmp    al, 20h
0000F715 7601                   <1>             jna    short pass_gfn_ext
0000F717 AA                     <1>             stosb
                                <1> pass_gfn_ext:
0000F718 30C0                   <1>             xor    al, al
0000F71A AA                     <1>             stosb
0000F71B 5E                     <1>             pop    esi
0000F71C 5F                     <1>             pop    edi
0000F71D C3                     <1>             retn
                                <1>
                                <1> set_hardware_int_vector:
                                <1>             ; 18/03/2017
                                <1>             ; 03/03/2017
                                <1>             ; 28/02/2017 - TRDOS 386 (TRDOS v2.0)
                                <1>             ;
                                <1>             ; SET/RESET HARDWARE INTERRUPT GATE
                                <1>             ;
                                <1>             ; Changes interrupt gate descriptor table
                                <1>             ; (without changing default interrupt list)
                                <1>             ;
                                <1>             ; INPUT:
                                <1>             ;     AL = IRQ number (0 to 15)
                                <1>             ;     AH > 0 -> set
                                <1>             ;     AH = 0 -> reset
                                <1>             ;
                                <1>             ; Modified registers: eax, ebx, edx, edi
                                <1>             ;
                                <1>
0000F71E C0E002                 <1>             shl    al, 2 ; IRQ number * 4
0000F721 0FB6D8                 <1>             movzx  ebx, al
                                <1>
0000F724 08E4                   <1>             or     ah, ah
0000F726 7508                   <1>             jnz    short shintv_1 ; set (for user call service)
                                <1>
                                <1>             ; 18/03/2017
0000F728 81C3[AC100100]         <1>             add    ebx, IRQ_list ; reset to default interrupt list
0000F72E EB06                   <1>             jmp    short shintv_2
                                <1> shintv_1:
0000F730 81C3[57F70000]         <1>             add    ebx, IRQ_u_list
                                <1> shintv_2:
0000F736 8B13                   <1>             mov    edx, [ebx] ; IRQ handler address
                                <1>
                                <1>             ; 03/03/2017
0000F738 D0E0                   <1>             shl    al, 1 ; IRQ number * 8
                                <1>             ; 18/03/2017
0000F73A 0FB6F8                 <1>             movzx  edi, al
0000F73D 81C7[18500100]         <1>             add    edi, idt + (8*32) ; IRQ 0 offset = idt + 256
                                <1>
0000F743 89D0                   <1>             mov    eax, edx ; IRQ handler address
0000F745 BB00000800             <1>             mov    ebx, 80000h
                                <1>
                                <1>             ;mov   edx, eax
0000F74A 66BA008E               <1>             mov    dx, 8E00h
0000F74E 6689C3                 <1>             mov    bx, ax
0000F751 89D8                   <1>             mov    eax, ebx ; /* selector = 0x0008 = cs */
                                <1>                                     ; /* interrupt gate - dpl=0, present */
0000F753 AB                     <1>             stosd  ; selector & offset bits 0-15
0000F754 8917                   <1>             mov    [edi], edx ; attributes & offset bits 16-23
                                <1>
0000F756 C3                     <1>             retn
                                <1> IRQ_u_list:
                                <1>             ; 28/02/2017
0000F757 [8B060000]             <1>             dd     timer_int
0000F75B [FF0D0000]             <1>             dd     kb_int
0000F75F [6D080000]             <1>             dd     irq2
0000F763 [97F70000]             <1>             dd     IRQ_service3
```

```
1180 0000F767 [A1F70000]          <1>             dd    IRQ_service4
1181 0000F76B [ABF70000]          <1>             dd    IRQ_service5
1182 0000F76F [B0410000]          <1>             dd    fdc_int
1183 0000F773 [B5F70000]          <1>             dd    IRQ_service7
1184 0000F777 [F6070000]          <1>             dd    rtc_int
1185 0000F77B [BFF70000]          <1>             dd    IRQ_service9
1186 0000F77F [C9F70000]          <1>             dd    IRQ_service10
1187 0000F783 [D3F70000]          <1>             dd    IRQ_service11
1188 0000F787 [DDF70000]          <1>             dd    IRQ_service12
1189 0000F78B [E7F70000]          <1>             dd    IRQ_service13
1190 0000F78F [2C4B0000]          <1>             dd    hdc1_int
1191 0000F793 [534B0000]          <1>             dd    hdc2_int
1192                              <1>
1193                              <1>             ; 03/03/2017
1194                              <1>             ; 27/02/2017
1195                              <1> IRQ_service3:
1196 0000F797 36C605[82650100]03  <1>             mov   byte [ss:IRQnum], 3
1197 0000F79F EB4E                <1>             jmp   short IRQ_service
1198                              <1> IRQ_service4:
1199 0000F7A1 36C605[82650100]04  <1>             mov   byte [ss:IRQnum], 4
1200 0000F7A9 EB44                <1>             jmp   short IRQ_service
1201                              <1> IRQ_service5:
1202 0000F7AB 36C605[82650100]05  <1>             mov   byte [ss:IRQnum], 5
1203 0000F7B3 EB3A                <1>             jmp   short IRQ_service
1204                              <1> IRQ_service7:
1205 0000F7B5 36C605[82650100]07  <1>             mov   byte [ss:IRQnum], 7
1206 0000F7BD EB30                <1>             jmp   short IRQ_service
1207                              <1> IRQ_service9:
1208 0000F7BF 36C605[82650100]09  <1>             mov   byte [ss:IRQnum], 9
1209 0000F7C7 EB26                <1>             jmp   short IRQ_service
1210                              <1> IRQ_service10:
1211 0000F7C9 36C605[82650100]0A  <1>             mov   byte [ss:IRQnum], 10
1212 0000F7D1 EB1C                <1>             jmp   short IRQ_service
1213                              <1> IRQ_service11:
1214 0000F7D3 36C605[82650100]0B  <1>             mov   byte [ss:IRQnum], 11
1215 0000F7DB EB12                <1>             jmp   short IRQ_service
1216                              <1> IRQ_service12:
1217 0000F7DD 36C605[82650100]0C  <1>             mov   byte [ss:IRQnum], 12
1218 0000F7E5 EB08                <1>             jmp   short IRQ_service
1219                              <1> IRQ_service13:
1220 0000F7E7 36C605[82650100]0D  <1>             mov   byte [ss:IRQnum], 13
1221                              <1>             ;jmp  short IRQ_service
1222                              <1> IRQ_service:
1223                              <1>             ; 13/06/2017
1224                              <1>             ; 11/06/2017
1225                              <1>             ; 10/06/2017
1226                              <1>             ; 01/03/2017, 04/03/2017
1227                              <1>             ; 27/02/2017, 28/02/2017
1228 0000F7EF 1E                  <1>             push  ds
1229 0000F7F0 06                  <1>             push  es
1230 0000F7F1 0FA0                <1>             push  fs
1231 0000F7F3 0FA8                <1>             push  gs
1232                              <1>
1233 0000F7F5 60                  <1>             pushad ; eax,ecx,edx,ebx,esp,ebp,esi,edi
1234 0000F7F6 66B91000            <1>             mov    cx, KDATA
1235 0000F7FA 8ED9                <1>             mov    ds, cx
1236 0000F7FC 8EC1                <1>             mov    es, cx
1237 0000F7FE 8EE1                <1>             mov    fs, cx
1238 0000F800 8EE9                <1>             mov    gs, cx
1239                              <1>
1240 0000F802 0F20D8              <1>             mov   eax, cr3
1241 0000F805 A3[7E650100]        <1>             mov   [IRQ_cr3], eax
1242                              <1>
1243 0000F80A A1[00520100]        <1>             mov   eax, [k_page_dir]
1244 0000F80F 0F22D8              <1>             mov   cr3, eax
1245                              <1>
1246 0000F812 A0[82650100]        <1>             mov   al, [IRQnum]
1247                              <1>
1248                              <1>             ;mov  cl, [sysflg]
1249                              <1>             ;mov  [u.r_mode], cl  ; system (0) or user mode (FFh)
1250                              <1> IRQsrv_0:
1251 0000F817 0FB6D8              <1>             movzx  ebx, al
1252 0000F81A 8A9B[E20F0100]      <1>             mov   bl, [ebx+IRQenum] ; IRQ (available) index number + 1
1253                              <1>             ; 01/03/2017
1254 0000F820 FECB                <1>             dec   bl  ; IRQ index number, 0 to 8
1255 0000F822 0F8807010000        <1>             js    IRQsrv_5 ; not available to use here!?
1256                              <1>             ;
1257 0000F828 80BB[48650100]80    <1>             cmp   byte [ebx+IRQ.method], 80h ; using by a dev or kernel?
1258 0000F82F 7205                <1>             jb    short IRQsrv_1 ; no
1259                              <1>
1260                              <1>             ; If the IRQ service is already owned by TRDOS 386 kernel
1261                              <1>             ;      or a Device driver
1262                              <1>             ; we need to call 'dev_IRQ_service'
1263                              <1>
1264                              <1>             ; IRQ number in AL
1265 0000F831 E868020000            <1>             call  dev_IRQ_service     ; IRQ service for device drivers
1266                              <1>             ; IRQ number in AL
1267                              <1> IRQsrv_1:
1268                              <1>             ; check user callback service status
1269                              <1>             ; AL = IRQ number
1270                              <1>             ; EBX = IRQ (Available) Index number
1271                              <1>
1272 0000F836 A2[D7030300]        <1>             mov   [u.irqwait], al ; set waiting IRQ flag
1273                              <1>
1274 0000F83B 8A83[36650100]      <1>             mov   al, [ebx+IRQ.owner]
1275 0000F841 20C0                <1>             and   al, al
1276 0000F843 0F84E6000000        <1>             jz    IRQsrv_5 ; it is not owned by a user/proc
1277                              <1>
1278                              <1>             ; 03/03/2017
1279 0000F849 89DA                <1>             mov   edx, ebx
1280 0000F84B C0E202              <1>             shl   dl, 2
1281 0000F84E 8B92[5A650100]      <1>             mov   edx, [edx+IRQ.addr] ; S.R.B. or Callback service addr
```

```
1282                                    <1>
1283 0000F854 8AA3[48650100]            <1>             mov    ah, [ebx+IRQ.method]
1284 0000F85A F6C401                    <1>             test   ah, 1
1285 0000F85D 7534                      <1>             jnz    short IRQsrv_4 ; Callback service method
1286                                    <1>
1287                                    <1>             ; Signal Response Byte method
1288                                    <1>             ;mov   edx, [edx+IRQ.addr] ; Signal Response Byte address
1289                                    <1>             ;                       ; (Physical address, non-swappable)
1290 0000F85F 80E402                    <1>             and    ah, 2 ; bit 1, (S.R.B.) counter (auto increment) method
1291 0000F862 8AA3[51650100]            <1>             mov    ah, [ebx+IRQ.srb] ; Signal Response Byte value
1292 0000F868 7408                      <1>             jz     short IRQsrv_2 ; fixed S.R.B. value
1293                                    <1>             ; counter method (auto increment)
1294 0000F86A FEC4                      <1>             inc    ah
1295 0000F86C 88A3[51650100]            <1>             mov    [ebx+IRQ.srb], ah ; next (count) number
1296                                    <1> IRQsrv_2:
1297 0000F872 8822                      <1>             mov    [edx], ah ; put S.R.B. val to the user's S.R.B. addr
1298 0000F874 C605[D7030300]00          <1>             mov    byte [u.irqwait], 0 ; clear waiting IRQ flag
1299                                    <1>
1300 0000F87B 3A05[B3030300]            <1>             cmp    al, [u.uno]
1301 0000F881 0F84A8000000              <1>             je     IRQsrv_5 ; the owner is current user/process
1302                                    <1> IRQsrv_3:
1303                                    <1>             ; the owner is not current user/process
1304                                    <1>             ; AL = process number
1305 0000F887 B202                      <1>             mov    dl, 2 ; priority, 2 = event (high)
1306 0000F889 E837FAFFFF                <1>             call   set_run_sequence
1307                                    <1>
1308                                    <1>             ; [u.irqwait] = waiting IRQ number for callback service
1309                                    <1>
1310 0000F88E E99C000000                <1>             jmp    IRQsrv_5
1311                                    <1> IRQsrv_4:
1312 0000F893 3A05[B3030300]            <1>             cmp    al, [u.uno]  ; is the owner is current user/process?
1313 0000F899 75EC                      <1>             jne    short IRQsrv_3 ; no !
1314                                    <1>
1315                                    <1>             ; Check if an IRQ callback service already in progress
1316 0000F89B 803D[D8030300]00          <1>             cmp    byte [u.r_lock], 0
1317 0000F8A2 0F8787000000              <1>             ja     IRQsrv_5 ; nothing to do !
1318                                    <1>                             ; (we need to complete prev callback)
1319 0000F8A8 803D[D4030300]00          <1>             cmp    byte [u.t_lock], 0
1320 0000F8AF 777E                      <1>             ja     short IRQsrv_5 ; nothing to do !
1321                                    <1>                             ; (we need to complete timer callback)
1322                                    <1>
1323                                    <1>             ; 04/03/2017
1324 0000F8B1 C605[D7030300]00          <1>             mov    byte [u.irqwait], 0 ; reset/clear waiting IRQ flag
1325                                    <1>
1326 0000F8B8 FE05[D8030300]            <1>             inc    byte [u.r_lock] ; 'IRQ callback service in progress' flag
1327                                    <1>
1328 0000F8BE 8A0D[5B030300]            <1>             mov    cl, [sysflg]   ; (system call) mode flag (kernel/user)
1329 0000F8C4 880D[D9030300]            <1>             mov    [u.r_mode], cl ; system mode (0) or user mode (FFh)
1330                                    <1>
1331                                    <1>             ;
1332 0000F8CA 8B2D[9C510100]            <1>             mov    ebp, [tss.esp0] ; kernel stack address (for ring 0)
1333 0000F8D0 83ED14                    <1>             sub    ebp, 20             ; eip, cs, eflags, esp, ss
1334 0000F8D3 892D[5C030300]            <1>             mov    [u.sp], ebp
1335 0000F8D9 8925[60030300]            <1>             mov    [u.usp], esp
1336                                    <1>
1337                                    <1>             ;or    word [ebp+8], 200h ; 22/01/2017, force enabling interrupts
1338                                    <1>
1339 0000F8DF 8B44241C                  <1>             mov    eax, [esp+28] ; pushed eax
1340 0000F8E3 A3[64030300]              <1>             mov    [u.r0], eax
1341                                    <1>
1342 0000F8E8 E83EEEFFFF                <1>             call   wswap ; save user's registers & status
1343                                    <1>
1344                                    <1>             ; software int is in ring 0 but IRQ handler must return to ring 3
1345                                    <1>             ; so, ring 3 return address and stack registers
1346                                    <1>             ; (eip, cs, eflags, esp, ss)
1347                                    <1>             ; must be copied to IRQ handler return
1348                                    <1>             ; eip will be replaced by callback service routine address
1349                                    <1>
1350 0000F8ED C605[5B030300]FF          <1>             mov    byte [sysflg], 0FFh ; user mode
1351                                    <1>
1352                                    <1>             ; system mode (system call)
1353                                    <1>             ;mov   ebp, [u.sp] ; EIP (u), CS (UCODE), EFLAGS (u),
1354                                    <1>                                ; ESP (u), SS (UDATA)
1355                                    <1>
1356 0000F8F4 8B4510                    <1>             mov    eax, [ebp+16]; SS (UDATA)
1357 0000F8F7 89E6                      <1>             mov    esi, esp
1358 0000F8F9 50                        <1>             push   eax
1359 0000F8FA 50                        <1>             push   eax
1360 0000F8FB 89E7                      <1>             mov    edi, esp
1361 0000F8FD 893D[60030300]            <1>             mov    [u.usp], edi
1362 0000F903 B908000000                <1>             mov    ecx, ((ESPACE/4) - 4) ; except DS, ES, FS, GS
1363 0000F908 F3A5                      <1>             rep    movsd
1364 0000F90A B104                      <1>             mov    cl, 4
1365 0000F90C F3AB                      <1>             rep    stosd
1366 0000F90E 893D[5C030300]            <1>             mov    [u.sp], edi
1367 0000F914 89EE                      <1>             mov    esi, ebp
1368 0000F916 B105                      <1>             mov    cl, 5 ; EIP (u), CS (UCODE), EFLAGS (u), ESP (u), SS (UDATA)
1369 0000F918 F3A5                      <1>             rep    movsd
1370                                    <1>             ;
1371                                    <1>
1372 0000F91A 8B0D[B8030300]            <1>             mov    ecx, [u.pgdir]
1373 0000F920 890D[7E650100]            <1>             mov    [IRQ_cr3], ecx
1374                                    <1>
1375                                    <1> set_IRQ_callback_addr:
1376                                    <1>             ;
1377                                    <1>             ; This routine sets return address
1378                                    <1>             ; to start of user's interrupt
1379                                    <1>             ; service (callback) address
1380                                    <1>             ;
1381                                    <1>             ; INPUT:
1382                                    <1>             ;     EDX = callback routine/service address
1383                                    <1>             ;           (virtual, not physical address!)
```

```
1384                                    <1>             ;      [u.sp] = kernel stack, points to
1385                                    <1>             ;             user's EIP,CS,EFLAGS,ESP,SS
1386                                    <1>             ;             registers.
1387                                    <1>             ; OUTPUT:
1388                                    <1>             ;      EIP (user) = callback (service) address
1389                                    <1>             ;      CS (user) = UCODE
1390                                    <1>             ;      EFLAGS (user) = flags before callback
1391                                    <1>             ;        ESP (user) = ESP-4 (user, before callback)
1392                                    <1>             ;      [ESP](user) = EIP (user) before callback
1393                                    <1>             ;
1394                                    <1>             ; Note: If CPU was in user mode while entering
1395                                    <1>             ;      the timer interrupt service routine,
1396                                    <1>             ;      'IRET' will get return to callback routine
1397                                    <1>             ;      immediately. If CPU was in system/kernel mode
1398                                    <1>             ;      'iret' will get return to system call and
1399                                    <1>             ;      then, callback routine will be return address
1400                                    <1>             ;      from system call. (User's callback/service code
1401                                    <1>             ;      will be able to return to normal return address
1402                                    <1>             ;      via a 'sysrele' system call at the end.)
1403                                    <1>             ;
1404                                    <1>             ; Note: User's IRQ callback service code must be ended
1405                                    <1>             ;      with a 'sysrele' system call !
1406                                    <1>             ;
1407                                    <1>             ;      For example:
1408                                    <1>             ;
1409                                    <1>             ;      audio_IRQ_callback:
1410                                    <1>             ;          ...
1411                                    <1>             ;          <load DMA buffer with audio data>
1412                                    <1>             ;          ...
1413                                    <1>             ;          mov eax, 39 ; 'sysrele'
1414                                    <1>             ;          int 40h ; TRDOS 386 system call (interrupt)
1415                                    <1>             ;
1416                                    <1>
1417                                    <1>             ;mov   edx, [edx+IRQ.addr] ; Callback service address
1418                                    <1>             ;                          ; (Virtual address)
1419                                    <1>
1420 0000F926 8B2D[5C030300]           <1>             mov    ebp, [u.sp]; kernel's stack, points to EIP (user)
1421 0000F92C 895500                   <1>             mov    [ebp], edx
1422                                    <1> IRQsrv_5:
1423                                    <1>             ; EOI & return
1424                                    <1>             ; 11/06/2017
1425                                    <1>             ; 10/06/2017
1426 0000F92F A0[82650100]             <1>             mov    al, [IRQnum]
1427 0000F934 FA                       <1>             cli
1428 0000F935 3C07                     <1>             cmp    al, 7
1429 0000F937 7604                     <1>             jna    short IRQsrv_6
1430                                    <1>             ;
1431                                    <1>             ;mov   al, EOI      ; end of interrupt
1432 0000F939 B020                     <1>             mov    al, 20h
1433                                    <1>             ;cli        ; disable interrupts till stack cleared
1434                                    <1>             ;out   INTB00, al ; For controll2 #2
1435 0000F93B E6A0                     <1>             out    0A0h, al
1436                                    <1> IRQsrv_6:
1437                                    <1>             ;mov   byte [IRQnum], 0 ; reset
1438                                    <1>             ;mov   al, EOI      ; end of interrupt
1439 0000F93D B020                     <1>             mov    al, 20h
1440                                    <1>             ;cli        ; disable interrupts till stack cleared
1441                                    <1>             ;out   INTA00, al ; end of interrupt to 8259 - 1
1442 0000F93F E620                     <1>             out    20h, al
1443                                    <1> IRQsrv_7:
1444                                    <1>             ;; 13/06/2017
1445                                    <1>             ;or    word [ebp+8], 200h ; force enabling interrupts
1446                                    <1>             ;
1447 0000F941 8B0D[7E650100]           <1>             mov    ecx, [IRQ_cr3]      ; previous content of cr3 register
1448 0000F947 0F22D9                   <1>             mov    cr3, ecx     ; restore cr3 register content
1449                                    <1>             ;
1450 0000F94A 61                       <1>             popad ; edi,esi,ebp,(icrement esp by 4), ebx,edx,ecx,eax
1451                                    <1>             ;
1452 0000F94B 0FA9                     <1>             pop    gs
1453 0000F94D 0FA1                     <1>             pop    fs
1454 0000F94F 07                       <1>             pop    es
1455 0000F950 1F                       <1>             pop    ds
1456                                    <1>             ;
1457 0000F951 CF                       <1>             iretd  ; return from interrupt
1458                                    <1>
1459                                    <1> get_device_number:
1460                                    <1>             ; 08/10/2016
1461                                    <1>             ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
1462                                    <1>             ;
1463                                    <1>             ; This procedure compares name of requested
1464                                    <1>             ; device with kernel device names and
1465                                    <1>             ; installable device names. If names match,
1466                                    <1>             ; the relevant device index (entry) number
1467                                    <1>             ; will be returned the caller (sysopen)
1468                                    <1>             ; for the requested device.
1469                                    <1>             ;
1470                                    <1>             ; NOTE: Installable device drivers must
1471                                    <1>             ; be loaded before using 'sysopen'
1472                                    <1>             ; (opendev) system call.
1473                                    <1>             ;
1474                                    <1>             ; INPUT:
1475                                    <1>             ;    ESI = device name address (ASCIIZ)
1476                                    <1>             ;         (in kernel's memory space)
1477                                    <1>             ;    max name length = 8 without '/dev/')
1478                                    <1>             ;    Device name will be capitalized
1479                                    <1>             ;    and if there is, '/dev/' will be
1480                                    <1>             ;    removed from name before comparising)
1481                                    <1>             ;
1482                                    <1>             ; OUTPUT:
1483                                    <1>             ;    cf = 0 ->
1484                                    <1>             ;      EAX (AL) = device entry/index number
1485                                    <1>             ;    cf = 1 -> device not found (installed)
```

```
1486                            <1>             ;            or invalid device name
1487                            <1>             ;            (AL=0)
1488                            <1>             ;    device_name = device name address (asciiz)
1489                            <1>             ;
1490                            <1>             ; Modified registers: EAX, EBX, ESI, EDI
1491                            <1>
1492 0000F952 BF[C15F0100]     <1>             mov    edi, device_name
1493 0000F957 E805010000       <1>             call   lodsb_capitalize
1494 0000F95C 88C4             <1>             mov    ah, al
1495 0000F95E 3C2F             <1>             cmp    al, '/'
1496 0000F960 750E             <1>             jne    short gdn_1
1497 0000F962 BF[C15F0100]     <1>             mov    edi, device_name
1498 0000F967 E8F5000000       <1>             call   lodsb_capitalize
1499                            <1> gdn_0:
1500 0000F96C 20C0             <1>             and    al, al ; 0 ?
1501 0000F96E 7420             <1>             jz     short gdn_err ; null name after '/'
1502                            <1> gdn_1:
1503 0000F970 3C44             <1>             cmp    al, 'D'
1504 0000F972 7517             <1>             jne    short gdn_2
1505 0000F974 E8E8000000       <1>             call   lodsb_capitalize
1506 0000F979 3C45             <1>             cmp    al, 'E'
1507 0000F97B 750E             <1>             jne    short gdn_2
1508 0000F97D E8DF000000       <1>             call   lodsb_capitalize
1509 0000F982 3C56             <1>             cmp    al, 'V'
1510 0000F984 7505             <1>             jne    short gdn_2
1511 0000F986 AC               <1>             lodsb
1512 0000F987 3C2F             <1>             cmp    al, '/'
1513 0000F989 740D             <1>             je     short gdn_4
1514                            <1> gdn_2:
1515 0000F98B 80FC2F           <1>             cmp    ah, '/'
1516 0000F98E 750F             <1>             jne    short gdn_5
1517                            <1> gdn_err:
1518                            <1>             ; invalid device name or device not found
1519 0000F990 31C0             <1>             xor    eax, eax ; 0
1520 0000F992 F9               <1>             stc
1521 0000F993 C3               <1>             retn
1522                            <1> gdn_3:
1523 0000F994 3C2F             <1>             cmp    al, '/'
1524 0000F996 7507             <1>             jne    short gdn_5
1525                            <1> gdn_4:
1526 0000F998 BF[C15F0100]     <1>             mov    edi, device_name
1527 0000F99D EB04             <1>             jmp    short gdn_6
1528                            <1> gdn_5:
1529 0000F99F 3C00             <1>             cmp    al, 0
1530 0000F9A1 7419             <1>             je     short gdn_7
1531                            <1> gdn_6:
1532 0000F9A3 E8B9000000       <1>             call lodsb_capitalize
1533 0000F9A8 81FF[C95F0100]   <1>             cmp    edi, device_name + 8
1534 0000F9AE 72E4             <1>             jb     short gdn_3
1535 0000F9B0 3C00             <1>             cmp    al, 0
1536 0000F9B2 75DC             <1>             jne    short gdn_err
1537 0000F9B4 81FF[C25F0100]   <1>             cmp    edi, device_name + 1
1538 0000F9BA 76D4             <1>             jna    short gdn_err ; null name after '/'
1539                            <1> gdn_7:
1540 0000F9BC AA               <1>             stosb
1541                            <1>             ; zero padding ("NAME",0,0,0,0)
1542 0000F9BD 81FF[C95F0100]   <1>             cmp    edi, device_name + 8
1543 0000F9C3 72F7             <1>             jb     short gdn_7
1544                            <1> gdn_8:
1545                            <1>             ; search for kernel device names
1546 0000F9C5 BE[C15F0100]     <1>             mov    esi, device_name
1547 0000F9CA BF[C80D0100]     <1>             mov    edi, KDEV_NAME
1548 0000F9CF 31C0             <1>             xor    eax, eax
1549                            <1> gdn_9:
1550 0000F9D1 A7               <1>             cmpsd
1551 0000F9D2 7505             <1>             jne    short gdn_10
1552 0000F9D4 A7               <1>             cmpsd
1553 0000F9D5 7503             <1>             jne    short gdn_11
1554 0000F9D7 EB2B             <1>             jmp    short gdn_17 ; match
1555                            <1> gdn_10:
1556 0000F9D9 A7               <1>             cmpsd  ; add esi, 4 & add edi, 4
1557                            <1> gdn_11:
1558 0000F9DA BE[C15F0100]     <1>             mov    esi, device_name
1559 0000F9DF FEC0             <1>             inc    al
1560 0000F9E1 3C16             <1>             cmp    al, NumOfKernelDevNames
1561 0000F9E3 72EC             <1>             jb     short gdn_9
1562                            <1> gdn_12:
1563                            <1>             ; search for installable device names
1564                            <1>             ; esi = offset device_name
1565 0000F9E5 BF[EC5F0100]     <1>             mov    edi, IDEV_NAME
1566 0000F9EA 28C0             <1>             sub    al, al ; 0
1567                            <1> gdn_13:
1568 0000F9EC A7               <1>             cmpsd
1569 0000F9ED 7505             <1>             jne    short gdn_14
1570 0000F9EF A7               <1>             cmpsd
1571 0000F9F0 7503             <1>             jne    short gdn_15
1572 0000F9F2 EB3F             <1>             jmp    short gdn_19 ; match
1573                            <1> gdn_14:
1574 0000F9F4 A7               <1>             cmpsd  ; add esi, 4 & add edi, 4
1575                            <1> gdn_15:
1576 0000F9F5 BE[C15F0100]     <1>             mov    esi, device_name
1577 0000F9FA FEC0             <1>             inc    al
1578 0000F9FC 3C08             <1>             cmp    al, NumOfInstallableDevices
1579 0000F9FE 72EC             <1>             jb     short gdn_13
1580                            <1>
1581                            <1> gdn_16:     ; error: invalid device name (not found) !
1582 0000FA00 30C0             <1>             xor    al, al
1583 0000FA02 F9               <1>             stc
1584 0000FA03 C3               <1>             retn
1585                            <1>
1586                            <1> gdn_17:             ; name match (with one of kernel device names)
1587                            <1>             ;
```

```
1588                                    <1>             ; convert KDEV_NAME index to
1589                                    <1>             ; KDEV_NUMBER index
1590                                    <1>             ; (different names are used for same devices)
1591                                    <1>             ; (example: "COM1" & "TTY8" = device number 18)
1592 0000FA04 89C3                     <1>             mov    ebx, eax ; < 256
1593 0000FA06 8A83[780E0100]           <1>             mov    al, [KDEV_NUMBER+ebx]
1594                                    <1>
1595                                    <1>             ; check if empty dev entry in the list
1596 0000FA0C 80B8[70610100]00         <1>             cmp    byte [DEV_OPENMODE+eax], 0
1597 0000FA13 771B                     <1>             ja     short gdn_18 ; it must be already set
1598                                    <1>
1599                                    <1>             ; (re)set device name and access flags
1600                                    <1>             ; (remain open work will be easy after that)
1601                                    <1>             ; (NOTE: here, data will be copied to bss section)
1602 0000FA15 88C3                     <1>             mov    bl, al
1603 0000FA17 83EF08                   <1>             sub    edi, 8 ; kernel device name address (data)
1604 0000FA1A 66C1E302                 <1>             shl    bx, 2
1605 0000FA1E 89BB[8E610100]           <1>             mov    [DEV_NAME_PTR+ebx], edi ; (all) device names
1606 0000FA24 8A98[CE0F0100]           <1>             mov    bl, [KDEV_ACCESS+eax] ; kernel dev list (data)
1607 0000FA2A 8898[BC600100]           <1>             mov    [DEV_ACCESS+eax], bl ; (all) device list (bss)
1608                                    <1> gdn_18:
1609 0000FA30 FEC0                     <1>             inc    al ; 1 to NumOfKernelDevNames (<=7Fh)
1610                                    <1>             ; eax = device index/entry number
1611 0000FA32 C3                       <1>             retn
1612                                    <1>
1613                                    <1> gdn_19:             ; name match (with one of installable device names)
1614                                    <1>             ;
1615                                    <1>             ; al = 0 to NumOfInstallableDevices - 1 (<=7Fh)
1616                                    <1>
1617 0000FA33 89C3                     <1>             mov    ebx, eax
1618 0000FA35 80C316                   <1>             add    bl, NumOfKernelDevices    ; < NUMOFDEVICES
1619                                    <1>
1620                                    <1>             ; check if empty dev entry in the list
1621 0000FA38 80BB[70610100]00         <1>             cmp    byte [DEV_OPENMODE+ebx], 0
1622 0000FA3F 771D                     <1>             ja     short gdn_20 ; it must be already set
1623                                    <1>
1624                                    <1>             ; (re)set device name and access flags
1625                                    <1>             ; (remain open work will be easy after that)
1626 0000FA41 83EF08                   <1>             sub    edi, 8 ; installable device name address
1627 0000FA44 66C1E302                 <1>             shl    bx, 2 ;*4
1628 0000FA48 89BB[8E610100]           <1>             mov    [DEV_NAME_PTR+ebx], edi ; (all) device names
1629 0000FA4E 66C1EB02                 <1>             shr    bx, 2
1630 0000FA52 8A80[34600100]           <1>             mov    al, [IDEV_FLAGS+eax] ; installable dev list
1631 0000FA58 8883[BC600100]           <1>             mov    [DEV_ACCESS+ebx], al ; (all) device list
1632                                    <1> gdn_20:
1633 0000FA5E 88D8                     <1>             mov    al, bl
1634                                    <1>             ; eax = device index/entry number ; < NUMOFDEVICES
1635 0000FA60 C3                       <1>             retn
1636                                    <1>
1637                                    <1> lodsb_capitalize:
1638                                    <1>       ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
1639                                    <1>       ; INPUT -> [esi] = character
1640                                    <1>       ;          edi = destination
1641                                    <1>       ; OUTPUT -> AL contains capitalized character
1642                                    <1>       ;         esi = esi+1
1643                                    <1>       ;         edi = edi+1
1644                                    <1>       ;
1645 0000FA61 AC                       <1>       lodsb
1646 0000FA62 3C61                     <1>       cmp al, 61h
1647 0000FA64 7206                     <1>       jb  short lodsb_cap_retn
1648 0000FA66 3C7A                     <1>       cmp al, 7Ah
1649 0000FA68 7702                     <1>       ja  short lodsb_cap_retn
1650 0000FA6A 24DF                     <1>       and al, 0DFh
1651                                    <1> lodsb_cap_retn:
1652 0000FA6C AA                       <1>       stosb
1653 0000FA6D C3                       <1>       retn
1654                                    <1>
1655                                    <1> device_open:
1656                                    <1>       ; 08/10/2016 - TRDOS 386 (TRDOS v2.0)
1657                                    <1>       ; Complete device opening work for sysopen (device)
1658                                    <1>       ;
1659                                    <1>       ; INPUT ->
1660                                    <1>       ;      EAX = Device Number (AL)
1661                                    <1>       ;        CL = Open mode (1 = read, 2 = write)
1662                                    <1>       ;       CH = Device access byte (bit 0 = 0)
1663                                    <1>       ; OUTPUT ->
1664                                    <1>       ;      EAX = Device Number
1665                                    <1>       ;      CF = 0 -> device has been opened
1666                                    <1>       ;      CF = 1 -> device could not be opened
1667                                    <1>       ;
1668                                    <1>       ;  Modified registers: ebx, (edx, ecx, esi, edi, ebp)
1669                                    <1>       ;
1670                                    <1>
1671 0000FA6E 89C3                     <1>       mov    ebx, eax
1672 0000FA70 66C1E302                 <1>       shl    bx, 2 ; *4
1673                                    <1>
1674 0000FA74 F6C580                   <1>       test   ch, 80h ; bit 7, installable device driver flag
1675 0000FA77 7406                     <1>       jz     short d_open_2 ; Kernel device
1676                                    <1>       ; installable device
1677                                    <1> d_open_1:
1678 0000FA79 FFA3[38600100]           <1>         jmp  dword [ebx+IDEV_OADDR-4]
1679                                    <1> d_open_2:
1680 0000FA7F FFA3[8A0E0100]           <1>       jmp    dword [ebx+KDEV_OADDR-4]
1681                                    <1>
1682                                    <1> device_close:
1683                                    <1>       ; 08/10/2016 - TRDOS 386 (TRDOS v2.0)
1684                                    <1>       ; Complete device closing work for sysclose (device)
1685                                    <1>       ;
1686                                    <1>       ; INPUT ->
1687                                    <1>       ;      EAX = Device Number (AL)
1688                                    <1>       ;        CL = Open mode (1 = read, 2 = write)
1689                                    <1>       ;       CH = Device access byte (bit 0 = 0)
```

```
1690                                    <1>       ; OUTPUT ->
1691                                    <1>       ;      EAX = Device Number
1692                                    <1>       ;      CF = 0 -> device has been closed
1693                                    <1>       ;      CF = 1 -> device could not be closed
1694                                    <1>       ;
1695                                    <1>       ; Modified registers: ebx, (edx, ecx, esi, edi, ebp)
1696                                    <1>       ;
1697                                    <1>
1698 0000FA85 89C3                      <1>       mov    ebx, eax
1699 0000FA87 66C1E302                  <1>       shl    bx, 2 ; *4
1700                                    <1>
1701 0000FA8B F6C580                    <1>       test   ch, 80h ; bit 7, installable device driver flag
1702 0000FA8E 7406                      <1>       jz     short d_close_2 ; Kernel device
1703                                    <1>       ; installable device
1704                                    <1> d_close_1:
1705 0000FA90 FFA3[58600100]            <1>         jmp  dword [ebx+IDEV_CADDR-4]
1706                                    <1> d_close_2:
1707 0000FA96 FFA3[DA0E0100]            <1>       jmp    dword [ebx+KDEV_CADDR-4]
1708                                    <1>
1709                                    <1> rnull:
1710                                    <1>       ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
1711                                    <1>       ; read null (read from null device)
1712 0000FA9C C3                        <1>       retn
1713                                    <1>
1714                                    <1> wnull:
1715                                    <1>       ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
1716                                    <1>       ; write null (write to null device)
1717 0000FA9D C3                        <1>       retn
1718                                    <1>
1719                                    <1> dev_IRQ_service:
1720                                    <1>       ; 12/05/2017
1721                                    <1>       ; 13/04/2017
1722                                    <1>       ; 27/02/2017 - TRDOS 386 (TRDOS v2.0)
1723                                    <1>       ; INPUT ->
1724                                    <1>       ;      AL = IRQ Number (0 to 15)
1725                                    <1>       ;
1726 0000FA9E 53                        <1>       push   ebx
1727 0000FA9F 0FB6D8                    <1>       movzx  ebx, al
1728 0000FAA2 C0E302                    <1>       shl    bl, 2 ; * 4
1729 0000FAA5 8B9B[F6640100]            <1>       mov    ebx, [ebx+DEV_INT_HNDLR]
1730 0000FAAB 21DB                      <1>       and    ebx, ebx
1731 0000FAAD 7404                      <1>         jz   short dIRQ_s_retn
1732 0000FAAF 50                        <1>       push   eax
1733                                    <1>
1734 0000FAB0 FFD3                      <1>       call   ebx
1735                                    <1>
1736 0000FAB2 58                        <1>       pop    eax
1737                                    <1> dIRQ_s_retn:
1738 0000FAB3 5B                        <1>       pop    ebx
1739 0000FAB4 C3                        <1>       retn
1740                                    <1>
1741                                    <1>
1742                                    <1> set_dev_IRQ_service:
1743                                    <1>       ; 13/04/2017 - TRDOS 386 (TRDOS v2.0)
1744                                    <1>       ;
1745                                    <1>       ; Set Device Interrupt Service
1746                                    <1>       ;
1747                                    <1>       ; INPUT ->
1748                                    <1>       ;      AL = IRQ Number
1749                                    <1>       ;      EBX = Hardware Interrupt Service Address
1750                                    <1>       ;
1751                                    <1>       ; Note: There is not a validation check here
1752                                    <1>       ;      because this procedure is called by
1753                                    <1>       ;      TRDOS 386 kernel !
1754                                    <1>       ;       (Even if a device driver does not exist
1755                                    <1>       ;      this setting may be used by sysaudio
1756                                    <1>       ;      and other system calls for hardware
1757                                    <1>       ;      components which use IRQ method for I/O.)
1758                                    <1>       ;
1759                                    <1>       ;push  esi
1760 0000FAB5 0FB6F0                    <1>       movzx  esi, al
1761 0000FAB8 66C1E602                  <1>       shl    si, 2 ; * 4
1762 0000FABC 899E[F6640100]            <1>       mov    [esi+DEV_INT_HNDLR], ebx
1763                                    <1>       ;pop   esi
1764 0000FAC2 C3                        <1>       retn
1765                                    <1>
1766                                    <1>
1767                                    <1> sysaudio: ; AUDIO FUNCTIONS
1768                                    <1>       ; 10/10/2017
1769                                    <1>       ; 22/06/2017
1770                                    <1>       ; 28/05/2017, 04/06/2017, 05/06/2017, 10/06/2017
1771                                    <1>       ; 01/05/2017, 12/05/2017, 15/05/2017, 20/05/2017
1772                                    <1>       ; 21/04/2017, 22/04/2017, 23/04/2017, 24/04/2017
1773                                    <1>       ; 10/04/2017, 13/04/2017, 14/04/2017, 16/04/2017
1774                                    <1>       ; 03/04/2017 (VIA VT8237R)
1775                                    <1>       ; 01/04/2016 (trdosk6.s -> tdosk8.s)
1776                                    <1>       ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
1777                                    <1>       ;
1778                                    <1>       ; Inputs:
1779                                    <1>       ;
1780                                    <1>       ;      BH = 0 -> Beep (PC Speaker)
1781                                    <1>       ;           BL = Duration Counter (1 for 1/64 second)
1782                                    <1>       ;           CX = Frequency Divisor (1193180/Frequency)
1783                                    <1>       ;              (1331 for 886 Hz)
1784                                    <1>       ;
1785                                    <1>       ;      01/04/2017
1786                                    <1>       ;
1787                                    <1>       ;      BH = 1 -> DETECT (& ENABLE) AUDIO DEVICE
1788                                    <1>       ;           BL = 0 : PC SPEAKER
1789                                    <1>       ;                1 : SOUND BLASTER 16
1790                                    <1>       ;                2 : INTEL AC'97
1791                                    <1>       ;                3 : VIA VT8237R (VT8233)
```

```
1792                    <1>     ;               4 : INTEL HDA
1793                    <1>     ;             5-FEh : unknown/invalid
1794                    <1>     ;               ; 04/06/2017
1795                    <1>     ;              FFh : Get current audio device id
1796                    <1>     ;
1797                    <1>     ;       BH = 2 -> ALLOCATE AUDIO BUFFER (for user)
1798                    <1>     ;             ECX = Audio Buffer Size (must be equal to
1799                    <1>     ;                   the half of DMA buffer size)
1800                    <1>     ;             EDX = Virtual Address of the buffer
1801                    <1>     ;                   (This is not DMA buffer!)
1802                    <1>     ;
1803                    <1>     ;       BH = 3 -> INITIALIZE AUDIO DEVICE
1804                    <1>     ;            BL = 0,2 -> for Signal Response Byte
1805                    <1>     ;             CL = Signal Response Byte Value (fixed)
1806                    <1>     ;                         if BL = 0
1807                    <1>     ;                      auto increment of S.R.B. value
1808                    <1>     ;                         if BL = 2
1809                    <1>     ;               EDX = Signal Response (Return) Byte Address
1810                    <1>     ;
1811                    <1>     ;            BL = 1 for CallBack Method
1812                    <1>     ;             EDX = CallBack Service Address (Virtual)
1813                    <1>     ;
1814                    <1>     ;            BL > 2 -> invalid function
1815                    <1>     ;
1816                    <1>     ;           (Audio buffer must be allocated before
1817                    <1>     ;            initialization.)
1818                    <1>     ;
1819                    <1>     ;       BH = 4 -> START TO PLAY
1820                    <1>     ;            BL = Mode
1821                    <1>     ;                Bit 0 = mono/stereo (1 = stereo)
1822                    <1>     ;                Bit 1 = 8 bit / 16 bit (1 = 16 bit)
1823                    <1>     ;            CX = Sampling Rate (Hz)
1824                    <1>     ;
1825                    <1>     ;       BH = 5 -> PAUSE
1826                    <1>     ;            BL = Any
1827                    <1>     ;
1828                    <1>     ;       BH = 6 -> CONTINUE TO PLAY
1829                    <1>     ;            BL = Any
1830                    <1>     ;
1831                    <1>     ;       BH = 7 -> STOP
1832                    <1>     ;            BL = Any
1833                    <1>     ;
1834                    <1>     ;       BH = 8 -> RESET
1835                    <1>     ;            BL = Any
1836                    <1>     ;
1837                    <1>     ;       BH = 9 -> CANCEL (CALLBACK or S.R.B. SERVICE)
1838                    <1>     ;            BL = Any
1839                    <1>     ;
1840                    <1>     ;       BH = 10 -> DEALLOCATE AUDIO BUFFER (for user)
1841                    <1>     ;            BL = Any
1842                    <1>     ;
1843                    <1>     ;       BH = 11 -> SET VOLUME LEVEL
1844                    <1>     ;            BL: (Bit 0 to 6)
1845                    <1>     ;                0 = Master (Playback, Lineout) volume
1846                    <1>     ;            CL = Left Channel Volume
1847                    <1>     ;            CH = Right Channel Volume
1848                    <1>     ;
1849                    <1>     ;            Note: If BL >= 80h (Bit 7 of BL is set),
1850                    <1>     ;            volume level will be set for next playing
1851                    <1>     ;            (actual volume level will not be changed
1852                    <1>     ;            immediately)
1853                    <1>     ;
1854                    <1>     ;       BH = 12 -> DISABLE AUDIO DEVICE
1855                    <1>     ;            (reset audio device and unlink dma buffer)
1856                    <1>     ;            BL = Any
1857                    <1>     ;
1858                    <1>     ;       12/05/2017
1859                    <1>     ;       BH = 13 -> MAP DMA BUFFER TO USER
1860                    <1>     ;           (for direct access to system's dma buffer)
1861                    <1>     ;
1862                    <1>     ;            ECX = map size in bytes
1863                    <1>     ;                (will be rounded up to page borders)
1864                    <1>     ;            EDX = Virtual Address of the buffer
1865                    <1>     ;                (Will be rounded up to page borders)
1866                    <1>     ;
1867                    <1>     ;       05/06/2017
1868                    <1>     ;       04/06/2017
1869                    <1>     ;       BH = 14 -> GET AUDIO DEVICE INFO
1870                    <1>     ;            BL: 0 = Audio Controller Info
1871                    <1>     ;              > 0 = Invalid for now!
1872                    <1>     ;
1873                    <1>     ;       22/06/2017
1874                    <1>     ;       BH = 15 -> GET CURRENT SOUND DATA (for graphics)
1875                    <1>     ;            BL: 0 -> PCM OUT data
1876                    <1>     ;              > 0 -> Invalid for now!
1877                    <1>     ;            ECX = 0 -> Get DMA Buffer Pointer
1878                    <1>     ;               EDX = Not Used
1879                    <1>     ;            ECX > 0 -> Byte count for buffer (EDX)
1880                    <1>     ;                EDX = Buffer Address (Virtual)
1881                    <1>     ;
1882                    <1>     ;       10/10/2017
1883                    <1>     ;       BH = 16 -> UPDATE DMA BUFFER DATA
1884                    <1>     ;                (by using the Audio Buffer content)
1885                    <1>     ;            BL = 0 : Update dma half buffer in sequence
1886                    <1>     ;                     (automatic destination)
1887                    <1>     ;                1 : Update 1st half of the dma buffer
1888                    <1>     ;                2 : Update 2nd half of the dma buffer
1889                    <1>     ;                3-FEh: Invalid!
1890                    <1>     ;                FFh = Get current flag value
1891                    <1>     ;                     (Half buffer number -1)
1892                    <1>     ;
1893                    <1>     ;
```

```
                                <1>       ; Outputs:
                                <1>       ;
                                <1>       ;     For BH = 0 -> Beep
                                <1>       ;         None
                                <1>       ;
                                <1>       ;     01/04/2017
                                <1>       ;
                                <1>       ;     For BH = 1 -> DETECT (& ENABLE) AUDIO DEVICE
                                <1>       ;         AH = 0 : PC SPEAKER
                                <1>       ;              1 : SOUND BLASTER 16
                                <1>       ;              2 : INTEL AC'97
                                <1>       ;              3 : VIA VT8237R (VT8233)
                                <1>       ;              4 : INTEL HDA
                                <1>       ;            5-FFh : unknown/invalid
                                <1>       ;         AL = mode status
                                <1>       ;              bit 0 = mono /stereo (1 = stereo)
                                <1>       ;              bit 1 = 8 bit / 16 bit ( 1 = 16 bit)
                                <1>       ;         04/06/2017
                                <1>       ;         EBX = PCI DEVICE/VENDOR ID (if >0)
                                <1>       ;             (BX = VENDOR ID)
                                <1>       ;         (if CF = 1 -> Error code in EAX)
                                <1>       ;
                                <1>       ;     For BH = 2 -> ALLOCATE AUDIO BUFFER (for user)
                                <1>       ;         EAX = Physical Address of the buffer
                                <1>       ;         (if CF = 1 -> Error code in EAX)
                                <1>       ;
                                <1>       ;     For BH = 3 -> INITIALIZE AUDIO DEVICE
                                <1>       ;         (if CF = 1 -> Error code in EAX)
                                <1>       ;
                                <1>       ;     For BH = 4 -> START TO PLAY
                                <1>       ;         none (if CF = 1 -> Error code in EAX)
                                <1>       ;
                                <1>       ;     For BH = 5 -> PAUSE
                                <1>       ;         none (if CF = 1 -> Error code in EAX)
                                <1>       ;
                                <1>       ;     For BH = 6 -> CONTINUE TO PLAY
                                <1>       ;         none (if CF = 1 -> Error code in EAX)
                                <1>       ;
                                <1>       ;     For BH = 7 -> STOP
                                <1>       ;         none (if CF = 1 -> Error code in EAX)
                                <1>       ;
                                <1>       ;     For BH = 8 -> RESET
                                <1>       ;         none (if CF = 1 -> Error code in EAX)
                                <1>       ;
                                <1>       ;     For BH = 9 -> CANCEL (CALLBACK or S.R.B. SERVICE)
                                <1>       ;         none (if CF = 1 -> Error code in EAX)
                                <1>       ;
                                <1>       ;     For BH = 10 -> DEALLOCATE AUDIO BUFFER (for user)
                                <1>       ;         none (if CF = 1 -> Error code in EAX)
                                <1>       ;
                                <1>       ;     For BH = 11 -> SET VOLUME LEVEL
                                <1>       ;         none (if CF = 1 -> Error code in EAX)
                                <1>       ;
                                <1>       ;     For BH = 12 -> DISABLE AUDIO DEVICE
                                <1>       ;         none (if CF = 1 -> Error code in EAX)
                                <1>       ;
                                <1>       ;     12/05/2017
                                <1>       ;     For BH = 13 -> MAP DMA BUFFER TO USER
                                <1>       ;         EAX = Physical Address of the buffer
                                <1>       ;         (if CF = 1 -> Error code in EAX)
                                <1>       ;
                                <1>       ;     04/06/2017
                                <1>       ;     For BH = 14 -> GET AUDIO DEVICE INFO
                                <1>       ;     (for BL = 0) ; 05/06/2017
                                <1>       ;         EAX = IRQ Number in AL
                                <1>       ;               Audio Device Number in AH
                                <1>       ;         EBX = DEV/VENDOR ID
                                <1>       ;             (DDDDDDDDDDDDDDDDVVVVVVVVVVVVVVVV)
                                <1>       ;         ECX = BUS/DEV/FN
                                <1>       ;             (00000000BBBBBBBBDDDDDFFF00000000)
                                <1>       ;         EDX = NABMBAR/NAMBAR (for AC97)
                                <1>       ;               (Low word, DX = NAMBAR address)
                                <1>       ;         EDX = Base IO Addr (DX) for SB16 & VT8233
                                <1>       ;         (if CF = 1 -> Error code in EAX)
                                <1>       ;                       (ERR_DEV_NOT_RDY = 15)
                                <1>       ;
                                <1>       ;     22/06/2017
                                <1>       ;     For BH = 15 -> GET CURRENT SOUND DATA
                                <1>       ;                    (for graphics)
                                <1>       ;     (for BL = 0)
                                <1>       ;      If ECX input is 0
                                <1>       ;         EAX = DMA Buffer Current Position (Offset)
                                <1>       ;      If ECX input >  0
                                <1>       ;         EAX = Actual transfer count
                                <1>       ;         (Sound samples will be copied from
                                <1>       ;          Current DMA Buffer Position to EDX
                                <1>       ;          virtual address as EAX bytes.)
                                <1>       ;       ((If CF = 1 ->  Error code in EAX))
                                <1>       ;
                                <1>       ;
                                <1>       ;     10/10/2017
                                <1>       ;     For BH = 16 -> UPDATE DMA BUFFER DATA
                                <1>       ;         EAX = 0, if the updated (or current)
                                <1>       ;                 half buffer is DMA half buffer 1
                                <1>       ;         EAX = 1, if the updated (or current)
                                <1>       ;                 half buffer is DMA half buffer 2
                                <1>       ;         (If CF = 1 -> Error code in EAX)
                                <1>       ;
0000FAC3 80FF11                 <1>       cmp   bh, AUDIO1L/4
0000FAC6 0F83E7C9FFFF           <1>       jnb   sysret
                                <1>
```

```
1996 0000FACC C0E702              <1>       shl    bh, 2 ; *4
1997 0000FACF 0FB6F7              <1>       movzx  esi, bh
1998                              <1>
1999                              <1>       ; 22/04/2017
2000 0000FAD2 31C0                <1>       xor    eax, eax
2001 0000FAD4 A3[64030300]        <1>       mov    [u.r0], eax  ; 0
2002                              <1>
2003 0000FAD9 FF96[E4FA0000]      <1>       call   dword [esi+AUDIO1]
2004                              <1>       ;jc    error
2005 0000FADF E9CFC9FFFF          <1>       jmp    sysret
2006                              <1>
2007 0000FAE4 [A11D0000]          <1> AUDIO1:     dd    beep ; FUNCTION = 0 (bl = Duration Counter
2008                              <1>                       ;                 cx = Frequency Divisor
2009 0000FAE8 [28FB0000]          <1>       dd     soundc_detect
2010 0000FAEC [C4FB0000]          <1>       dd     sound_alloc
2011 0000FAF0 [7BFC0000]          <1>       dd     soundc_init
2012 0000FAF4 [33FE0000]          <1>       dd     sound_play
2013 0000FAF8 [C9FE0000]          <1>       dd     sound_pause
2014 0000FAFC [F3FE0000]          <1>       dd     sound_continue
2015 0000FB00 [1DFF0000]          <1>       dd     sound_stop
2016 0000FB04 [46FF0000]          <1>       dd     soundc_reset
2017 0000FB08 [77FF0000]          <1>       dd     soundc_cancel
2018 0000FB0C [9DFF0000]          <1>       dd     sound_dalloc
2019 0000FB10 [C8FF0000]          <1>       dd     sound_volume
2020 0000FB14 [1A000100]          <1>       dd     soundc_disable
2021 0000FB18 [8C000100]          <1>       dd     sound_dma_map
2022 0000FB1C [FB000100]          <1>       dd     soundc_info
2023 0000FB20 [5A010100]          <1>       dd     sound_data
2024 0000FB24 [07020100]          <1>       dd     sound_update
2025                              <1>
2026                              <1> AUDIO1L     EQU    $ - AUDIO1
2027                              <1>
2028                              <1> soundc_detect:
2029                              <1>       ; FUNCTION = 1
2030                              <1>       ; bl = Audio device type number
2031                              <1>       ; (0= pc speaker, 1 = sound blaster 16, 2 = intel ac97
2032                              <1>       ;  3= via vt823x, 4 = intel HDA, 0FFh= any)
2033                              <1>
2034                              <1>       ; 04/06/2017
2035 0000FB28 8A25[85650100]      <1>       mov    ah, [audio_device]
2036 0000FB2E 80FBFF              <1>       cmp    bl, 0FFh ; get current audio device id
2037 0000FB31 7408                <1>       je     short sysaudio0
2038                              <1>
2039 0000FB33 20E4                <1>       and    ah, ah
2040 0000FB35 741E                <1>       jz     short soundc_get_dev
2041                              <1>
2042 0000FB37 38DC                <1>       cmp    ah, bl
2043 0000FB39 7567                <1>       jne    short soundc_dev_err
2044                              <1>
2045                              <1> sysaudio0:
2046 0000FB3B A0[86650100]        <1>       mov    al, [audio_mode]
2047                              <1> sysaudio1:
2048 0000FB40 A3[64030300]        <1>       mov    [u.r0], eax
2049 0000FB45 8B1D[90650100]      <1>       mov    ebx, [audio_vendor] ; (DEVICE/VENDOR ID)
2050 0000FB4B 8B2D[60030300]      <1>       mov    ebp, [u.usp]
2051 0000FB51 895D10              <1>       mov    [ebp+16], ebx  ; ebx
2052 0000FB54 C3                  <1>       retn
2053                              <1>
2054                              <1> soundc_get_dev:
2055                              <1>       ; 28/05/2017
2056                              <1>       ; 03/04/2017, 24/04/2017
2057 0000FB55 C605[84650100]00    <1>       mov    byte [audio_pci], 0
2058 0000FB5C 80FB03              <1>       cmp    bl, 3 ; VIA VT8233 (VT8237R) Audio Controller & AC97 Codec
2059                              <1>       ;jne   short soundc_get_dev_sb
2060                              <1>       ; 28/05/2017
2061 0000FB5F 7220                <1>       jb     short soundc_get_dev_sb
2062 0000FB61 773F                <1>       ja     short soundc_dev_err  ; temporary (28/05/2017)
2063                              <1>       ;
2064 0000FB63 E83C180000          <1>       call   DetectVT8233
2065 0000FB68 7238                <1>       jc     short soundc_dev_err
2066                              <1>       ; eax = 0
2067                              <1>
2068                              <1>       ;mov   ebx, [audio_vendor]
2069                              <1>       ; ebx = DEVICE/VENDOR ID
2070                              <1>       ;      DDDDDDDDDDDDDDDDVVVVVVVVVVVVVVVV
2071                              <1>
2072 0000FB6A B003                <1>       mov    al, 3  ; VIA VT8237R (VT3233) Audio Controller
2073 0000FB6C 88C4                <1>       mov    ah, al
2074                              <1>
2075                              <1> soundc_get_pci_dev_ok: ; 28/05/2017
2076 0000FB6E FE05[84650100]      <1>       inc    byte [audio_pci] ; = 1
2077                              <1> soundc_get_dev_ok:
2078                              <1>
2079                              <1> soundc_get_dev_sb16_ok:
2080 0000FB74 A2[85650100]        <1>       mov    [audio_device], al
2081 0000FB79 8825[86650100]      <1>       mov    [audio_mode], ah ; stereo (bit0), 16 bit (bit1) capability
2082 0000FB7F EBBF                <1>       jmp    short sysaudio1
2083                              <1>
2084                              <1> soundc_get_dev_sb:
2085                              <1>       ; 24/04/2017
2086 0000FB81 80FB01              <1>       cmp    bl, 1 ; Sound Blaster 16
2087 0000FB84 750E                <1>       jne    short soundc_get_dev_ich ; 28/05/2017
2088                              <1>       ;
2089 0000FB86 E8471D0000          <1>       call   DetectSB
2090 0000FB8B 7215                <1>       jc     short soundc_dev_err
2091 0000FB8D B801030000          <1>       mov    eax, 0301h ; Sound Blaster 16
2092 0000FB92 EBE0                <1>       jmp    short soundc_get_dev_sb16_ok
2093                              <1>
2094                              <1> soundc_get_dev_ich:
2095                              <1>       ; 28/05/2017
2096                              <1>       ;cmp   bl, 2 ; Intel AC'97 Audio Controller (ICH)
2097                              <1>       ;jne   short soundc_dev_err ; Temporary  (28/05/2017)
```

```
                                <1>       ;                       ; (Here will be modified just after
                                <1>       ;                       ; new sound card code will be ready!)
0000FB94 E8FE170000             <1>       call  DetectICH
0000FB99 7207                   <1>       jc    short soundc_dev_err
                                <1>       ;
0000FB9B B802030000             <1>       mov   eax, 0302h ; AC'97 (ICH)
0000FBA0 EBCC                   <1>       jmp   short soundc_get_pci_dev_ok
                                <1>
                                <1> soundc_dev_err:
0000FBA2 B80F000000             <1>       mov   eax, ERR_DEV_NOT_RDY ; Device not ready !
0000FBA7 EB0C                   <1>       jmp   short sysaudio_err
                                <1>
                                <1> sound_buff_error:
0000FBA9 B82E000000             <1>       mov   eax, ERR_BUFFER ; Buffer error !
0000FBAE EB05                   <1>       jmp   short sysaudio_err
                                <1>
                                <1> soundc_respond_err:
                                <1>       ; ERR_TIME_OUT ; 'time out !' error
0000FBB0 B819000000             <1>       mov   eax, ERR_DEV_NOT_RESP ; 'device not responding !' error
                                <1> sysaudio_err:
0000FBB5 A3[64030300]           <1>       mov   [u.r0], eax
0000FBBA A3[C8030300]           <1>       mov   [u.error], eax
0000FBBF E9CFC8FFFF             <1>       jmp   error
                                <1>
                                <1> sound_alloc:
                                <1>       ; FUNCTION =  2
                                <1>       ; ecx = audio buffer size (in bytes)
                                <1>       ; edx = audio buffer address (virtual)
                                <1>       ; 28/05/2017
                                <1>       ; 01/05/2017, 15/05/2017
                                <1>       ; 21/04/2017, 24/04/2017
0000FBC4 803D[84650100]00       <1>       cmp   byte [audio_pci], 0
0000FBCB 7708                   <1>       ja    short snd_alloc_0
                                <1>       ; Max. 64KB DMA buffer !!!
0000FBCD 81F900800000           <1>       cmp   ecx, 32768
0000FBD3 77D4                   <1>       ja    short sound_buff_error
                                <1> snd_alloc_0:
                                <1>       ; 15/05/2017
0000FBD5 81F900100000           <1>       cmp   ecx, 4096 ; PAGE_SIZE
0000FBDB 72CC                   <1>       jb    short sound_buff_error
                                <1>       ;
0000FBDD A1[98650100]           <1>       mov   eax, [audio_buffer] ; audio buffer address (current)
0000FBE2 09C0                   <1>       or    eax, eax
0000FBE4 7445                   <1>       jz    short snd_alloc_2
                                <1>       ; audio buffer exists !
0000FBE6 8A1D[B3030300]         <1>       mov   bl, [u.uno]
0000FBEC 3A1D[AD650100]         <1>       cmp   bl, [audio_user]
0000FBF2 0F85F5000000           <1>       jne   sndc_owner_error ; not owner !
0000FBF8 39D0                   <1>       cmp   eax, edx ; same virtual buffer address ?
0000FBFA 7508                   <1>       jne   short snd_alloc_1
0000FBFC 3B0D[A0650100]         <1>       cmp   ecx, [audio_buff_size]
0000FC02 746C                   <1>       je    short snd_alloc_3 ; Nothing to do !
                                <1>                               ; Buffer has been set already!
                                <1> snd_alloc_1:
0000FC04 51                     <1>       push  ecx
0000FC05 52                     <1>       push  edx
0000FC06 89C3                   <1>       mov   ebx, eax ; audio buffer address (current)
0000FC08 8B0D[A0650100]         <1>       mov   ecx, [audio_buff_size]
0000FC0E E86B5BFFFF             <1>       call  deallocate_user_pages
0000FC13 5A                     <1>       pop   edx
0000FC14 59                     <1>       pop   ecx
0000FC15 31C0                   <1>       xor   eax, eax ; 0
0000FC17 A3[98650100]           <1>       mov   [audio_buffer], eax  ; 0
0000FC1C A3[9C650100]           <1>       mov   [audio_p_buffer], eax  ; 0
0000FC21 A3[A0650100]           <1>       mov   [audio_buff_size], eax
0000FC26 A2[AD650100]           <1>       mov   [audio_user], al ; 0
                                <1> snd_alloc_2:
0000FC2B 89D3                   <1>       mov   ebx, edx
                                <1>       ; 01/05/2017
0000FC2D BA00F0FFFF             <1>       mov   edx, ~PAGE_OFF ; truncating page offsets
                                <1>                            ; for aligning to page borders
                                <1>       ;and   eax, edx
0000FC32 21D3                   <1>       and   ebx, edx
0000FC34 21D1                   <1>       and   ecx, edx
                                <1>       ; 15/05/2017
                                <1>       ; EAX = Beginning address (physical)
                                <1>       ; EAX = 0 -> Allocate mem block from the 1st proper aperture
                                <1>       ; ECX = Number of bytes to be allocated
0000FC36 E8E857FFFF             <1>       call  allocate_memory_block
0000FC3B 0F8268FFFFFF           <1>       jc    sound_buff_error
                                <1>       ; EAX = Physical address of the allocated memory block
                                <1>       ; ECX = Allocated bytes (as truncated to page border)
                                <1>       ; EBX = Virtual address (as truncated to page border)
0000FC41 50                     <1>       push  eax
0000FC42 53                     <1>       push  ebx
0000FC43 51                     <1>       push  ecx
0000FC44 E82A5CFFFF             <1>       call  allocate_user_pages
0000FC49 59                     <1>       pop   ecx
0000FC4A 5B                     <1>       pop   ebx
0000FC4B 58                     <1>       pop   eax
0000FC4C 7223                   <1>       jc    short snd_alloc_4  ; insufficient memory, buff error
                                <1>       ; eax = physical address of the user's audio buffer
                                <1>       ; ebx = virtual address of the user's audio buffer
                                <1>       ; ecx = buffer size (in bytes)
0000FC4E A3[9C650100]           <1>       mov   [audio_p_buffer], eax
0000FC53 891D[98650100]         <1>       mov   [audio_buffer], ebx
0000FC59 890D[A0650100]         <1>       mov   [audio_buff_size], ecx
0000FC5F 8A15[B3030300]         <1>       mov   dl, [u.uno]
0000FC65 8815[AD650100]         <1>       mov   [audio_user], dl
0000FC6B A3[64030300]           <1>       mov   [u.r0], eax
                                <1> snd_alloc_3:
0000FC70 C3                     <1>       retn
```

```
                                <1> snd_alloc_4:
                                <1>       ; 15/05/2017
                                <1>       ; EAX = Beginning address (physical)
                                <1>       ; ECX = Number of bytes to be deallocated
0000FC71 E8BA59FFFF             <1>       call   deallocate_memory_block
0000FC76 E92EFFFFFF             <1>       jmp    sound_buff_error  ; insufficient memory, buff error
                                <1>
                                <1> soundc_init:
                                <1>       ; FUNCTION = 3
                                <1>       ; bl = method (0= s.r.b., 1= callback, 2= auto incr s.r.b.)
                                <1>       ; cl = signal response byte (initial or fixed) value
                                <1>       ; edx = signal response byte or callback address
                                <1>       ; 28/05/2017
                                <1>       ; 12/05/2017, 20/05/2017
                                <1>       ; 22/04/2017, 23/04/2017, 24/04/2017
                                <1>       ; 13/04/2017, 14/04/2017, 16/04/2017, 21/04/2017
                                <1>       ; 03/04/2017, 10/04/2017
                                <1>
0000FC7B A0[85650100]           <1>       mov    al, [audio_device]
0000FC80 20C0                   <1>       and    al, al
0000FC82 7549                   <1>       jnz    short sndc_init_6
                                <1>       ;
0000FC84 C605[84650100]00       <1>       mov    byte [audio_pci], 0
0000FC8B 52                     <1>       push   edx
0000FC8C 53                     <1>       push   ebx
0000FC8D 51                     <1>       push   ecx
0000FC8E E83F1C0000             <1>       call   DetectSB
0000FC93 7213                   <1>       jc     short sndc_init_8
0000FC95 66B80103               <1>       mov    ax, 0301h ; Sound Blaster 16
0000FC99 EB1E                   <1>       jmp    short sndc_init_7
                                <1>
                                <1> sndc_init_11:
                                <1>       ; 28/05/2017
0000FC9B E8F7160000             <1>       call   DetectICH ; Detect AC'97 (ICH) Audio Controller
0000FCA0 7217                   <1>       jc     short sndc_init_7
0000FCA2 66B80203               <1>       mov    ax, 0302h ; Intel AC'97 Audio Device
0000FCA6 EB0B                   <1>       jmp    short sndc_init_12 ; (PCI device)
                                <1>
                                <1> sndc_init_8:
0000FCA8 E8F7160000             <1>       call   DetectVT8233
                                <1>       ;jc    short sndc_init_7
0000FCAD 72EC                   <1>       jc     sndc_init_11 ; 28/05/2017
                                <1>       ; eax = 0
0000FCAF B003                   <1>       mov    al, 3  ; VIA VT8237R (VT3233) Audio Controller
0000FCB1 88C4                   <1>       mov    ah, al
                                <1>
                                <1> sndc_init_12:
0000FCB3 FE05[84650100]         <1>       inc    byte [audio_pci] ; = 1
                                <1> sndc_init_7:
0000FCB9 59                     <1>       pop    ecx
0000FCBA 5B                     <1>       pop    ebx
0000FCBB 5A                     <1>       pop    edx
0000FCBC 0F82E0FEFFFF           <1>       jc     soundc_dev_err
                                <1>       ;
0000FCC2 A2[85650100]           <1>       mov    [audio_device], al
0000FCC7 8825[86650100]         <1>       mov    [audio_mode], ah ; stereo (bit0), 16 bit (bit1) capability
                                <1>
                                <1> sndc_init_6:
0000FCCD 833D[98650100]00       <1>       cmp    dword [audio_buffer], 0
0000FCD4 0F86CFFEFFFF           <1>       jna    sound_buff_error
                                <1>
0000FCDA A0[B3030300]           <1>       mov    al, [u.uno]
0000FCDF 8A25[AD650100]         <1>       mov    ah, [audio_user]
0000FCE5 08E4                   <1>       or     ah, ah
0000FCE7 7418                   <1>       jz     short sndc_init0
0000FCE9 38E0                   <1>       cmp    al, ah
0000FCEB 7419                   <1>       je     short sndc_init1
                                <1>
                                <1> sndc_owner_error:
0000FCED B80B000000             <1>       mov    eax, ERR_NOT_OWNER ; 'permission denied !' error
                                <1> sndc_perm_error:
0000FCF2 A3[64030300]           <1>       mov    [u.r0], eax
0000FCF7 A3[C8030300]           <1>       mov    [u.error], eax
0000FCFC E992C7FFFF             <1>       jmp    error
                                <1> sndc_init0:
0000FD01 A2[AD650100]           <1>       mov    [audio_user], al
                                <1> sndc_init1:
0000FD06 8915[B0650100]         <1>       mov    [audio_cb_addr], edx
0000FD0C 881D[AE650100]         <1>       mov    [audio_cb_mode], bl
0000FD12 880D[AF650100]         <1>       mov    [audio_srb], cl
                                <1>
                                <1>       ; 24/04/2017
0000FD18 803D[85650100]03       <1>       cmp    byte [audio_device], 3 ; VT8233 (VT8237R)
0000FD1F 7438                   <1>       je     short sndc_init_9
                                <1>       ;ja    short soundc_respond_err ; temporary (28/05/2017)
0000FD21 803D[85650100]01       <1>       cmp    byte [audio_device], 1 ; SB 16
0000FD28 7510                   <1>       jne    short sndc_init_13
0000FD2A BB[F71A0100]           <1>       mov    ebx, sb16_int_handler
                                <1>       ; Note: 'SbInit' is at 'Start to Play' stage
                                <1>       ; 20/05/2017
0000FD2F 66C705[BA650100]08-    <1>       mov    word [audio_master_volume], 0808h ; 2/8
0000FD37 08                     <1>
0000FD38 EB3F                   <1>       jmp    short sndc_init_10
                                <1> sndc_init_13:
                                <1>       ; 28/05/2017
0000FD3A 803D[85650100]02       <1>       cmp    byte [audio_device], 2 ; AC 97 (ICH)
0000FD41 0F8569FEFFFF           <1>       jne    soundc_respond_err ; temporary (28/05/2017)
                                <1>
0000FD47 E8001F0000             <1>       call   ac97_codec_config
0000FD4C 0F825EFEFFFF           <1>       jc     soundc_respond_err ; codec error !
                                <1>
0000FD52 BB[331E0100]           <1>       mov    ebx, ac97_int_handler
```

```
2301 0000FD57 EB20              <1>       jmp    short sndc_init_10
2302                            <1>
2303                            <1> sndc_init_9:
2304                            <1>       ;call  reset_codec
2305                            <1>       ;; eax = 1
2306                            <1>       ;call  codec_io_w16 ; w32
2307 0000FD59 E8BD170000        <1>       call   init_codec ; 28/05/2017
2308 0000FD5E 0F824CFEFFFF      <1>       jc     soundc_respond_err ; codec error !
2309                            <1>
2310 0000FD64 E8EE190000        <1>       call   channel_reset
2311                            <1>
2312                            <1>       ; setup the Codec (actually mixer registers)
2313 0000FD69 E8F8180000        <1>         call    codec_config  ; unmute codec, set rates.
2314 0000FD6E 0F823CFEFFFF      <1>       jc     soundc_respond_err ; codec error !
2315                            <1>
2316 0000FD74 BB[D3160100]      <1>       mov    ebx, vt8233_int_handler
2317                            <1> sndc_init_10:
2318                            <1>       ; 13/04/2017
2319 0000FD79 A0[87650100]      <1>       mov    al, [audio_intr] ; IRQ number
2320 0000FD7E E832FDFFFF        <1>       call   set_dev_IRQ_service
2321                            <1>
2322                            <1>       ; SETUP (audio) INTERRUPT CALLBACK SERVICE
2323 0000FD83 8A1D[87650100]    <1>       mov    bl, [audio_intr] ; IRQ number
2324 0000FD89 8A3D[AE650100]    <1>       mov    bh, [audio_cb_mode]
2325 0000FD8F FEC7              <1>       inc    bh  ; 1 = Signal Response Byte method (fixed value)
2326                            <1>                ; 2 = Callback service method
2327                            <1>                ; 3 = Auto Increment S.R.B. method
2328 0000FD91 8A0D[AF650100]    <1>       mov    cl, [audio_srb]
2329 0000FD97 8B15[B0650100]    <1>       mov    edx, [audio_cb_addr]
2330 0000FD9D A0[AD650100]      <1>       mov    al, [audio_user]
2331                            <1>       ; 14/04/2017
2332 0000FDA2 E8DB040000        <1>       call   set_irq_callback_service
2333                            <1>       ; 16/04/2017
2334 0000FDA7 A3[64030300]      <1>       mov    [u.r0], eax
2335                            <1>       ;jnc   sysret
2336 0000FDAC 7316              <1>       jnc    short sndc_init2 ; 21/04/2017
2337                            <1>       ;
2338 0000FDAE A3[C8030300]      <1>       mov    dword [u.error], eax
2339                            <1>
2340 0000FDB3 A0[87650100]      <1>       mov    al, [audio_intr] ; IRQ number
2341 0000FDB8 31DB              <1>       xor    ebx, ebx ; reset IRQ handler address
2342 0000FDBA E8F6FCFFFF        <1>       call   set_dev_IRQ_service
2343                            <1>
2344 0000FDBF E9CFC6FFFF        <1>       jmp    error
2345                            <1>
2346                            <1> sndc_init2:
2347                            <1>       ; 21/04/2017
2348 0000FDC4 8B0D[A0650100]    <1>       mov    ecx, [audio_buff_size] ; audio buffer size
2349 0000FDCA D1E1              <1>       shl    ecx, 1 ; *2
2350 0000FDCC A1[A4650100]      <1>       mov    eax, [audio_dma_buff]
2351 0000FDD1 21C0              <1>       and    eax, eax
2352 0000FDD3 7415              <1>       jz     short sndc_init3
2353                            <1>
2354 0000FDD5 8B15[A8650100]    <1>       mov    edx, [audio_dmabuff_size] ; dma buffer size
2355 0000FDDB 39D1              <1>       cmp    ecx, edx
2356 0000FDDD 744D              <1>       je     short sndc_init5
2357                            <1>
2358 0000FDDF 87CA              <1>       xchg   ecx, edx
2359 0000FDE1 E84A58FFFF        <1>       call   deallocate_memory_block
2360 0000FDE6 87D1              <1>       xchg   edx, ecx
2361 0000FDE8 31C0              <1>       xor    eax, eax
2362                            <1> sndc_init3:
2363                            <1>       ; 12/05/2017
2364 0000FDEA 803D[85650100]01  <1>       cmp    byte [audio_device], 1 ; SB 16
2365 0000FDF1 7515              <1>       jne    short sndc_init4
2366 0000FDF3 C705[A4650100]-   <1>       mov    dword [audio_dma_buff], sb16_dma_buffer
2366 0000FDF9 [00000200]        <1>
2367 0000FDFD C705[A8650100]0000- <1>     mov    dword [audio_dmabuff_size], 65536
2367 0000FE05 0100              <1>
2368                            <1>       ;xor   eax, eax
2369                            <1>       ;mov   [u.r0], eax ; 0 = no error, successful
2370 0000FE07 C3                <1>       retn
2371                            <1>
2372                            <1> sndc_init4:
2373                            <1>       ; EAX = Beginning address (physical)
2374                            <1>       ; EAX = 0 -> Allocate mem block from the 1st proper aperture
2375                            <1>       ; ECX = Number of bytes to be allocated (>0)
2376 0000FE08 E81656FFFF        <1>       call   allocate_memory_block
2377 0000FE0D 0F8296FDFFFF      <1>       jc     sound_buff_error
2378                            <1>
2379                            <1>       ; set dma buffer address and size parameters
2380 0000FE13 A3[A4650100]      <1>       mov    [audio_dma_buff], eax ; dma buffer address
2381 0000FE18 890D[A8650100]    <1>       mov    [audio_dmabuff_size], ecx ; dma buffer size
2382                            <1> ;     ; EAX = Beginning (physical) addr of the allocated mem block
2383                            <1> ;     ; ECX = Num of allocated bytes (rounded up to page borders)
2384                            <1> ;     cmp    byte [audio_pci], 0 ; AC97 audio controller ?
2385                            <1> ;     ja     short sndc_init4
2386                            <1> ;
2387                            <1> ;     ; Sound Blaster 16 uses classic DMA
2388                            <1> ;     mov    edx, eax
2389                            <1> ;     add    edx, ecx
2390                            <1> ;     cmp    edx, 1000000h ; 1st 16 MB
2391                            <1> ;     jna    short sndc_init4
2392                            <1> ;
2393                            <1> ;     ; error !
2394                            <1> ;     ; restore Memory Allocation Table Content
2395                            <1> ;     ; EAX = Beginning address (physical)
2396                            <1> ;     ; ECX = Number of bytes to be deallocated
2397                            <1> ;     call   deallocate_memory_block
2398                            <1> ;     ; reset dma buffer address and size parameters
2399                            <1> ;     xor    eax, eax ; 0
2400                            <1> ;     mov    [audio_dma_buff], eax ; 0
```

```
2401                                    <1> ;     mov   [audio_dmabuff_size], ecx ; 0
2402                                    <1> ;     jmp   sound_buff_error
2403                                    <1> ;
2404                                    <1> ;sndc_init4:
2405 0000FE1E 803D[85650100]03         <1>       cmp   byte [audio_device], 3
2406                                    <1>       ;jne  short sndc_init5
2407 0000FE25 7506                     <1>       jne   short sndc_init14 ; 28/05/2017
2408 0000FE27 E86C190000               <1>       call  set_vt8233_bdl
2409                                    <1> sndc_init5:
2410                                    <1>       ;sub  eax, eax ; 0
2411                                    <1>       ;mov  [u.r0], eax ; 0 = no error, successful
2412 0000FE2C C3                       <1>       retn
2413                                    <1> sndc_init14:
2414 0000FE2D E8331F0000               <1>       call  set_ac97_bdl
2415                                    <1>       ;jmp  short sndc_init5
2416 0000FE32 C3                       <1>       retn
2417                                    <1>
2418                                    <1> sound_play:
2419                                    <1>       ; FUNCTION = 4
2420                                    <1>       ; bl = Mode
2421                                    <1>       ;      bit 0 = mono/stereo (1 = stereo)
2422                                    <1>       ;      bit 1 = 8 bit / 16 bit (1 = 16 bit)
2423                                    <1>       ; cx = Sampling Rate (Hz)
2424                                    <1>
2425                                    <1>       ; 13/06/2017
2426                                    <1>       ; Note: Even if Mode bits are not 11b,
2427                                    <1>       ;     AC'97 Audio Controller (&Codec)
2428                                    <1>       ;     will play audio samples as 16 bit, stereo
2429                                    <1>       ;     samples.
2430                                    <1>       ;     (Program must fill the audio buffer
2431                                    <1>       ;     as required; 8 bit samples must be converted
2432                                    <1>       ;     to 16 bit samples and mono samples must be
2433                                    <1>       ;     converted to stereo samples...)
2434                                    <1>       ;
2435                                    <1>       ; 28/05/2017
2436                                    <1>       ; 15/05/2017, 20/05/2017
2437                                    <1>       ; 21/04/2017, 24/04/2017
2438                                    <1>       ; ... device check at first
2439 0000FE33 A0[85650100]             <1>       mov   al, [audio_device]
2440 0000FE38 08C0                     <1>       or    al, al ; 0 ; pc speaker or invalid
2441 0000FE3A 0F845B1FFFFF             <1>       jz    beeper_gfx ; 'video.s' ; temporary !
2442                                    <1> ;     cmp   al, 3 ; VIA VT 8237R (vt8233)
2443                                    <1> ;     je    short snd_play_1
2444                                    <1> ;     cmp   al, 1 ; SB 16
2445                                    <1> ;     jne   soundc_dev_err ; temporary !
2446                                    <1> ;snd_play_0:
2447                                    <1>       ; ... buffer & (buffer) owner check at second
2448 0000FE40 833D[98650100]00         <1>       cmp   dword [audio_buffer], 0
2449 0000FE47 0F865CFDFFFF             <1>       jna   sound_buff_error
2450 0000FE4D A0[B3030300]             <1>       mov   al, [u.uno]
2451 0000FE52 3A05[AD650100]           <1>       cmp   al, [audio_user]
2452 0000FE58 0F858FFEFFFF             <1>       jne   sndc_owner_error
2453                                    <1>
2454 0000FE5E 66890D[B6650100]         <1>       mov   [audio_freq], cx ; sample frequency (Hertz)
2455 0000FE65 88D8                     <1>       mov   al, bl
2456 0000FE67 2401                     <1>       and   al, 1 ; mono/stereo (1= stereo)
2457 0000FE69 FEC0                     <1>       inc   al ; channels
2458 0000FE6B A2[B5650100]             <1>       mov   [audio_stmo], al ; sound channels (1 or 2)
2459 0000FE70 B008                     <1>       mov   al, 8
2460 0000FE72 F6C302                   <1>       test  bl, 2 ; bits per sample (1= 16 bit)
2461 0000FE75 7402                     <1>       jz    short snd_play_bps
2462 0000FE77 D0E0                     <1>       shl   al, 1
2463                                    <1> snd_play_bps:
2464 0000FE79 A2[B4650100]             <1>       mov   [audio_bps], al
2465                                    <1>       ; Transfer ring 3 (user's) audio buffer content to dma buffer
2466 0000FE7E 8B3D[A4650100]           <1>       mov   edi, [audio_dma_buff] ; dma buffer (ring 0)
2467 0000FE84 09FF                     <1>       or    edi, edi
2468 0000FE86 0F841DFDFFFF             <1>       jz    sound_buff_error
2469 0000FE8C 8B35[9C650100]           <1>       mov   esi, [audio_p_buffer] ; physical address (ring 3)
2470 0000FE92 8B0D[A0650100]           <1>       mov   ecx, [audio_buff_size] ; 15/05/2017
2471                                    <1>       ;rep  movsb
2472 0000FE98 C1E902                   <1>       shr   ecx, 2
2473 0000FE9B F3A5                     <1>       rep   movsd
2474                                    <1>       ; 20/05/2017
2475 0000FE9D C605[AC650100]01         <1>       mov   byte [audio_flag], 1 ; next half (on next time)
2476                                    <1>
2477                                    <1>       ; 24/04/2017
2478 0000FEA4 A0[85650100]             <1>       mov   al, [audio_device]
2479 0000FEA9 3C03                     <1>       cmp   al, 3 ; VT8233 (VT8237R)
2480 0000FEAB 7410                     <1>       je    short snd_play_1
2481 0000FEAD 3C01                     <1>       cmp   al, 1 ; Sound Blaster 16
2482 0000FEAF 7512                     <1>       jne   short snd_play_2  ; 28/05/2017
2483 0000FEB1 E8EA1A0000               <1>       call  SbInit_play
2484 0000FEB6 0F82F4FCFFFF             <1>       jc    soundc_respond_err
2485 0000FEBC C3                       <1>       retn
2486                                    <1>
2487                                    <1> snd_play_1:
2488 0000FEBD E806190000               <1>       call  vt8233_start_play
2489 0000FEC2 C3                       <1>       retn
2490                                    <1>
2491                                    <1> snd_play_2:
2492                                    <1>       ; 28/05/2017
2493                                    <1>       ;cmp  al, 2 ; AC'97
2494                                    <1>       ;jne  short snd_play_3
2495                                    <1>
2496 0000FEC3 E8D11E0000               <1>       call  ac97_start_play
2497 0000FEC8 C3                       <1>       retn
2498                                    <1>
2499                                    <1> ;snd_play_3:
2500                                    <1> ;     ;call hda_start_play
2501                                    <1> ;     retn
2502                                    <1>
```

```
2503                            <1> sound_pause:
2504                            <1>       ; FUNCTION = 5
2505                            <1>       ; Pause
2506                            <1>       ; 28/05/2017
2507                            <1>       ; 24/04/2017
2508                            <1>       ; 22/04/2017
2509 0000FEC9 E814030000       <1>       call   snd_dev_check
2510 0000FECE 7275             <1>       jc     short snd_nothing ; temporary.
2511 0000FED0 E81A030000       <1>       call   snd_buf_check
2512 0000FED5 726E             <1>       jc     short snd_nothing ; temporary.
2513 0000FED7 A0[85650100]     <1>       mov    al, [audio_device]
2514 0000FEDC 3C03             <1>       cmp    al, 3 ; VIA VT 8237R (vt8233)
2515 0000FEDE 7409             <1>       je     short snd_pause_1
2516 0000FEE0 3C01             <1>       cmp    al, 1 ; Sound Blaster 16
2517 0000FEE2 750A             <1>       jne    short snd_pause_2 ; 28/05/2017
2518 0000FEE4 E9951C0000       <1>       jmp    sb16_pause
2519                            <1> snd_pause_1:
2520 0000FEE9 E998190000       <1>       jmp    vt8233_pause
2521                            <1> snd_pause_2:
2522                            <1>       ; 28/05/2017
2523                            <1>       ;cmp   al, 2 ; AC'97
2524                            <1>       ;jne   short snd_nothing ; temporary.
2525 0000FEEE E934200000       <1>       jmp    ac97_pause
2526                            <1>
2527                            <1> sound_continue:
2528                            <1>       ; FUNCTION = 6
2529                            <1>       ; Continue to play
2530                            <1>       ; 28/05/2017
2531                            <1>       ; 22/04/2017
2532 0000FEF3 E8EA020000       <1>       call   snd_dev_check
2533 0000FEF8 724B             <1>       jc     short snd_nothing ; temporary.
2534 0000FEFA E8F0020000       <1>       call   snd_buf_check
2535 0000FEFF 7244             <1>       jc     short snd_nothing ; temporary.
2536 0000FF01 A0[85650100]     <1>       mov    al, [audio_device]
2537 0000FF06 3C03             <1>       cmp    al, 3 ; VIA VT 8237R (vt8233)
2538 0000FF08 7409             <1>       je     short snd_cont_1
2539 0000FF0A 3C01             <1>       cmp    al, 1 ; Sound Blaster 16
2540 0000FF0C 750A             <1>       jne    short snd_cont_2 ; 28/05/2017
2541 0000FF0E E98E1C0000       <1>       jmp    sb16_continue
2542                            <1> snd_cont_1:
2543 0000FF13 E91B190000       <1>       jmp    vt8233_play
2544                            <1> snd_cont_2:
2545                            <1>       ; 28/05/2017
2546                            <1>       ;cmp   al, 2 ; AC'97
2547                            <1>       ;jne   short snd_nothing ; temporary.
2548 0000FF18 E9D21E0000       <1>       jmp    ac97_play
2549                            <1>
2550                            <1> sound_stop:
2551                            <1>       ; FUNCTION = 7
2552                            <1>       ; Stop playing
2553                            <1>       ; 28/05/2017
2554                            <1>       ; 24/05/2017
2555                            <1>       ; 21/04/2017, 22/04/2017, 24/04/2017
2556 0000FF1D E8C0020000       <1>       call   snd_dev_check
2557 0000FF22 7221             <1>       jc     short snd_nothing ; temporary.
2558                            <1>       ;call  snd_buf_check
2559 0000FF24 E8CF020000       <1>       call   snd_user_check ; 24/05/2017
2560 0000FF29 721A             <1>       jc     short snd_nothing ; temporary.
2561                            <1>
2562 0000FF2B A0[85650100]     <1>       mov    al, [audio_device]
2563 0000FF30 3C03             <1>       cmp    al, 3 ; VIA VT 8237R (vt8233)
2564 0000FF32 0F8457180000     <1>       je     vt8233_stop
2565                            <1>       ; 28/05/2017
2566                            <1>       ;ja    short snd_nothing
2567 0000FF38 3C01             <1>       cmp    al, 1 ; Sound Blaster 16
2568 0000FF3A 0F84841C0000     <1>       je     sb16_stop
2569                            <1>       ;cmp   al, 2
2570                            <1>       ;je    short ac97_stop
2571 0000FF40 E9B41F0000       <1>       jmp    ac97_stop ; temporary.
2572                            <1>       ;jmp   hda_stop
2573                            <1>
2574                            <1> snd_nothing:
2575                            <1>       ; 21/04/2017
2576 0000FF45 C3               <1>       retn
2577                            <1>
2578                            <1> soundc_reset:
2579                            <1>       ; FUNCTION = 8
2580                            <1>       ; Reset Audio Controller
2581                            <1>       ; 28/05/2017
2582                            <1>       ; 22/04/2017
2583 0000FF46 E897020000       <1>       call   snd_dev_check
2584 0000FF4B 72F8             <1>       jc     snd_nothing ; temporary.
2585 0000FF4D E89D020000       <1>       call   snd_buf_check
2586 0000FF52 72F1             <1>       jc     snd_nothing ; temporary.
2587                            <1>
2588 0000FF54 A0[85650100]     <1>       mov    al, [audio_device]
2589 0000FF59 3C03             <1>       cmp    al, 3 ; VIA VT 8237R (vt8233)
2590 0000FF5B 0F8433190000     <1>       je     vt8233_reset
2591 0000FF61 77E2             <1>       ja     short snd_nothing ; temporary.
2592                            <1>       ;ja    hda_reset
2593 0000FF63 3C01             <1>       cmp    al, 1 ; Sound Blaster 16
2594 0000FF65 0F850D200000     <1>       jne    ac97_reset
2595 0000FF6B E8A61C0000       <1>       call   sb16_reset
2596 0000FF70 0F823AFCFFFF     <1>       jc     soundc_respond_err
2597 0000FF76 C3               <1>       retn
2598                            <1>
2599                            <1> soundc_cancel:
2600                            <1>       ; FUNCTION = 9
2601                            <1>       ; Cancel audio callback service
2602                            <1>       ; 22/04/2017
2603 0000FF77 A0[AD650100]     <1>       mov    al, [audio_user]
2604 0000FF7C 3A05[B3030300]   <1>       cmp    al, [u.uno]
```

```
2605 0000FF82 75C1                <1>       jne    short snd_nothing
2606                              <1>       ; RESET (audio) INTERRUPT CALLBACK SERVICE
2607 0000FF84 8A1D[87650100]      <1>       mov    bl, [audio_intr] ; IRQ number
2608 0000FF8A A0[B3030300]        <1>       mov    al, [u.uno]
2609 0000FF8F 28FF                <1>       sub    bh, bh ; 0 ; unlink IRQ from user service
2610 0000FF91 E8EC020000          <1>       call   set_irq_callback_service
2611 0000FF96 0F8256FDFFFF        <1>       jc     sndc_perm_error ; 'permission denied' error
2612 0000FF9C C3                  <1>       retn
2613                              <1>
2614                              <1> sound_dalloc:
2615                              <1>       ; FUNCTION = 10
2616                              <1>       ; Deallocate (ring 3) audio buffer
2617                              <1>       ; 22/04/2017
2618 0000FF9D A0[AD650100]        <1>       mov    al, [audio_user]
2619 0000FFA2 3A05[B3030300]      <1>       cmp    al, [u.uno]
2620 0000FFA8 759B                <1>       jne    short snd_nothing
2621 0000FFAA 8B1D[98650100]      <1>       mov    ebx, [audio_buffer]
2622                              <1>       ;or    ebx, ebx
2623                              <1>       ;jz    short snd_nothing
2624 0000FFB0 8B0D[A0650100]      <1>       mov    ecx, [audio_buff_size]
2625 0000FFB6 E8C357FFFF          <1>       call   deallocate_user_pages
2626 0000FFBB 31C0                <1>       xor    eax, eax
2627 0000FFBD A3[98650100]        <1>       mov    [audio_buffer], eax ; 0
2628 0000FFC2 A2[AD650100]        <1>       mov    [audio_user], al ; 0
2629 0000FFC7 C3                  <1>       retn
2630                              <1>
2631                              <1> sound_volume:
2632                              <1>       ; FUNCTION = 11
2633                              <1>       ; Set sound volume level
2634                              <1>       ; 28/05/2017
2635                              <1>       ; 20/05/2017
2636                              <1>       ; 22/04/2017, 24/04/2017
2637                              <1>       ; bl = component (0 = master/playback/lineout volume)
2638                              <1>       ; cl = left channel volume level (0 to 31)
2639                              <1>       ; ch = right channel volume level (0 to 31)
2640                              <1>
2641 0000FFC8 80FB80              <1>       cmp    bl, 80h
2642 0000FFCB 720E                <1>       jb     short snd_vol_1
2643 0000FFCD 0F8772FFFFFF        <1>       ja     snd_nothing ; temporary.
2644                              <1>       ; Set volume level for next play (BL>= 80h)
2645 0000FFD3 66890D[BA650100]    <1>       mov    [audio_master_volume], cx
2646 0000FFDA C3                  <1>       retn
2647                              <1> snd_vol_1:
2648                              <1>       ; set volume level immediate (BL< 80h)
2649 0000FFDB 80FB00              <1>       cmp    bl, 0
2650 0000FFDE 0F8761FFFFFF        <1>       ja     snd_nothing ; temporary.
2651                              <1>
2652 0000FFE4 E8F9010000          <1>       call   snd_dev_check
2653 0000FFE9 0F8256FFFFFF        <1>       jc     snd_nothing ; temporary.
2654 0000FFEF E8FB010000          <1>       call   snd_buf_check
2655 0000FFF4 0F824BFFFFFF        <1>       jc     snd_nothing ; temporary.
2656                              <1>
2657 0000FFFA A0[85650100]        <1>       mov    al, [audio_device]
2658 0000FFFF 3C03                <1>       cmp    al, 3 ; VIA VT 8237R (vt8233)
2659 00010001 0F84A6180000        <1>       je     vt8233_volume
2660                              <1>       ; 28/05/2017
2661 00010007 0F8738FFFFFF        <1>       ja     snd_nothing ; temporary.
2662                              <1>       ;ja    hda_volume
2663                              <1>       ; Sound Blaster 16
2664 0001000D 3C01                <1>       cmp    al, 1 ; SB 16
2665 0001000F 0F84341B0000        <1>       je     sb16_volume
2666 00010015 E9F11D0000          <1>       jmp    ac97_volume
2667                              <1>
2668                              <1> soundc_disable:
2669                              <1>       ; FUNCTION = 12
2670                              <1>       ; Disable audio device (and unlink DMA memory)
2671                              <1>       ; 28/05/2017
2672                              <1>       ; 24/05/2017
2673                              <1>       ; 22/04/2017
2674 0001001A E8C3010000          <1>       call   snd_dev_check
2675 0001001F 0F827DFBFFFF        <1>       jc     soundc_dev_err ; temporary.
2676                              <1>       ;call  snd_buf_check
2677                              <1>       ;jc    sndc_owner_error ; temporary.
2678                              <1>
2679 00010025 A0[85650100]        <1>       mov    al, [audio_device]
2680 0001002A 3C03                <1>       cmp    al, 3 ; VIA VT 8237R (vt8233)
2681 0001002C 7418                <1>       je     short snd_disable_1
2682 0001002E 0F8711FFFFFF        <1>       ja     snd_nothing ; temporary.
2683 00010034 3C01                <1>       cmp    al, 1 ; Sound Blaster 16
2684 00010036 7507                <1>       jne    short snd_disable_0
2685 00010038 E8871B0000          <1>       call   sb16_stop
2686 0001003D EB0C                <1>       jmp    short snd_disable_2
2687                              <1> snd_disable_0:
2688 0001003F E8B51E0000          <1>       call   ac97_stop
2689 00010044 EB05                <1>       jmp    short snd_disable_2
2690                              <1> snd_disable_1:
2691 00010046 E844170000          <1>       call   vt8233_stop
2692                              <1> snd_disable_2:
2693 0001004B A0[87650100]        <1>       mov    al, [audio_intr]
2694 00010050 29DB                <1>       sub    ebx, ebx ; 0 = reset
2695 00010052 E85EFAFFFF          <1>       call   set_dev_IRQ_service
2696                              <1>
2697                              <1>       ;mov   al, [audio_intr]
2698 00010057 28E4                <1>       sub    ah, ah ; 0 = reset
2699 00010059 E8C0F6FFFF          <1>       call   set_hardware_int_vector
2700                              <1>
2701 0001005E 31C0                <1>       xor    eax, eax
2702 00010060 A2[85650100]        <1>       mov    byte [audio_device], al
2703 00010065 A2[87650100]        <1>       mov    byte [audio_intr], al
2704 0001006A 8705[A4650100]      <1>       xchg   eax, [audio_dma_buff]
2705                              <1>       ; 24/05/2017
2706                              <1>       ;or    eax, eax
```

```
2707                                    <1>       ;jz    short snd_disable_3
2708                                    <1>       ;cmp   eax, sb16_dma_buffer ; default DMA buffer
2709                                    <1>       ;je    short snd_disable_3
2710 00010070 803D[84650100]00         <1>       cmp    byte [audio_pci], 0 ; AC97 audio controller ?
2711 00010077 7612                     <1>       jna    short snd_disable_3
2712 00010079 C605[84650100]00         <1>       mov    byte [audio_pci], 0
2713                                    <1>       ;sub   ecx, ecx
2714                                    <1>       ;xchg  ecx, [audio_dmabuff_size]
2715 00010080 8B0D[A8650100]           <1>       mov    ecx, [audio_dmabuff_size]
2716 00010086 E8A555FFFF               <1>       call   deallocate_memory_block
2717                                    <1> snd_disable_3:
2718 0001008B C3                       <1>       retn
2719                                    <1>
2720                                    <1> sound_dma_map:
2721                                    <1>       ; FUNCTION = 13
2722                                    <1>       ; Map audio dma buff addr to user's buffer addr
2723                                    <1>       ; 12/05/2017
2724 0001008C 21C9                     <1>       and    ecx, ecx
2725 0001008E 0F8415FBFFFF             <1>       jz     sound_buff_error
2726 00010094 803D[85650100]01         <1>       cmp    byte [audio_device], 1
2727 0001009B 7229                     <1>       jb     short snd_dma_map_1
2728                                    <1> snd_dma_map_0:
2729 0001009D A1[A4650100]             <1>       mov    eax, [audio_dma_buff]
2730 000100A2 21C0                     <1>       and    eax, eax
2731 000100A4 7420                     <1>       jz     short snd_dma_map_1
2732                                    <1>       ;
2733 000100A6 8A1D[AD650100]           <1>       mov    bl, [audio_user]
2734 000100AC 08DB                     <1>       or     bl, bl
2735 000100AE 7416                     <1>       jz     short snd_dma_map_1
2736 000100B0 3A1D[B3030300]           <1>       cmp    bl, [u.uno]
2737 000100B6 0F8531FCFFFF             <1>       jne    sndc_owner_error
2738                                    <1>       ;
2739 000100BC 8B1D[A8650100]           <1>       mov    ebx, [audio_dmabuff_size]
2740 000100C2 21DB                     <1>       and    ebx, ebx
2741 000100C4 750A                     <1>       jnz    short snd_dma_map_2
2742                                    <1> snd_dma_map_1:
2743 000100C6 B8[00000200]             <1>       mov    eax, sb16_dma_buffer
2744 000100CB BB00000100               <1>       mov    ebx, 65536
2745                                    <1> snd_dma_map_2:
2746 000100D0 81C1FF0F0000             <1>       add    ecx, PAGE_SIZE-1 ; 4095
2747 000100D6 6681E100F0               <1>       and    cx, ~PAGE_OFF ; not 4095
2748 000100DB 39D9                     <1>       cmp    ecx, ebx
2749 000100DD 0F87C6FAFFFF             <1>       ja     sound_buff_error
2750 000100E3 50                       <1>       push   eax
2751 000100E4 89D3                     <1>       mov    ebx, edx
2752 000100E6 C1E90C                   <1>       shr    ecx, 12 ; byte count to page count
2753                                    <1>       ; eax = physical address of (audio) dma buffer
2754                                    <1>       ; ebx = virtual address of (audio) dma buffer (user's pgdir)
2755                                    <1>       ; ecx = page count (>0)
2756 000100E9 E8B255FFFF               <1>       call   direct_memory_access
2757 000100EE 58                       <1>       pop    eax
2758 000100EF 0F82B4FAFFFF             <1>       jc     sound_buff_error
2759 000100F5 A3[64030300]             <1>       mov    [u.r0], eax
2760 000100FA C3                       <1>       retn
2761                                    <1>
2762                                    <1> soundc_info:
2763                                    <1>       ; FUNCTION = 14
2764                                    <1>       ; Get Audio Controller Info
2765                                    <1>       ; 10/06/2017
2766                                    <1>       ; 05/06/2017
2767 000100FB 20DB                     <1>       and    bl, bl ; 0
2768 000100FD 740A                     <1>       jz     short sndc_info_0
2769                                    <1>       ; invalid parameter !
2770 000100FF B817000000               <1>       mov    eax, ERR_INV_PARAMETER ; 23
2771                                    <1> ;sndc_inf_error:
2772                                    <1> ;     mov    [u.r0], eax
2773                                    <1> ;     mov    [u.error], eax
2774                                    <1> ;     jmp    error
2775 00010104 E9ACFAFFFF               <1>       jmp    sysaudio_err
2776                                    <1>
2777                                    <1> sndc_info_0:
2778 00010109 E8D4000000               <1>       call   snd_dev_check
2779 0001010E 0F828EFAFFFF             <1>       jc     soundc_dev_err
2780                                    <1>
2781 00010114 8B1D[90650100]           <1>       mov    ebx, [audio_vendor]
2782 0001011A 8B0D[8C650100]           <1>       mov    ecx, [audio_dev_id]
2783                                    <1>       ;mov   al,  [audio_device]
2784 00010120 3C02                     <1>       cmp    al, 2 ; AC'97 (ICH)
2785 00010122 7513                     <1>       jne    short sndc_info_1
2786                                    <1>       ; Intel AC97 (ICH) Audio Controller (=2)
2787 00010124 668B15[BE650100]         <1>       mov    dx, [NABMBAR]
2788 0001012B C1E210                   <1>       shl    edx, 16
2789 0001012E 668B15[BC650100]         <1>       mov    dx, [NAMBAR]
2790 00010135 EB07                     <1>       jmp    short sndc_info_2
2791                                    <1> sndc_info_1:
2792                                    <1>       ; 05/06/2017
2793                                    <1>       ; Note: Intel HDA code (here) is not ready yet!
2794                                    <1>       ; !!! SB16 or VT8233 (VT8237R) !!!
2795 00010137 0FB715[8A650100]         <1>       movzx  edx, word [audio_io_base]
2796                                    <1> sndc_info_2:
2797 0001013E 88C4                     <1>       mov    ah, al ;  [audio_device]
2798 00010140 A0[87650100]             <1>       mov    al, [audio_intr]
2799                                    <1>
2800                                    <1>       ; EAX = IRQ Number in AL
2801                                    <1>       ;     Audio Device Number in AH
2802                                    <1>       ; EBX = DEV/VENDOR ID
2803                                    <1>       ;     (DDDDDDDDDDDDDDDDVVVVVVVVVVVVVVVV)
2804                                    <1>       ; ECX = BUS/DEV/FN
2805                                    <1>       ;     (00000000BBBBBBBBDDDDDFFF00000000)
2806                                    <1>       ; EDX = NABMBAR/NAMBAR (for AC97)
2807                                    <1>       ;     (Low word, DX = NAMBAR address)
2808                                    <1>       ; EDX = Base IO Addr (DX) for SB16 & VT8233
```

```
2809                                    <1>
2810                                    <1>       ; 10/06/2017
2811 00010145 A3[64030300]              <1>       mov    [u.r0], eax
2812 0001014A 8B2D[60030300]            <1>       mov    ebp, [u.usp]
2813 00010150 895D10                    <1>       mov    [ebp+16], ebx  ; ebx
2814 00010153 895514                    <1>       mov    [ebp+20], edx  ; edx
2815 00010156 894D18                    <1>       mov    [ebp+24], ecx  ; ecx
2816                                    <1>
2817 00010159 C3                        <1>       retn
2818                                    <1>
2819                                    <1> sound_data:
2820                                    <1>       ; FUNCTION = 15
2821                                    <1>       ; Get Current Sound data for graphics
2822                                    <1>       ; 22/06/2017
2823                                    <1>       ;
2824 0001015A E883000000                <1>       call   snd_dev_check
2825 0001015F 0F823DFAFFFF              <1>       jc     soundc_dev_err ; Device not ready !
2826                                    <1>
2827 00010165 80FB00                    <1>       cmp    bl, 0
2828 00010168 760A                      <1>       jna    short sound_data_0
2829                                    <1>
2830                                    <1>       ; Only PCM OUT buffer data is valid for now!
2831 0001016A B817000000                <1>       mov    eax, ERR_INV_PARAMETER  ; 23
2832 0001016F E941FAFFFF                <1>       jmp    sysaudio_err
2833                                    <1>
2834                                    <1> sound_data_0:
2835 00010174 A1[A4650100]              <1>       mov    eax, [audio_dma_buff]
2836 00010179 09C0                      <1>       or     eax, eax
2837 0001017B 0F8428FAFFFF              <1>       jz     sound_buff_error
2838                                    <1>
2839 00010181 803D[85650100]04          <1>       cmp    byte [audio_device], 4 ; Intel HDA
2840 00010188 744F                      <1>       je     short sound_data_4 ; temporary ! (22/06/2017)
2841                                    <1>
2842 0001018A 21C9                      <1>       and    ecx, ecx
2843                                    <1>       ;jnz   short sound_data_1 ; sample tranfer
2844                                    <1>
2845                                    <1>       ; Return only DMA Buffer pointer/offset...
2846                                    <1>       ; (If DMA Buffer has been mapped to user's
2847                                    <1>       ;  memory space; program can get graphics
2848                                    <1>       ;  data by using only this pointer value.)
2849                                    <1>
2850                                    <1>       ;call  get_dma_buffer_offset
2851                                    <1>       ;; eax = DMA buffer offset
2852                                    <1>       ;;     (!not half buffer offset!)
2853                                    <1>       ;mov   [u.r0], eax
2854                                    <1>       ;retn
2855                                    <1>
2856 0001018C 0F84461F0000              <1>       jz     get_dma_buffer_offset
2857                                    <1>
2858                                    <1> sound_data_1:
2859                                    <1>       ;mov   eax, [audio_dmabuff_size]
2860                                    <1>       ;shr   eax, 1 ; half buffer size
2861                                    <1>       ;cmp   ecx, eax
2862                                    <1>       ;ja    short sound_buff_error
2863                                    <1>
2864 00010192 3B0D[A8650100]            <1>       cmp    ecx, [audio_dmabuff_size]
2865 00010198 0F870BFAFFFF              <1>       ja     sound_buff_error
2866                                    <1>
2867 0001019E 89D0                      <1>       mov    eax, edx
2868 000101A0 25FF0F0000                <1>       and    eax, PAGE_OFF ; 4095 (0FFFh)
2869 000101A5 81F900100000              <1>       cmp    ecx, 4096
2870 000101AB 7605                      <1>       jna    short sound_data_2
2871 000101AD B900100000                <1>       mov    ecx, 4096 ; max. 1 page
2872                                    <1> sound_data_2:
2873 000101B2 01C8                      <1>       add    eax, ecx
2874 000101B4 3D00100000                <1>       cmp    eax, 4096
2875 000101B9 7606                      <1>       jna    short sound_data_3
2876 000101BB 6625FF0F                  <1>       and    ax, PAGE_OFF ; 4095 (0FFFh)
2877 000101BF 29C1                      <1>       sub    ecx, eax
2878                                    <1>       ; here, ECX has been adjusted to fit
2879                                    <1>       ;      in page border..  (<= 4096, >0)
2880                                    <1> sound_data_3:
2881 000101C1 51                        <1>       push   ecx
2882 000101C2 52                        <1>       push   edx
2883 000101C3 89D3                      <1>       mov    ebx, edx
2884 000101C5 E8C450FFFF                <1>       call   get_physical_addr
2885 000101CA 5A                        <1>       pop    edx
2886 000101CB 59                        <1>       pop    ecx
2887 000101CC 0F82D7F9FFFF              <1>       jc     sound_buff_error
2888                                    <1>
2889                                    <1>       ; eax = physical address of user's buffer
2890 000101D2 89C3                      <1>       mov    ebx, eax
2891                                    <1>       ; ecx = byte (transfer) count
2892                                    <1>       ;call  get_current_sound_data
2893                                    <1>       ;retn
2894 000101D4 E9741E0000                <1>       jmp    get_current_sound_data
2895                                    <1>
2896                                    <1> sound_data_4:
2897                                    <1>       ; Intel HDA code is not ready yet !
2898                                    <1>       ; 22/06/2017
2899 000101D9 31C0                      <1>       xor    eax, eax
2900 000101DB 48                        <1>       dec    eax
2901 000101DC A3[64030300]              <1>       mov    [u.r0], eax ; 0FFFFFFFFh
2902 000101E1 C3                        <1>       retn
2903                                    <1>
2904                                    <1> snd_dev_check:
2905                                    <1>       ; 10/06/2017
2906                                    <1>       ; 05/06/2017
2907                                    <1>       ; 24/05/2017
2908                                    <1>       ; 22/04/2017
2909                                    <1>       ; 21/04/2017
2910                                    <1>       ; ... device check at first
```

```
2911 000101E2 A0[85650100]        <1>       mov    al, [audio_device]
2912 000101E7 3C01                <1>       cmp    al, 1 ; SB 16
2913 000101E9 7203                <1>       jb     short snd_dev_chk_retn ; error !
2914                              <1>       ;cmp   al, 4 ; Intel HDA
2915                              <1>       ;ja    short snd_dbchk_stc ; invalid !
2916                              <1>       ; 10/06/2017
2917 000101EB 3C05                <1>       cmp    al, 5
2918 000101ED F5                  <1>       cmc
2919                              <1> snd_dev_chk_retn:
2920 000101EE C3                  <1>       retn
2921                              <1>
2922                              <1> snd_buf_check:
2923                              <1>       ; 10/06/2017
2924                              <1>       ; 22/04/2017
2925                              <1>       ; 21/04/2017
2926                              <1>       ; ... buffer & (buffer) owner check at second
2927 000101EF 833D[98650100]00    <1>       cmp    dword [audio_buffer], 0
2928 000101F6 760D                <1>       jna    short snd_dbchk_stc
2929                              <1> snd_user_check:
2930 000101F8 A0[B3030300]        <1>       mov    al, [u.uno]
2931 000101FD 3A05[AD650100]      <1>       cmp    al, [audio_user]
2932                              <1>       ;jne   short snd_dbchk_stc
2933                              <1>       ;retn
2934 00010203 74E9                <1>       je     short snd_dev_chk_retn
2935                              <1>
2936                              <1> snd_dbchk_stc:
2937 00010205 F9                  <1>       stc
2938 00010206 C3                  <1>       retn
2939                              <1>
2940                              <1> sound_update:
2941                              <1>       ; FUNCTION = 16
2942                              <1>       ; bl =
2943                              <1>       ;    0 = automatic (sequental) update (with flag switch!)
2944                              <1>       ;    1 = update dma half buffer 1 (without flag switch!)
2945                              <1>       ;    2 = update dma half buffer 2 (without flag switch!)
2946                              <1>       ;  FFh = get current flag value
2947                              <1>       ;      0 = dma half buffer 1 (will be played next)
2948                              <1>       ;      1 = dma half buffer 2 (will be played next)
2949                              <1>
2950                              <1>       ; 10/10/2017
2951                              <1>
2952                              <1>       ; ... device check at first
2953 00010207 A0[85650100]        <1>       mov    al, [audio_device]
2954 0001020C 08C0                <1>       or     al, al ; 0 ; pc speaker or invalid
2955 0001020E 0F848EF9FFFF        <1>       jz     soundc_dev_err
2956                              <1>
2957                              <1>       ; ... buffer & (buffer) owner check at second
2958 00010214 833D[98650100]00    <1>       cmp    dword [audio_buffer], 0
2959 0001021B 0F8688F9FFFF        <1>       jna    sound_buff_error
2960 00010221 A0[B3030300]        <1>       mov    al, [u.uno]
2961 00010226 3A05[AD650100]      <1>       cmp    al, [audio_user]
2962 0001022C 0F85BBFAFFFF        <1>       jne    sndc_owner_error
2963                              <1>
2964                              <1>       ; Transfer ring 3 (user's) audio buffer content to dma buffer
2965 00010232 8B3D[A4650100]      <1>       mov    edi, [audio_dma_buff] ; dma buffer (ring 0)
2966 00010238 09FF                <1>       or     edi, edi
2967 0001023A 0F8469F9FFFF        <1>       jz     sound_buff_error
2968 00010240 8B35[9C650100]      <1>       mov    esi, [audio_p_buffer] ; physical address (ring 3)
2969 00010246 8B0D[A0650100]      <1>       mov    ecx, [audio_buff_size]
2970                              <1>
2971                              <1>       ;movzx eax, byte [audio_flag]
2972 0001024C A0[AC650100]        <1>       mov    al, [audio_flag]
2973                              <1>
2974 00010251 FEC3                <1>       inc    bl
2975 00010253 7427                <1>       jz     short snd_update_3 ; bl = 0FFh
2976 00010255 FECB                <1>       dec    bl
2977 00010257 7411                <1>       jz     short snd_update_0 ; bl = 0
2978                              <1>
2979 00010259 80FB02              <1>       cmp    bl, 2
2980 0001025C 7417                <1>       je     short snd_update_1 ; dma half buffer 2
2981 0001025E 7217                <1>       jb     short snd_update_2 ; dma half buffer 1
2982                              <1>
2983                              <1>       ; invalid parameter !
2984 00010260 B817000000          <1>       mov    eax, ERR_INV_PARAMETER ; 23
2985                              <1> ;     mov    [u.r0], eax
2986                              <1> ;     mov    [u.error], eax
2987                              <1> ;     jmp    error
2988 00010265 E94BF9FFFF          <1>       jmp    sysaudio_err
2989                              <1>
2990                              <1> snd_update_0:
2991 0001026A 8035[AC650100]01    <1>       xor    byte [audio_flag], 1 ; update flag !!!
2992 00010271 3C01                <1>       cmp    al, 1
2993 00010273 7202                <1>       jb     short snd_update_2 ; dma half buffer 1
2994                              <1> snd_update_1:
2995                              <1>       ; dma half buffer 2
2996 00010275 01CF                <1>       add    edi, ecx
2997                              <1> snd_update_2:
2998                              <1>       ;rep   movsb
2999 00010277 C1E902              <1>       shr    ecx, 2
3000 0001027A F3A5                <1>       rep    movsd
3001                              <1> snd_update_3:
3002 0001027C A3[64030300]        <1>       mov    [u.r0], eax
3003                              <1>
3004 00010281 C3                  <1>       retn
3005                              <1>
3006                              <1>
3007                              <1> set_irq_callback_service:
3008                              <1>       ; 10/06/2017
3009                              <1>       ; 12/05/2017
3010                              <1>       ; 24/04/2017
3011                              <1>       ; 22/04/2017
3012                              <1>       ; caller: 'syscalbac' or 'sysaudio' or ...
```

```
3013                                 <1>       ; 13/04/2017, 14/04/2017, 17/04/2017
3014                                 <1>       ; 24/02/2017, 26/02/2017, 28/02/2017
3015                                 <1>       ; 21/02/2017 - TRDOS 386 (TRDOS v2.0)
3016                                 <1>       ;
3017                                 <1>       ; Link or unlink IRQ callback service to/from user (ring 3)
3018                                 <1>       ;
3019                                 <1>       ; INPUT ->
3020                                 <1>       ;     If AL = 0, the caller is 'syscalbac';
3021                                 <1>       ;        otherwise, the caller is 'sysaudio' or ...
3022                                 <1>       ;        (AL = user number)
3023                                 <1>       ;
3024                                 <1>       ;     BL = IRQ number (Hardware interrupt request number)
3025                                 <1>       ;          (0 t0 15 but IRQ 0,1,2,6,8,14,15 are prohibited)
3026                                 <1>       ;          IRQ numbers 3,4,5,7,9,10,11,12,13 are valid
3027                                 <1>       ;          (numbers >15 are invalid)
3028                                 <1>       ;
3029                                 <1>       ;     BH = 0 = Unlink IRQ (in BL) from user (ring 3) service
3030                                 <1>       ;          1 = Link IRQ by using Signal Response Byte method
3031                                 <1>       ;          2 = Link IRQ by using Callback service method
3032                                 <1>       ;          3 = Link IRQ by using Auto Increment S.R.B. method
3033                                 <1>       ;         >3 = invalid
3034                                 <1>       ;             (syscallback version will return to user)
3035                                 <1>       ;
3036                                 <1>       ;     CL = Signal Return/Response Byte value
3037                                 <1>       ;
3038                                 <1>       ;     If BH = 2, kernel will put a counter value
3039                                 <1>       ;               (into the S.R.B. addr)
3040                                 <1>       ;              between 0 to 255. (start value = CL+1)
3041                                 <1>       ;
3042                                 <1>       ;     NOTE: counter value, for example: even and odd numbers
3043                                 <1>       ;           may be used for -audio- DMA buffer switch
3044                                 <1>       ;           within double buffer method, etc.
3045                                 <1>       ;
3046                                 <1>       ;     EDX = Signal return (Response) byte address
3047                                 <1>       ;                           - or -
3048                                 <1>       ;           Interrupt/Callback service/routine address
3049                                 <1>       ;
3050                                 <1>       ;           (virtual address in user's memory space)
3051                                 <1>       ;
3052                                 <1>       ; OUTPUT ->
3053                                 <1>       ;     CF = 0 & EAX = 0 -> Successful setting
3054                                 <1>       ;     CF = 1 & EAX > 0 -> IRQ is prohibited or locked
3055                                 <1>       ;                  by another process
3056                                 <1>       ;             eax = ERR_PERM_DENIED -> prohibited or locked
3057                                 <1>       ;             eax = ERR_INV_PARAMETER ->
3058                                 <1>       ;                   invalid parameter/option or bad address
3059                                 <1>       ;
3060                                 <1>       ; TRDOS 386 - IRQ CALLBACK structures (parameters):
3061                                 <1>       ;
3062                                 <1>       ;       [u.irqlock] = 1 word, IRQ flags (0-15) that indicates
3063                                 <1>       ;                 which IRQs are locked by (that) user.
3064                                 <1>       ;                  Lock and unlock (by user) will change
3065                                 <1>       ;                 these flags or 'terminate process' (sysexit)
3066                                 <1>       ;                 will clear these flags and unlock those IRQs.
3067                                 <1>       ;
3068                                 <1>       ;                 Bit 0 is for IRQ 0 and Bit 15 is for IRQ 15
3069                                 <1>       ;
3070                                 <1>       ;       IRQ(x).owner      : 1 byte, user, [u.uno], 0 = free (unlocked)
3071                                 <1>       ;
3072                                 <1>       ;       IRQ(x).method : 1 byte for callback method & status
3073                                 <1>       ;                   0 = Signal Response Byte method
3074                                 <1>       ;                   1 = Callback service method
3075                                 <1>       ;                   >1 = invalid for current 'syscalback'.
3076                                 <1>       ;                 or(+) 80h = IRQ is in use by system (ring 0)
3077                                 <1>       ;                             function (audio etc.) or
3078                                 <1>       ;                           a device driver.
3079                                 <1>       ;                 (system function will ignore the lock/owner)
3080                                 <1>       ;
3081                                 <1>       ;       IRQ(x).srb: 1 byte, Signal Return/Response byte value
3082                                 <1>       ;                   (a fixed value by user or a counter value
3083                                 <1>       ;                  from 0 to 255, which is increased by every
3084                                 <1>       ;                  interrupt just before putting it into
3085                                 <1>       ;                  the Signal Response byte address
3086                                 <1>       ;                  (This is not used in callback serv method)
3087                                 <1>       ;
3088                                 <1>       ;       IRQ(x).addr      : 1 dword
3089                                 <1>       ;                  Signal Response Byte address (physical)
3090                                 <1>       ;                             -or-
3091                                 <1>       ;                  Callback service address (virtual)
3092                                 <1>       ;
3093                                 <1>       ;       IRQ(x).dev: 1 byte
3094                                 <1>       ;                  0 = Default device or kernel function
3095                                 <1>       ;                             -or-
3096                                 <1>       ;                  1-255 = Assigned device driver number
3097                                 <1>       ;
3098                                 <1>       ;       (x) = 3,4,5,7,9,10,11,12,13
3099                                 <1>       ;
3100                                 <1>
3101 00010282 80FB0F                <1>       cmp    bl, 15
3102 00010285 7729                  <1>       ja     short scbs_2
3103                                 <1>
3104 00010287 80FF03                <1>       cmp    bh, 3
3105 0001028A 7724                  <1>       ja     short scbs_2  ; invalid parameter
3106                                 <1>
3107 0001028C 0FB6FB                <1>       movzx  edi, bl ; save IRQ number
3108                                 <1>
3109                                 <1>              ;  IRQ 0,1,2,6,8,14,15 are prohibited
3110                                 <1>       ;IRQenum: ; 'trdosk9.s'
3111                                 <1>       ;      db  0,0,0,1,2,3,0,4,0,5,6,7,8,9,0,0
3112                                 <1>
3113 0001028F 0FB6B7[E20F0100]      <1>       movzx  esi, byte [edi+IRQenum] ; IRQ availability
3114                                 <1>                                      ; enumeration/index
```

```
                                <1>       ;dec   esi
00010296 664E                   <1>       dec    si
00010298 780F                   <1>       js     short scbs_1 ;  0 -> 0FFFFh
                                <1>
                                <1>       ; ESI = IRQ callback parameters index number (0 to 8)
                                <1>
0001029A 08FF                   <1>       or     bh, bh
0001029C 7419                   <1>       jz     short scbs_4 ; unlink the IRQ (in BL)
                                <1>
0001029E FECF                   <1>       dec    bh
                                <1>       ; bh = method (0 = signal response byte, 1 = callback)
                                <1>       ;            (2 = auto increment of signal response byte)
                                <1>
000102A0 80BE[36650100]00       <1>       cmp    byte [esi+IRQ.owner], 0 ; locked ?
000102A7 7637                   <1>       jna    short scbs_6 ; no... OK...
                                <1>
                                <1> scbs_1:
                                <1>       ; permission denied (prohibited IRQ)
000102A9 B80B000000             <1>       mov    eax, ERR_PERM_DENIED
000102AE F9                     <1>       stc
000102AF C3                     <1>       retn
                                <1> scbs_2:
000102B0 F9                     <1>       stc
                                <1> scbs_3:
000102B1 B817000000             <1>       mov    eax, ERR_INV_PARAMETER
000102B6 C3                     <1>       retn
                                <1>
                                <1> scbs_4: ; unlink the requested IRQ (if it belongs to current user)
                                <1>       ; 10/06/2017
                                <1>       ; 22/04/2017
                                <1>       ; 14/04/2017
                                <1>       ; If AL = 0 -> The caller is 'syscalbac'
000102B7 8AA6[36650100]         <1>       mov    ah, [esi+IRQ.owner]
000102BD 3A25[B3030300]         <1>       cmp    ah, [u.uno]
000102C3 75E4                   <1>       jne    short scbs_1
                                <1>
000102C5 FE0D[D6030300]         <1>       dec    byte [u.irqc] ; decrease IRQ count (in use)
                                <1>
                                <1>       ;sub   ah, ah
                                <1>       ;mov   [esi+IRQ.owner], ah ; 0 ; free !!!
                                <1>       ;and   byte [esi+IRQ.method], 80h
                                <1>       ;mov   [esi+IRQ.srb], ah ; 0
                                <1>       ;mov   [esi+IRQ.dev], ah ; 0
                                <1>       ;mov   dword [esi+IRQ.addr], 0
                                <1>       ;mov   dword [u.r0], 0
                                <1>
                                <1>       ;mov   byte [esi+IRQ.owner], 0
                                <1>
                                <1>       ; 22/04/2017
000102CB 29C0                   <1>       sub    eax, eax
000102CD 8886[36650100]         <1>       mov    [esi+IRQ.owner], al ; 0
                                <1>       ; 10/06/2017
000102D3 8686[48650100]         <1>       xchg   al, [esi+IRQ.method]
000102D9 2480                   <1>       and    al, 80h
000102DB 745E                   <1>       jz     short scbs_12
                                <1>       ; Audio device must be disabled -later- ! ([IRQ.medhod] = 80h)
                                <1>
                                <1> ;     cmp    byte [esi+IRQ.method], 80h ; device drv or kernel extension ?
                                <1> ;     jb     short scbs_12 ; bh = 0 reset to default IRQ handler
                                <1> ;
                                <1> ;     and    al, al
                                <1> ;     jz     short  scbs_5  ; the caller is 'syscalbac'
                                <1> ;     ; The caller is 'sysaudio' or ...
000102DD 30C0                   <1>       xor    al, al
                                <1> ;     mov    [esi+IRQ.method], al ; 0 ; reset kernel extension flag
                                <1> ;scbs_5:
                                <1> ;     sub    ah, ah
                                <1>       ;mov   [u.r0], eax ; 0
000102DF C3                     <1>       retn
                                <1>
                                <1> scbs_6:
                                <1>       ; 14/04/2017
000102E0 20C0                   <1>       and    al, al
000102E2 7405                   <1>       jz     short scbs_7  ; the caller is 'syscalbac'
                                <1>       ; AL = user number ([u.uno] or [audio.user] or ...)
                                <1>       ; The caller is 'sysaudio' or ...
                                <1>       ;
                                <1>       ; bh = method (0 = signal response byte, 1 = callback)
                                <1>       ;            (2 = auto increment of signal response byte)
                                <1>
000102E4 80CF80                 <1>       or     bh, 80h             ; Kernel extension flag !
000102E7 EB0A                   <1>       jmp    short scbs_8
                                <1> scbs_7:
000102E9 8A86[48650100]         <1>       mov    al, [esi+IRQ.method] ; >= 80h = kernel is using this IRQ
000102EF 2480                   <1>       and    al, 80h ; use only bit 7 (kernel function flag)
000102F1 08C7                   <1>       or     bh, al         ; method
                                <1>                             ; 0 = signal response byte, 1 = callback
                                <1>                             ; 2 = auto increment of s.r.b.
                                <1> scbs_8:
000102F3 A0[B3030300]           <1>       mov    al, [u.uno] ; user (process) number (1 to 16)
000102F8 8886[36650100]         <1>       mov    [esi+IRQ.owner], al ; lock the IRQ for user
000102FE 88BE[48650100]         <1>       mov    [esi+IRQ.method], bh
                                <1>
                                <1> ;     test   bh, 1
                                <1> ;     jnz    short scbs_9  ; Callback method, CX will not be used
                                <1> ;
                                <1> ;     test   bh, 2         ; use auto increment (counter) method
                                <1> ;     jz     short scbs_10 ; (count can be used for buffer switch)
                                <1> ;scbs_9:
                                <1> ;     xor    ecx, ecx ; 0
                                <1> scbs_10:
                                <1>       ;mov   [esi+IRQ.method], bh
```

```
3217 00010304 888E[51650100]      <1>       mov    [esi+IRQ.srb], cl
3218 0001030A C686[3F650100]00    <1>       mov    byte [esi+IRQ.dev], 0 ; device number is always 0
3219                              <1>                                    ; for this system call
3220                              <1>       ;test  bh, 1
3221 00010311 80E701              <1>       and    bh, 1 ; 17/04/2017
3222 00010314 7513                <1>       jnz    short scbs_11 ; callback method, use virtual address
3223                              <1>
3224 00010316 53                  <1>       push   ebx ; IRQ number (in BL)
3225 00010317 89D3                <1>       mov    ebx, edx
3226                              <1>       ; ebx = virtual address
3227                              <1>       ; [u.pgdir] = page directory's physical address
3228 00010319 FE05[D6640100]      <1>       inc     byte [no_page_swap] ; 1
3229                              <1>                    ; Do not add this page to swap queue
3230                              <1>                    ; and remove it from swap queue if it is
3231                              <1>                    ; on the queue.
3232 0001031F E86A4FFFFF          <1>       call   get_physical_addr
3233 00010324 5B                  <1>       pop    ebx
3234 00010325 728A                <1>       jc     scbs_3 ; invalid address !
3235                              <1>       ; eax = physical address of the virtual address in user's space
3236 00010327 89C2                <1>       mov    edx, eax
3237                              <1> scbs_11:
3238 00010329 66C1E602            <1>       shl    si, 2        ; byte (index) to dword (offset)
3239 0001032D 8996[5A650100]      <1>       mov    [esi+IRQ.addr], edx
3240                              <1>
3241 00010333 FE05[D6030300]      <1>       inc    byte [u.irqc]; increase IRQ (in use) count
3242                              <1>
3243 00010339 FEC7                <1>       inc     bh ; 17/04/2017
3244                              <1>       ; bh > 0 -> set to requested IRQ handler (IRQ_u_list)
3245                              <1> scbs_12:
3246 0001033B 88D8                <1>       mov    al, bl ; IRQ number
3247 0001033D 88FC                <1>       mov    ah, bh ; 0 = reset, >0 = set
3248 0001033F E8DAF3FFFF          <1>       call   set_hardware_int_vector
3249                              <1>
3250 00010344 31C0                <1>       xor    eax, eax
3251                              <1>       ;mov   [u.r0], eax ; 0
3252                              <1>
3253 00010346 C3                  <1>       retn   ; return with success (cf=0, eax=0)
3254                              <1>
3255                              <1>
3256                              <1> sysdma: ; DMA FUNCTIONS
3257                              <1>       ; 02/09/2017
3258                              <1>       ; 28/08/2017
3259                              <1>       ; 20/08/2017 - TRDOS 386 (TRDOS v2.0)
3260                              <1>       ;
3261                              <1>       ; Inputs:
3262                              <1>       ;     BH = 0 -> Allocate DMA buffer
3263                              <1>       ;        BL = 0 -> Use the system's default DMA
3264                              <1>       ;                  (SB16) Buffer
3265                              <1>       ;             Buffer Size (max.) = 65536 bytes
3266                              <1>       ;        BL > 0 -> Allocate (a new) DMA buffer
3267                              <1>       ;        ECX = DMA Buffer Size in bytes (<=128KB)
3268                              <1>       ;        EDX = Virtual Address of DMA buffer
3269                              <1>       ;
3270                              <1>       ;     BH = 1 -> Initialize (Start) DMA service
3271                              <1>       ;          BL, bit 0 to 3 = Channel Number (0 to 7)
3272                              <1>       ;          BL, bit 7 = Auto Initialized Mode
3273                              <1>       ;                  (If bit 7 is set)
3274                              <1>       ;             bit 6 = Record (read) mode (0= playback)
3275                              <1>       ;          ECX = byte count (0 = use dma buffer size)
3276                              <1>       ;          EDX = physical buffer address
3277                              <1>       ;              (0 = use dma buffer -start- address)
3278                              <1>       ;
3279                              <1>       ;     BH = 2 -> Get Current DMA Buffer Offset
3280                              <1>       ;          BL = DMA channel number
3281                              <1>       ;
3282                              <1>       ;     BH = 3 -> Get Current DMA count down value
3283                              <1>       ;          BL = DMA channel number (0 tO 7)
3284                              <1>       ;
3285                              <1>       ;     BH = 4 -> Get Current DMA channel (in progress)
3286                              <1>       ;
3287                              <1>       ;     BH = 5 -> Get System's Default DMA Buffer Address
3288                              <1>       ;
3289                              <1>       ;     BH = 6 -> Get Current DMA Buffer Address
3290                              <1>       ;
3291                              <1>       ;     BH = 7 -> Stop DMA service
3292                              <1>       ;
3293                              <1>       ;
3294                              <1>       ; Outputs:
3295                              <1>       ;
3296                              <1>       ;     For BH = 0 ; Allocate DMA buffer
3297                              <1>       ;         EAX = Physical address of DMA buffer
3298                              <1>       ;         ECX = Allocated buffer size in bytes
3299                              <1>       ;             - page count * 4096 -
3300                              <1>       ;             (may be bigger than requested)
3301                              <1>       ;         If BL input > 0,
3302                              <1>       ;            'sysalloc:' system call will be used with
3303                              <1>       ;            EBX (for 'sysalloc') = EDX (for 'sysdma')
3304                              <1>       ;            ECX is same, byte count (buffer size)
3305                              <1>       ;            EDX = 1024*1024*16 ; 16 MB upper limit
3306                              <1>       ;         If BL input = 0,
3307                              <1>       ;            Default DMA buffer (SB16 buffer) will be
3308                              <1>       ;            checked and if it is free, it's address
3309                              <1>       ;            will be returned in EAX and it's size
3310                              <1>       ;            will be returned in ECX (as 65536)
3311                              <1>       ;
3312                              <1>       ;         If CF = 1, error code is in EAX
3313                              <1>       ;            EAX = -1 ; DMA buffer allocation error!
3314                              <1>       ;            EAX = 11 ; 'Permission Denied' error !
3315                              <1>       ;
3316                              <1>       ;            Note: 'sysalloc' error return method
3317                              <1>       ;                  will be applied if BL input > 0 !
3318                              <1>       ;
```

```
3319                                 <1>       ;      For BH = 1 ; Initialize (Start) DMA
3320                                 <1>       ;          EAX = 0 (Successful)
3321                                 <1>       ;          If CF = 1, error code is in EAX
3322                                 <1>       ;
3323                                 <1>       ;      For BH = 2 ; Get Current DMA Buffer Offset
3324                                 <1>       ;          EAX = DMA Buffer Offset (in bytes)
3325                                 <1>       ;          ;
3326                                 <1>       ;           AX = DMA buffer offset
3327                                 <1>       ;          EAX bits 16 to 23 = Page register value
3328                                 <1>       ;
3329                                 <1>       ;      For BH = 3 ; Get Current DMA count down value
3330                                 <1>       ;          EAX = Count down value (remain bytes)
3331                                 <1>       ;
3332                                 <1>       ;      For BH = 4 ; Get Current DMA channel (in progress)
3333                                 <1>       ;          EAX = DMA channel number (0 to 7)
3334                                 <1>       ;            AH = 0 if the owner is the caller process
3335                                 <1>       ;            AH > 0 if the dma channel is in use by
3336                                 <1>       ;                   another user/process
3337                                 <1>       ;          EAX = -1 (0FFFFFFFFh)
3338                                 <1>       ;                if DMA service is not in use
3339                                 <1>       ;                (stopped or not initialized/started)
3340                                 <1>       ;
3341                                 <1>       ;      For BH = 5 ; Get System's Default DMA Buff Addr
3342                                 <1>       ;          EAX = Default DMA Buffer Address (Physical)
3343                                 <1>       ;            = offset 'sb16_dma_buffer:'
3344                                 <1>       ;          ECX = Buffer size
3345                                 <1>       ;            = 65536
3346                                 <1>       ;
3347                                 <1>       ;      For BH = 6 ; Get Current DMA Buffer Address
3348                                 <1>       ;          EAX = Current DMA buffer address (Physical)
3349                                 <1>       ;          ECX = Current DMA buffer size (setting value)
3350                                 <1>       ;           Note: These values are for current dma channel
3351                                 <1>       ;                settings for the user/process
3352                                 <1>       ;           ** For now (for current TRDOS 386 version)
3353                                 <1>       ;             only one user/process can use only one
3354                                 <1>       ;             dma channel & one dma buffer at same time
3355                                 <1>       ;             (no multi tasking on DMA service) !!! **
3356                                 <1>       ;           (Once, current DMA user must stop it's own DMA
3357                                 <1>       ;            DMA service, than another user/program
3358                                 <1>       ;            can use DMA service with same dma channel
3359                                 <1>       ;            or with another DMA channel.)
3360                                 <1>       ;
3361                                 <1>       ;      For BH = 7 ; Stop DMA service (for current user
3362                                 <1>       ;          and current DMA channel)
3363                                 <1>       ;          EAX = 0 ; successful
3364                                 <1>       ;          CF = 1 & EAX > 0 (= -1) -> Error
3365                                 <1>
3366 00010347 80FF07               <1>       cmp    bh, 7
3367 0001034A 7612                 <1>       jna    short sysdma_0
3368                                 <1>
3369                                 <1> sysdma_err:
3370 0001034C 31C0                 <1>       xor    eax, eax
3371 0001034E 48                   <1>       dec    eax ; -1
3372                                 <1> sysdma_perm_err:
3373 0001034F A3[64030300]         <1>       mov    [u.r0], eax
3374 00010354 A3[C8030300]         <1>       mov    [u.error], eax ; DMA service error !
3375 00010359 E935C1FFFF           <1>       jmp    error
3376                                 <1>
3377                                 <1> sysdma_0:
3378 0001035E 08FF                 <1>       or     bh, bh
3379 00010360 0F85BA000000         <1>       jnz    sysdma_1
3380                                 <1>
3381 00010366 20DB                 <1>       and    bl, bl
3382 00010368 7416                 <1>       jz     short sysdma_01
3383                                 <1>
3384                                 <1>       ; redirect system call to 'sysalloc'
3385 0001036A 89D3                 <1>       mov    ebx, edx ; virtual address of DMA buffer
3386                                 <1>       ;ecx = Buffer size in bytes
3387                                 <1>       ; DMA buffer address <= 16MB upper limit
3388 0001036C BA00000001           <1>       mov    edx, 1024*1024*16 ; 16MB limit for DMA buff
3389                                 <1>
3390 00010371 C705[C8690100]FFFF-  <1>       mov    dword [dma_addr], 0FFFFFFFFh ; -1
3390 00010379 FFFF                 <1>
3391                                 <1>
3392 0001037B E9DBECFFFF           <1>       jmp    sysalloc
3393                                 <1>
3394                                 <1> sysdma_01:
3395 00010380 B8[00000200]         <1>       mov    eax, sb16_dma_buffer
3396                                 <1>
3397 00010385 803D[85650100]01     <1>       cmp    byte [audio_device], 1
3398 0001038C 722A                 <1>       jb     short sysdma_03
3399                                 <1>
3400 0001038E 3B05[A4650100]       <1>       cmp    eax, [audio_dma_buff]
3401 00010394 7507                 <1>       jne    short sysdma_02
3402                                 <1>
3403                                 <1> sysdma_0_err:
3404 00010396 B80B000000           <1>       mov    eax, ERR_PERM_DENIED
3405 0001039B EBB2                 <1>       jmp    short sysdma_perm_err
3406                                 <1>
3407                                 <1> sysdma_02:
3408                                 <1>       ; Only one user is permitted for audio/dma functions
3409                                 <1>
3410 0001039D 833D[A4650100]00     <1>       cmp    dword [audio_dma_buff], 0
3411 000103A4 7612                 <1>       jna    short sysdma_03
3412                                 <1>
3413 000103A6 8A1D[AD650100]       <1>       mov    bl, [audio_user]
3414 000103AC 08DB                 <1>       or     bl, bl
3415 000103AE 7408                 <1>       jz     short sysdma_03
3416                                 <1>
3417 000103B0 3A1D[B3030300]       <1>       cmp    bl, [u.uno]
3418 000103B6 75DE                 <1>       jne    short sysdma_0_err
```

```
                                <1>
                                <1> sysdma_03:
000103B8 8A1D[C5690100]         <1>       mov   bl, [dma_user]
000103BE 20DB                   <1>       and   bl, bl
000103C0 750E                   <1>       jnz   short sysdma_04
                                <1>
000103C2 8A1D[B3030300]         <1>       mov   bl, [u.uno]
000103C8 881D[C5690100]         <1>       mov   [dma_user], bl
                                <1>
000103CE EB15                   <1>       jmp   short sysdma_05
                                <1>
                                <1> sysdma_04:
000103D0 8B35[C8690100]         <1>       mov   esi, [dma_addr]
000103D6 21F6                   <1>       and   esi, esi
000103D8 740B                   <1>       jz    short sysdma_05
                                <1>
000103DA 46                     <1>       inc   esi ; -1 -> 0
000103DB 7408                   <1>       jz    short sysdma_05
                                <1>
000103DD 3A1D[B3030300]         <1>       cmp   bl, [u.uno]
000103E3 75B1                   <1>       jne   short sysdma_0_err
                                <1>
                                <1> sysdma_05:
                                <1>       ; edx = virtual address (user's buffer address)
                                <1>       ;
000103E5 81F900000100           <1>       cmp   ecx, 65536   ; byte count (buffer size)
000103EB 0F875BFFFFFF           <1>       ja    sysdma_err
                                <1>       ;
000103F1 81C1FF0F0000           <1>       add   ecx, PAGE_SIZE-1 ; 4095
000103F7 6681E100F0             <1>       and   cx, ~PAGE_OFF ; not 4095
                                <1>       ;cmp   ecx, 65536
                                <1>       ;ja    sysdma_err ;
000103FC 51                     <1>       push  ecx    ; buffer size (allocated pages * 4096)
000103FD 50                     <1>       push  eax    ; offset sb16_dma_buffer
000103FE 89D3                   <1>       mov   ebx, edx
00010400 C1E90C                 <1>       shr   ecx, 12 ; byte count to page count
                                <1>       ; eax = physical address of (audio) dma buffer
                                <1>       ; ebx = virtual address of (audio) dma buffer (user's pgdir)
                                <1>       ; ecx = page count (>0)
00010403 E89852FFFF             <1>       call  direct_memory_access
00010408 58                     <1>       pop   eax
00010409 59                     <1>       pop   ecx
0001040A 0F823CFFFFFF           <1>       jc    sysdma_err
                                <1>
00010410 A3[C8690100]           <1>       mov   [dma_addr], eax
00010415 890D[CC690100]         <1>       mov   [dma_size], ecx ; dma buffer size (in bytes)
                                <1>
                                <1>       ;mov   [u.r0], eax ; DMA Buffer Address (Physical)
                                <1>
                                <1>       ;mov   ebp, [u.usp]  ; ebp points to user's registers
                                <1>       ;mov   [ebp+24], ecx ; return to user with ecx value
                                <1>
                                <1>       ;jmp   sysret
                                <1>
                                <1>       ; 28/08/2017
0001041B E9C4000000             <1>       jmp   sysdma_51
                                <1>
                                <1> sysdma_1:
00010420 80FF01                 <1>       cmp   bh, 1
00010423 0F87A6000000           <1>       ja    sysdma_5
                                <1>
00010429 F6C340                 <1>       test  bl, 40h      ; record (read) mode -BL, bit 6-
0001042C 0F851AFFFFFF           <1>       jnz   sysdma_err ; not ready yet!
                                <1>
00010432 A1[C8690100]           <1>       mov   eax, [dma_addr] ; physical address of dma buffer
00010437 21C0                   <1>       and   eax, eax
00010439 0F840DFFFFFF           <1>       jz    sysdma_err
                                <1>
0001043F 09D2                   <1>       or    edx, edx
00010441 7504                   <1>       jnz   short sysdma_11
                                <1>
00010443 89C2                   <1>       mov   edx, eax
00010445 EB08                   <1>       jmp   short sysdma_12
                                <1> sysdma_11:
00010447 39C2                   <1>       cmp   edx, eax
00010449 0F82FDFEFFFF           <1>       jb    sysdma_err
                                <1> sysdma_12:
0001044F 21C9                   <1>       and   ecx, ecx
00010451 7508                   <1>       jnz   short sysdma_13
                                <1>
00010453 8B0D[CC690100]         <1>       mov   ecx, [dma_size]
00010459 EB0C                   <1>       jmp   short sysdma_14
                                <1> sysdma_13:
0001045B 3B0D[CC690100]         <1>       cmp   ecx, [dma_size]
00010461 0F87E5FEFFFF           <1>       ja    sysdma_err
                                <1> sysdma_14:
00010467 89C6                   <1>       mov   esi, eax
00010469 0335[CC690100]         <1>       add   esi, [dma_size]
                                <1>
0001046F 89D0                   <1>       mov   eax, edx
00010471 01C8                   <1>       add   eax, ecx
00010473 0F82D3FEFFFF           <1>       jc    sysdma_err ; 02/09/2017
                                <1>
00010479 39F0                   <1>       cmp   eax, esi
0001047B 0F87CBFEFFFF           <1>       ja    sysdma_err
                                <1>
00010481 8B3D[A4650100]         <1>       mov   edi, [audio_dma_buff]
00010487 8B35[C8690100]         <1>       mov   esi, [dma_addr]
                                <1>
0001048D 09FF                   <1>       or    edi, edi
0001048F 7424                   <1>       jz    short sysdma_16
                                <1>
```

```
3521 00010491 803D[85650100]01    <1>       cmp    byte [audio_device], 1
3522 00010498 7208                <1>       jb     short sysdma_15
3523                              <1>
3524                              <1>       ; Sound Blaster 16
3525 0001049A 39FE                <1>       cmp    esi, edi
3526 0001049C 0F84F4FEFFFF        <1>       je     sysdma_0_err ; permmission denied !
3527                              <1>
3528                              <1> sysdma_15:
3529 000104A2 C605[C7690100]48    <1>       mov    byte [dma_mode], 48h ; single mode playback
3530                              <1>
3531 000104A9 F6C380              <1>       test   bl, 80h ; DMA mode - BL, bit 7, auto init -
3532 000104AC 7407                <1>       jz     short sysdma_16
3533                              <1>       ; Auto initialized playback (write) mode
3534 000104AE 8005[C7690100]10    <1>       add    byte [dma_mode], 10h ; = 58h
3535                              <1> sysdma_16:
3536 000104B5 80E307              <1>       and    bl, 07h
3537 000104B8 881D[C6690100]      <1>       mov    [dma_channel], bl
3538 000104BE 8915[D0690100]      <1>       mov    [dma_start], edx
3539 000104C4 890D[D4690100]      <1>       mov    [dma_count], ecx
3540                              <1>
3541                              <1>       ; 28/08/2017
3542                              <1>       ;call  dma_init
3543                              <1>       ;jmp   sysret
3544 000104CA E94B010000          <1>       jmp    dma_init
3545                              <1>
3546                              <1> sysdma_5:
3547 000104CF 80FF05              <1>       cmp    bh, 5
3548 000104D2 7223                <1>       jb     short sysdma_3
3549 000104D4 0F87CE000000        <1>       ja     sysdma_6
3550                              <1>
3551                              <1>       ; Get the system's default dma buffer addr and size
3552 000104DA B8[00000200]        <1>       mov    eax, sb16_dma_buffer
3553 000104DF B900000100          <1>       mov    ecx, 65536 ; Buffer size in bytes
3554                              <1>
3555                              <1> sysdma_51:
3556                              <1>       ; 0 = there is not a dma buffer (in use or available)
3557 000104E4 A3[64030300]        <1>       mov    [u.r0], eax
3558                              <1>
3559 000104E9 8B2D[60030300]      <1>       mov    ebp, [u.usp]  ; ebp points to user's registers
3560 000104EF 894D18              <1>       mov    [ebp+24], ecx ; return to user with ecx value
3561                              <1>
3562 000104F2 E9BCBFFFFF          <1>       jmp    sysret
3563                              <1>
3564                              <1> sysdma_3:
3565 000104F7 80FF03              <1>       cmp    bh, 3
3566 000104FA 7231                <1>       jb     short sysdma_2
3567 000104FC 776B                <1>       ja     short sysdma_4
3568                              <1>
3569                              <1>       ; Get current dma count down value (remain bytes)
3570                              <1>       ; 28/08/2017
3571 000104FE 0FB635[C6690100]    <1>       movzx  esi, byte [dma_channel]
3572 00010505 0FB696[1A100100]    <1>       movzx  edx, byte [dma_flip+esi]
3573 0001050C EE                  <1>       out    dx, al              ; flip-flop clear
3574 0001050D 8A96[FA0F0100]      <1>       mov    dl, [dma_cnt+esi] ; dma count register addr
3575 00010513 EC                  <1>       in     al, dx
3576 00010514 0FB6D8              <1>       movzx  ebx, al
3577 00010517 EC                  <1>       in     al, dx
3578 00010518 88C7                <1>       mov     bh, al
3579                              <1>
3580 0001051A 6683FE04            <1>       cmp    si, 4 ; channel number ?
3581 0001051E 7202                <1>       jb     short sysdma_31 ; 8 bit dma channel
3582                              <1>
3583 00010520 D1E3                <1>       shl    ebx, 1 ; word count to byte count
3584                              <1>
3585                              <1> sysdma_31:
3586 00010522 891D[64030300]      <1>       mov    [u.r0], ebx
3587                              <1>
3588 00010528 E986BFFFFF          <1>       jmp    sysret
3589                              <1>
3590                              <1> sysdma_2:
3591                              <1>       ; Get current dma buffer offset (& page)
3592                              <1>       ; 28/08/2017
3593 0001052D 0FB635[C6690100]    <1>       movzx  esi, byte [dma_channel]
3594 00010534 0FB696[1A100100]    <1>       movzx  edx, byte [dma_flip+esi]
3595 0001053B EE                  <1>       out    dx, al              ; flip-flop clear
3596 0001053C 8A96[F20F0100]      <1>       mov    dl, [dma_adr+esi]
3597 00010542 EC                  <1>       in     al, dx              ; get dma position
3598 00010543 0FB6D8              <1>       movzx  ebx, al
3599 00010546 EC                  <1>       in     al, dx
3600 00010547 88C7                <1>       mov    bh, al
3601                              <1>
3602 00010549 6683FE04            <1>       cmp    si, 4 ; channel number ?
3603 0001054D 7202                <1>       jb     short sysdma_21 ; 8 bit dma channel
3604                              <1>
3605 0001054F D1E3                <1>       shl    ebx, 1 ; word offset to byte offset
3606                              <1>
3607                              <1> sysdma_21:
3608 00010551 891D[64030300]      <1>       mov    [u.r0], ebx
3609                              <1>
3610 00010557 8A96[02100100]      <1>       mov    dl, [dma_page+esi]
3611 0001055D EC                  <1>       in     al, dx              ; get dma page
3612                              <1>
3613                              <1>       ;add   [u.ro+2], al
3614 0001055E 0805[66030300]      <1>       or     [u.r0+2], al
3615                              <1>
3616 00010564 E94ABFFFFF          <1>       jmp    sysret
3617                              <1>
3618                              <1> sysdma_4:
3619                              <1>       ; Get current DMA channel number
3620                              <1>       ; 28/08/2017
3621 00010569 8A25[C5690100]      <1>       mov    ah, [dma_user]
3622 0001056F 20E4                <1>       and    ah, ah
```

```
3623 00010571 750F                <1>       jnz    short sysdma_42
3624                              <1>
3625                              <1> sysdma_41:
3626                              <1>       ; Not a valid dma channel (in use)
3627 00010573 C705[64030300]FFFF- <1>       mov    dword [u.r0], -1 ; 0FFFFFFFFh
3627 0001057B FFFF                <1>
3628 0001057D E931BFFFFF          <1>       jmp    sysret
3629                              <1>
3630                              <1> sysdma_42:
3631 00010582 8B35[C8690100]      <1>       mov    esi, [dma_addr]
3632 00010588 21F6                <1>       and    esi, esi
3633 0001058A 74E7                <1>       jz     short sysdma_41
3634                              <1>
3635 0001058C 46                  <1>       inc    esi ; -1 -> 0
3636 0001058D 74E4                <1>       jz     short sysdma_41
3637                              <1>
3638 0001058F A0[C6690100]        <1>       mov    al, [dma_channel]
3639                              <1>
3640 00010594 3A25[B3030300]      <1>       cmp    ah, [u.uno]
3641 0001059A 7502                <1>       jne    short sysdma_43
3642                              <1>
3643 0001059C 30E4                <1>       xor    ah, ah ; DMA channel in use by current user
3644                              <1>
3645                              <1> sysdma_43:
3646 0001059E A3[64030300]        <1>       mov    [u.r0], eax ; AL = dma channel number
3647                              <1>                          ; AH > 0 if the the channel
3648                              <1>                          ; in use by another user/process
3649 000105A3 E90BBFFFFF          <1>       jmp    sysret
3650                              <1>
3651                              <1> sysdma_6:
3652 000105A8 80FF06              <1>       cmp    bh, 6
3653 000105AB 7710                <1>       ja     short sysdma_7
3654                              <1>
3655                              <1>       ; 28/08/2017
3656                              <1>       ; Get current DMA buffer addr and size
3657 000105AD A1[C8690100]        <1>       mov    eax, [dma_addr] ; dma buffer address
3658 000105B2 8B0D[CC690100]      <1>       mov    ecx, [dma_size] ; dma buffer size (in bytes)
3659                              <1>
3660 000105B8 E927FFFFFF          <1>       jmp    sysdma_51
3661                              <1>
3662                              <1> sysdma_7:
3663                              <1>       ; DMA service STOP
3664 000105BD A0[B3030300]        <1>       mov    al, [u.uno]
3665 000105C2 3A05[C5690100]      <1>       cmp    al, [dma_user]
3666 000105C8 751D                <1>       jne    short sysdma_72
3667                              <1>
3668 000105CA 28C0                <1>       sub    al, al ; 0
3669                              <1>
3670 000105CC A2[C5690100]        <1>       mov    [dma_user], al ; clear user
3671                              <1>
3672 000105D1 8605[C7690100]      <1>       xchg   al, [dma_mode]
3673 000105D7 20C0                <1>       and    al, al
3674                              <1>       ;jz    short sysdma_err
3675 000105D9 7527                <1>       jnz    short sysdma_73
3676                              <1>
3677                              <1> sysdma_71:
3678 000105DB 31C0                <1>       xor    eax, eax
3679 000105DD A3[64030300]        <1>       mov    [u.r0], eax; 0
3680 000105E2 E9CCBEFFFF          <1>       jmp    sysret
3681                              <1>
3682                              <1> sysdma_72:
3683                              <1>       ; 28/08/2017
3684 000105E7 803D[C5690100]00    <1>       cmp    byte [dma_user], 0
3685 000105EE 76EB                <1>       jna    short sysdma_71 ; Nothing to do !
3686                              <1>
3687 000105F0 833D[C8690100]00    <1>       cmp    dword [dma_addr], 0
3688 000105F7 0F8799FDFFFF        <1>       ja     sysdma_0_err
3689                              <1>
3690 000105FD A2[C5690100]        <1>       mov    [dma_user], al ; reset to current user
3691                              <1>
3692                              <1> sysdma_73:
3693                              <1>       ; 28/08/2017
3694 00010602 0FB635[C6690100]    <1>       movzx  esi, byte [dma_channel]
3695 00010609 0FB696[0A100100]    <1>       movzx  edx, byte [dma_mask+esi]
3696 00010610 A0[C6690100]        <1>       mov    al, [dma_channel]
3697 00010615 0C04                <1>       or     al, 4
3698 00010617 EE                  <1>       out    dx, al
3699                              <1>
3700 00010618 EBC1                <1>       jmp    short sysdma_71
3701                              <1>
3702                              <1> dma_init:
3703                              <1>       ; 28/08/2017
3704                              <1>       ; 20/08/2017
3705                              <1>       ; DMA initialization
3706                              <1>       ; 14/08/2017
3707                              <1>       ; 03/08/2017, 06/08/2017, 08/08/2017
3708                              <1>       ; 02/07/2017, 13/07/2017, 16/07/2017, 30/07/2017
3709                              <1>       ; (Derived from 'DMA_INIT' procedure in SB16MOD.ASM)
3710                              <1>       ; Modified for TRDOS 386 DMA buffer allocation & initialization !
3711                              <1>
3712 0001061A 8B1D[D0690100]      <1>       mov    ebx, [dma_start]
3713 00010620 8B0D[D4690100]      <1>       mov    ecx, [dma_count]
3714                              <1>
3715 00010626 0FB635[C6690100]    <1>       movzx  esi, byte [dma_channel]
3716                              <1>
3717 0001062D 6683FE04            <1>       cmp    si, 4
3718 00010631 7205                <1>       jb     short gdmi1
3719                              <1>       ; 08/08/2017
3720 00010633 66D1E9              <1>       shr    cx, 1 ; word count
3721 00010636 D1EB                <1>       shr    ebx, 1 ; convert byte offset to word offset
3722                              <1> gdmi1:
3723                              <1>       ;mov   [dma_poff], bx ; 08/08/2017
```

```
3724 00010638 6649                <1>       dec    cx                  ; dma size = block size - 1
3725                              <1>
3726 0001063A 0FB696[0A100100]    <1>       movzx  edx, byte [dma_mask+esi] ; 30/07/2017
3727 00010641 A0[C6690100]        <1>       mov    al, [dma_channel]
3728 00010646 0C04                <1>       or     al, 4
3729 00010648 EE                  <1>       out    dx, al              ; dma channel mask
3730                              <1>
3731 00010649 30C0                <1>       xor    al, al ; 0 ; any value ! 08/08/2017
3732 0001064B 8A96[1A100100]      <1>       mov    dl, [dma_flip+esi]
3733 00010651 EE                  <1>       out    dx, al              ; flip-flop clear
3734                              <1>
3735 00010652 8A96[12100100]      <1>       mov    dl, [dma_mod+esi]
3736 00010658 A0[C6690100]        <1>       mov    al, [dma_channel]  ; 13/07/2017
3737 0001065D 2403                <1>       and    al, 3
3738                              <1>       ; 08/08/2017
3739 0001065F 0A05[C7690100]      <1>       or     al, [dma_mode] ; 58h    ; dma mode for SB16
3740 00010665 EE                  <1>       out    dx, al
3741                              <1>
3742 00010666 8A96[F20F0100]      <1>       mov    dl, [dma_adr+esi]
3743 0001066C 88D8                <1>       mov    al, bl
3744 0001066E EE                  <1>       out    dx, al              ; offset low
3745                              <1>
3746 0001066F 88F8                <1>       mov    al, bh
3747 00010671 EE                  <1>       out    dx, al              ; offset high
3748                              <1>
3749 00010672 8A96[FA0F0100]      <1>       mov    dl, [dma_cnt+esi]
3750 00010678 88C8                <1>       mov    al, cl
3751 0001067A EE                  <1>       out    dx, al              ; size low
3752                              <1>
3753 0001067B 88E8                <1>       mov    al, ch
3754 0001067D EE                  <1>       out    dx, al              ; size high
3755                              <1>
3756 0001067E 8A96[02100100]      <1>       mov    dl, [dma_page+esi]
3757                              <1>       ; 14/08/2017
3758 00010684 6683FE04            <1>       cmp    si, 4
3759 00010688 7305                <1>       jnb    short gdmi2
3760 0001068A C1EB10              <1>       shr    ebx, 16
3761 0001068D EB06                <1>       jmp    short gdmi3
3762                              <1> gdmi2:
3763                              <1>       ; 09/08/2017
3764 0001068F C1EB0F              <1>       shr    ebx, 15       ; complete 16 bit shift
3765 00010692 80E3FE              <1>       and    bl, 0FEh ; clear bit 0 (not necessary)
3766                              <1> gdmi3:
3767 00010695 88D8                <1>       mov    al, bl
3768 00010697 EE                  <1>       out    dx, al              ; page
3769                              <1>
3770 00010698 8A96[0A100100]      <1>       mov    dl, [dma_mask+esi]
3771 0001069E A0[C6690100]        <1>       mov    al, [dma_channel]  ; 13/07/2017
3772 000106A3 2403                <1>       and    al, 3
3773 000106A5 EE                  <1>       out    dx, al              ; dma channel unmask
3774                              <1>
3775                              <1>       ;retn
3776                              <1>       ; 28/08/2017
3777 000106A6 E908BEFFFF          <1>       jmp    sysret
3778                              <1>
3779                              <1> otty:
3780                              <1> sret:
3781                              <1> ocvt:
3782                              <1> ctty:
3783                              <1> cret:
3784                              <1> ccvt:
3785                              <1> rtty:
3786                              <1> wtty:
3787                              <1> rmem:
3788                              <1> wmem:
3789                              <1> rfd:
3790                              <1> rhd:
3791                              <1> wfd:
3792                              <1> whd:
3793                              <1> rlpt:
3794                              <1> wlpt:
3795                              <1> rcvt:
3796                              <1> xmtt:
3797 000106AB C3                  <1>       retn
2312                                  %include 'trdosk9.s' ; 04/01/2016
   1                              <1> ; ****************************************************************************
   2                              <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - INITIALIZED DATA : trdosk9.s
   3                              <1> ; ----------------------------------------------------------------------------
   4                              <1> ; Last Update: 22/10/2017
   5                              <1> ; ----------------------------------------------------------------------------
   6                              <1> ; Beginning: 04/01/2016
   7                              <1> ; ----------------------------------------------------------------------------
   8                              <1> ; ----------------------------------------------------------------------------
   9                              <1> ; Assembler: NASM version 2.11 (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 000106AC 01                  <1>             db 1 ; A: = 0 & B: = 1
  22                              <1>
  23                              <1> Restore_CDIR:
  24 000106AD FF                  <1>             db 0FFh ; Initial value -> any number except 0
  25                              <1>
  26                              <1> msg_CRLF_temp:
  27 000106AE 070D0A00            <1>             db 07h, 0Dh, 0Ah, 0
```

```
                                <1>
                                <1> Magic_Bytes:
000106B2 04                     <1>             db 4
000106B3 01                     <1>             db 1
                                <1> mainprog_Version:
000106B4 07                     <1>             db 7
000106B5 5B5452444F535D204D-    <1>             db "[TRDOS] Main Program v2.0.221017"
000106BE 61696E2050726F6772-    <1>
000106C7 616D2076322E302E32-    <1>
000106D0 3231303137             <1>
000106D5 0D0A                   <1>             db 0Dh, 0Ah
000106D7 286329204572646F67-    <1>             db "(c) Erdogan Tan 2005-2017"
000106E0 616E2054616E203230-    <1>
000106E9 30352D32303137         <1>
000106F0 0D0A00                 <1>             db 0Dh, 0Ah, 0
                                <1>
                                <1> MainProgCfgFile: ; 14/04/2016
000106F3 4D41494E50524F472E-    <1>             db "MAINPROG.CFG", 0
000106FC 43464700               <1>
                                <1>
                                <1> TRDOSPromptLabel:
00010700 5452444F53             <1>             db "TRDOS"
00010705 00                     <1>             db 0
00010706 00<rept>               <1>                times 5 db 0
0001070B 00                     <1>             db 0
                                <1>
                                <1> ; INTERNAL COMMANDS
                                <1> Command_List:
0001070C 44495200               <1> Cmd_Dir:    db "DIR", 0
00010710 434400                 <1> Cmd_Cd:        db "CD", 0
00010713 433A00                 <1> Cmd_Drive:  db "C:", 0
00010716 56455200               <1> Cmd_Ver:    db "VER", 0
0001071A 4558495400             <1> Cmd_Exit:   db "EXIT", 0
0001071F 50524F4D505400         <1> Cmd_Prompt: db "PROMPT", 0
00010726 564F4C554D4500         <1> Cmd_Volume: db "VOLUME", 0
0001072D 4C4F4E474E414D4500     <1> Cmd_LongName:  db "LONGNAME", 0
00010736 4441544500             <1> Cmd_Date:   db "DATE", 0
0001073B 54494D4500             <1> Cmd_Time:   db "TIME", 0
00010740 52554E00               <1> Cmd_Run:    db "RUN", 0
00010744 53455400               <1> Cmd_Set:    db "SET", 0
00010748 434C5300               <1> Cmd_Cls:    db "CLS", 0
0001074C 53484F5700             <1> Cmd_Show:   db "SHOW", 0
00010751 44454C00               <1> Cmd_Del:    db "DEL", 0
00010755 41545452494200         <1> Cmd_Attrib: db "ATTRIB", 0
0001075C 52454E414D4500         <1> Cmd_Rename: db "RENAME", 0
00010763 524D44495200           <1> Cmd_Rmdir:  db "RMDIR", 0
00010769 4D4B44495200           <1> Cmd_Mkdir:  db "MKDIR", 0
0001076F 434F505900             <1> Cmd_Copy:   db "COPY", 0
00010774 4D4F564500             <1> Cmd_Move:   db "MOVE", 0
00010779 5041544800             <1> Cmd_Path:   db "PATH", 0
0001077E 4D454D00               <1> Cmd_Mem:    db "MEM", 0
00010782 00                     <1>             db 0
00010783 46494E4400             <1> Cmd_Find:   db "FIND", 0
00010788 4543484F00             <1> Cmd_Echo:   db "ECHO", 0
0001078D 2A00                   <1> Cmd_Remark: db "*", 0
0001078F 3F00                   <1> Cmd_Help:   db "?", 0
00010791 44455649434500         <1> Cmd_Device: db "DEVICE", 0
00010798 4445564C49535400       <1> Cmd_DevList:db "DEVLIST", 0
000107A0 434844495200           <1> Cmd_Chdir:  db "CHDIR", 0
000107A6 4245455000             <1> Cmd_Beep:   db "BEEP", 0
                                <1>
000107AB 00                     <1>             db 0
                                <1>
                                <1> ; 15/02/2016 (FILE.ASM, 09/10/2011)
                                <1> invalid_fname_chars:
000107AC 222728292A2B2C2F       <1>             db 22h, 27h, 28h, 29h, 2Ah, 2Bh, 2Ch, 2Fh
000107B4 3A3B3C3D3E3F40         <1>             db 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh, 40h
000107BB 5B5C5D5E60             <1>             db 5Bh, 5Ch, 5Dh, 5Eh, 60h
                                <1> sizeInvFnChars  equ ($ - invalid_fname_chars)
                                <1> ;
                                <1>
                                <1> Msg_Enter_Date:
000107C0 456E746572206E6577-    <1>                 db 'Enter new date (dd-mm-yy): '
000107C9 206461746520286464-    <1>
000107D2 2D6D6D2D7979293A20     <1>
000107DB 00                     <1>                 db 0
                                <1> Msg_Show_Date:
000107DC 43757272656E742064-    <1>                 db   'Current date is '
000107E5 61746520697320         <1>
000107EC 30                     <1> Day:            db   '0'
000107ED 30                     <1>             db   '0'
000107EE 2F                     <1>                 db   '/'
000107EF 30                     <1> Month:          db   '0'
000107F0 30                     <1>             db   '0'
000107F1 2F                     <1>                 db   '/'
000107F2 30                     <1> Century:        db   '0'
000107F3 30                     <1>                 db   '0'
000107F4 30                     <1> Year:           db   '0'
000107F5 30                     <1>             db   '0'
000107F6 0D0A00                 <1>                 db   0Dh, 0Ah, 0
                                <1>
                                <1> Msg_Enter_Time:
000107F9 456E746572206E6577-    <1>             db 'Enter new time: '
00010802 2074696D653A20         <1>
00010809 00                     <1>             db 0
                                <1> Msg_Show_Time:
0001080A 43757272656E742074-    <1>             db   'Current time is '
00010813 696D6520697320         <1>
0001081A 30                     <1> Hour:           db   '0'
0001081B 30                     <1>             db   '0'
0001081C 3A                     <1>             db   ':'
0001081D 30                     <1> Minute:         db   '0'
```

```
119 0001081E 30                  <1>             db   '0'
120 0001081F 3A                  <1>             db   ':'
121 00010820 30                  <1> Second:         db   '0'
122 00010821 30                  <1>             db   '0'
123 00010822 0D0A00              <1>             db   0Dh, 0Ah, 0
124                              <1>
125                              <1> ;VolSize_Unit1:   dd 0
126                              <1> ;VolSize_Unit2:   dd 0
127                              <1>
128                              <1> VolSize_KiloBytes:
129 00010825 206B696C6F62797465- <1>             db " kilobytes", 0Dh, 0Ah, 0
129 0001082E 730D0A00            <1>
130                              <1> VolSize_Bytes:
131 00010832 2062797465730D0A00  <1>             db " bytes", 0Dh, 0Ah, 0
132                              <1> Volume_in_drive:
133 0001083B 0D0A                <1>             db 0Dh, 0Ah
134                              <1> Vol_FS_Name:
135 0001083D 54522046533120      <1>             db "TR FS1 "
136 00010844 566F6C756D6520696E- <1>             db "Volume in drive "
136 0001084D 20647269766520      <1>
137 00010854 30                  <1> Vol_Drv_Name:   db 30h
138 00010855 3A                  <1>             db ":"
139 00010856 20697320            <1>             db " is "
140 0001085A 0D0A00              <1>             db 0Dh, 0Ah, 0
141                              <1> Dir_Drive_Str:
142 0001085D 54522D444F53204472- <1>                 db "TR-DOS Drive "
142 00010866 69766520            <1>
143                              <1> Dir_Drive_Name:
144 0001086A 303A                <1>                 db "0:"
145 0001086C 0D0A                <1>                 db  0Dh, 0Ah
146                              <1> Vol_Str_Header:
147 0001086E 566F6C756D65204E61- <1>                 db "Volume Name: "
147 00010877 6D653A20            <1>
148                              <1> Vol_Name:
149 0001087B 00<rept>            <1>             times 64 db 0
150 000108BB 00                  <1>             db 0
151                              <1> Vol_Serial_Header:
152 000108BC 0D0A                <1>             db 0Dh, 0Ah
153 000108BE 566F6C756D65205365- <1>             db "Volume Serial No: "
153 000108C7 7269616C204E6F3A20  <1>
154                              <1> Vol_Serial1:
155 000108D0 30303030            <1>             db "0000"
156 000108D4 2D                  <1>             db "-"
157                              <1> Vol_Serial2:
158 000108D5 30303030            <1>             db "0000"
159 000108D9 0D0A00              <1>             db 0Dh, 0Ah, 0
160                              <1>
161                              <1> ;Vol_Tot_Sec_Str_Start:
162                              <1> ;           dd 0
163                              <1> Vol_Total_Sector_Header:
164 000108DC 0D0A                <1>             db 0Dh, 0Ah
165 000108DE 566F6C756D65205369- <1>             db "Volume Size : ", 0
165 000108E7 7A65203A2000        <1>
166                              <1> ;Vol_Tot_Sec_Str:
167                              <1> ;           db "0000000000"
168                              <1> ;Vol_Tot_Sec_Str_End:
169                              <1> ;           db 0
170                              <1> ;Vol_Free_Sectors_Str_Start:
171                              <1> ;           dd 0
172                              <1> Vol_Free_Sectors_Header:
173 000108ED 467265652053706163- <1>             db "Free Space  : ", 0
173 000108F6 6520203A2000        <1>
174                              <1> ;Vol_Free_Sectors_Str:
175                              <1> ;           db "0000000000"
176                              <1> ;Vol_Free_Sectors_Str_End:
177                              <1> ;           db 0
178                              <1>
179                              <1> Dir_Str_Header:
180 000108FC 4469726563746F7279- <1>                 db "Directory: "
180 00010905 3A20                <1>
181 00010907 2F                  <1> Dir_Str_Root:   db "/"
182 00010908 00<rept>            <1> Dir_Str:        times 64 db 0
183 00010948 00000000            <1>                 dd 0
184 0001094C 00                  <1>                 db 0
185                              <1>
186                              <1> Msg_Bad_Command:
187 0001094D 42616420636F6D6D61- <1>                 db "Bad command or file name!"
187 00010956 6E64206F722066696C- <1>
187 0001095F 65206E616D6521      <1>
188 00010966 0D0A00              <1>                 db 0Dh, 0Ah, 0
189                              <1>
190                              <1> msgl_drv_not_ready:
191 00010969 070D0A              <1>             db 07h, 0Dh, 0Ah
192                              <1>
193                              <1> ; CMD_INTR.ASM - 09/11/2011 - Messages
194                              <1>
195                              <1> Msg_Not_Ready_Read_Err:
196 0001096C 4472697665206E6F74- <1>                 db "Drive not ready or read error!"
196 00010975 207265616479206F72- <1>
196 0001097E 207265616420657272- <1>
196 00010987 6F7221              <1>
197 0001098A 0D0A00              <1>                 db 0Dh, 0Ah, 0
198                              <1>
199                              <1> Msg_Not_Ready_Write_Err:
200 0001098D 4472697665206E6F74- <1>                 db "Drive not ready or write error!"
200 00010996 207265616479206F72- <1>
200 0001099F 207772697465206572- <1>
200 000109A8 726F7221            <1>
201 000109AC 0D0A00              <1>                 db 0Dh, 0Ah, 0
202                              <1>
203                              <1> Msg_Dir_Not_Found:
204 000109AF 4469726563746F7279- <1>                 db "Directory not found!"
```

```
204 000109B8 206E6F7420666F756E- <1>
204 000109C1 6421               <1>
205 000109C3 0D0A00             <1>                 db 0Dh, 0Ah, 0
206                             <1>
207                             <1> Msg_File_Not_Found:
208 000109C6 46696C65206E6F7420- <1>                 db "File not found!"
208 000109CF 666F756E6421        <1>
209 000109D5 0D0A00             <1>                 db 0Dh, 0Ah, 0
210                             <1>
211                             <1> Msg_File_Directory_Not_Found:
212 000109D8 46696C65206F722064- <1>                 db "File or directory not found!"
212 000109E1 69726563746F727920- <1>
212 000109EA 6E6F7420666F756E64- <1>
212 000109F3 21                 <1>
213 000109F4 0D0A00             <1>                 db 0Dh, 0Ah, 0
214                             <1>
215                             <1> Msg_LongName_Not_Found:
216 000109F7 4C6F6E67206E616D65- <1>                 db "Long name not found!"
216 00010A00 206E6F7420666F756E- <1>
216 00010A09 6421               <1>
217 00010A0B 0D0A00             <1>                 db 0Dh, 0Ah, 0
218                             <1>
219                             <1> beep_Insufficient_Memory: ; 20/02/2017
220 00010A0E 0D0A               <1>                 db 0Dh, 0Ah
221 00010A10 07                 <1>                 db 07h
222                             <1> Msg_Insufficient_Memory:
223 00010A11 496E73756666696369- <1>                 db "Insufficient memory!"
223 00010A1A 656E74206D656D6F72- <1>
223 00010A23 7921               <1>
224 00010A25 0D0A00             <1>                 db 0Dh, 0Ah, 0
225                             <1>
226                             <1> Msg_Error_Code:
227 00010A28 436F6D6D616E642066- <1>                 db 'Command failed! Error code : '
227 00010A31 61696C656421204572- <1>
227 00010A3A 726F7220636F646520- <1>
227 00010A43 3A20               <1>
228 00010A45 303068             <1> error_code_hex: db '00h'
229 00010A48 0A0A00             <1>                 db 0Ah, 0Ah, 0
230                             <1>
231 00010A4B 90                 <1> align 2
232                             <1>
233                             <1> ; 10/02/2016
234                             <1> ; DIR.ASM - 09/10/2011
235                             <1>
236 00010A4C 3C4449523E20202020- <1> Type_Dir:       db '<DIR>     ' ; 10 bytes
236 00010A55 20                 <1>
237                             <1>
238                             <1> File_Name:
239 00010A56 20<rept>           <1>                 times 12 db 20h
240 00010A62 20                 <1>             db 20h
241                             <1> Dir_Or_FileSize:
242 00010A63 20<rept>           <1>                 times 10 db 20h
243 00010A6D 20                 <1>             db 20h
244                             <1> File_Attribute:
245 00010A6E 20202020           <1>             dd 20202020h
246 00010A72 20                 <1>             db 20h
247                             <1> File_Day:
248 00010A73 3030               <1>                 db '0','0'
249 00010A75 2F                 <1>             db '/'
250                             <1> File_Month:
251 00010A76 3030               <1>                 db '0','0'
252 00010A78 2F                 <1>             db '/'
253                             <1> File_Year:
254 00010A79 30303030           <1>                 db '0','0','0','0'
255 00010A7D 20                 <1>             db 20h
256                             <1> File_Hour:
257 00010A7E 3030               <1>                 db '0','0'
258 00010A80 3A                 <1>             db ':'
259                             <1> File_Minute:
260 00010A81 3030               <1>                 db '0','0'
261 00010A83 00                 <1>             db 0
262                             <1>
263                             <1> Decimal_File_Count_Header:
264 00010A84 0D0A               <1>             db 0Dh, 0Ah
265                             <1> Decimal_File_Count:
266 00010A86 00<rept>           <1>             times 6 db 0
267                             <1>
268 00010A8C 2066696C6528732920- <1> str_files:   db " file(s) & "
268 00010A95 2620               <1>
269                             <1> Decimal_Dir_Count:
270 00010A97 00<rept>           <1>             times 6 db 0
271                             <1> str_dirs:
272 00010A9D 206469726563746F72- <1>             db " directory(s) "
272 00010AA6 7928732920         <1>
273 00010AAB 0D0A00             <1>             db 0Dh, 0Ah, 0
274                             <1>
275 00010AAE 206279746528732920- <1> str_bytes:   db " byte(s) in file(s)"
275 00010AB7 696E2066696C652873- <1>
275 00010AC0 29                 <1>
276 00010AC1 0D0A00             <1>             db 0Dh, 0Ah, 0
277                             <1>
278                             <1> ; CMD_INTR.ASM - 09/11/2011
279                             <1> ; 07/10/2010
280                             <1> Msg_invalid_name_chars:
281 00010AC4 496E76616C69642066- <1>                 db "Invalid file or directory name characters!"
281 00010ACD 696C65206F72206469- <1>
281 00010AD6 726563746F7279206E- <1>
281 00010ADF 616D65206368617261- <1>
281 00010AE8 637465727321       <1>
282 00010AEE 0D0A00             <1>             db 0Dh, 0Ah, 0
283                             <1> ; 21/02/2016
284 00010AF1 46696C65206F722064- <1> Msg_Name_Exists: db "File or directory name exists!"
```

```
284 00010AFA 69726563746F727920- <1>
284 00010B03 6E616D652065786973- <1>
284 00010B0C 747321              <1>
285 00010B0F 0D0A00              <1>                 db 0Dh, 0Ah, 0
286                              <1> Msg_DoYouWantMkdir:
287 00010B12 446F20796F75207761- <1>                 db "Do you want to make directory ", 0
287 00010B1B 6E7420746F206D616B- <1>
287 00010B24 65206469726563746F- <1>
287 00010B2D 72792000            <1>
288 00010B31 2028592F4E29203F20- <1> Msg_YesNo:      db " (Y/N) ? ", 0
288 00010B3A 00                  <1>
289 00010B3B 000D0A00            <1> Y_N_nextline:      db 0, 0Dh, 0Ah, 0
290 00010B3F 4F4B2E0D0A00        <1> Msg_OK:            db "OK.", 0Dh, 0Ah, 0
291                              <1>
292                              <1> ; 27/02/2016
293                              <1> Msg_DoYouWantRmDir:
294 00010B45 446F20796F75207761- <1>                 db "Do you want to delete directory ", 0
294 00010B4E 6E7420746F2064656C- <1>
294 00010B57 657465206469726563- <1>
294 00010B60 746F72792000        <1>
295                              <1> Msg_Dir_Not_Empty:
296 00010B66 4469726563746F7279- <1>                 db "Directory not empty!"
296 00010B6F 206E6F7420656D7074- <1>
296 00010B78 7921                <1>
297 00010B7A 0D0A00              <1>                 db 0Dh, 0Ah, 0
298                              <1>
299                              <1> Msg_DoYouWantDelete:
300 00010B7D 446F20796F75207761- <1>                 db "Do you want to delete file ",0
300 00010B86 6E7420746F2064656C- <1>
300 00010B8F 6574652066696C6520- <1>
300 00010B98 00                  <1>
301                              <1>
302 00010B99 44656C657465642E2E- <1> Msg_Deleted:    db "Deleted...", 0Dh, 0Ah, 0
302 00010BA2 2E0D0A00            <1>
303                              <1>
304                              <1> Msg_Permission_Denied:
305 00010BA6 07                  <1>                 db 7
306 00010BA7 5065726D697373696F- <1>                 db "Permission denied!", 0Dh, 0Ah, 0
306 00010BB0 6E2064656E69656421- <1>
306 00010BB9 0D0A00              <1>
307                              <1>
308                              <1> ; 04/03/2016
309 00010BBC 4E657720            <1> Msg_New:        db "New "
310 00010BC0 00                  <1>                 db 0
311                              <1> Str_Attributes:
312 00010BC1 417474726962757465- <1>                 db "Attributes : "
312 00010BCA 73203A20            <1>
313 00010BCE 4E4F524D414C        <1> Attr_Chars:     db "NORMAL"
314 00010BD4 00                  <1>                 db 0
315                              <1>
316                              <1> ; 06/03/2016
317                              <1> ; CMD_INTR.ASM - 16/11/2010
318                              <1> Msg_DoYouWantRename:
319 00010BD5 446F20796F75207761- <1>                 db "Do you want to rename ", 0
319 00010BDE 6E7420746F2072656E- <1>
319 00010BE7 616D652000          <1>
320 00010BEC 66696C652000        <1> Rename_File:    db "file ", 0
321 00010BF2 6469726563746F7279- <1> Rename_Directory: db "directory ", 0
321 00010BFB 2000                <1>
322 00010BFD 00<rept>            <1> Rename_OldName: times 13 db 0
323 00010C0A 20617320            <1> Msg_File_rename_as: db " as "
324 00010C0E 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 00010C1B 4E6F742073616D6520- <1>                 db "Not same drive!"
329 00010C24 647269766521        <1>
330 00010C2A 0D0A00              <1>                 db 0Dh, 0Ah, 0
331                              <1>
332                              <1> Msg_DoYouWantMoveFile:
333 00010C2D 446F20796F75207761- <1>                 db "Do you want to move file", 0
333 00010C36 6E7420746F206D6F76- <1>
333 00010C3F 652066696C6500      <1>
334                              <1>
335                              <1> msg_insufficient_disk_space:
336 00010C46 496E73756666696369- <1>                 db "Insufficient disk space!"
336 00010C4F 656E74206469736B20- <1>
336 00010C58 737061636521        <1>
337 00010C5E 0D0A00              <1>                 db 0Dh, 0Ah, 0
338                              <1>
339                              <1> ; 01/08/2010
340                              <1> msg_source_file:
341 00010C61 0D0A536F7572636520- <1>                 db 0Dh, 0Ah, "Source file name      :   "
341 00010C6A 66696C65206E616D65- <1>
341 00010C73 2020202020203A2020- <1>
341 00010C7C 20                  <1>
342                              <1> msg_source_file_drv:
343 00010C7D 203A00              <1>                 db " :", 0
344                              <1> msg_destination_file:
345 00010C80 0D0A44657374696E61- <1>                 db 0Dh, 0Ah, "Destination file name :   "
345 00010C89 74696F6E2066696C65- <1>
345 00010C92 206E616D65203A2020- <1>
345 00010C9B 20                  <1>
346                              <1> msg_destination_file_drv:
347 00010C9C 203A00              <1>                 db " :", 0
348                              <1> msg_copy_nextline:
349 00010C9F 0D0A00              <1>                 db 0Dh, 0Ah, 0
350                              <1>
351                              <1> ; 15/03/2016
352                              <1> ; CMD_INTR.ASM
353                              <1>
```

```
                            <1> Msg_DoYouWantOverWriteFile:
00010CA2 446F20796F75207761- <1>                 db "Do you want to overwrite file ",0
00010CAB 6E7420746F206F7665- <1>
00010CB4 727772697465206669- <1>
00010CBD 6C652000            <1>
                            <1>
                            <1> Msg_DoYouWantCopyFile:
00010CC1 446F20796F75207761- <1>                 db "Do you want to copy file",0
00010CCA 6E7420746F20636F70- <1>
00010CD3 792066696C6500      <1>
                            <1>
                            <1> Msg_read_file_error_before_EOF:
00010CDA 46696C652072656164- <1>             db "File reading error! (before EOF)"
00010CE3 696E67206572726F72- <1>
00010CEC 2120286265666F7265- <1>
00010CF5 20454F4629          <1>
00010CFA 0A0A00              <1>             db 0Ah, 0Ah, 0
                            <1>
                            <1> ; 18/03/2016
                            <1> ; TRDOS 386 (v2.0) mainprog copy procedure
                            <1> msg_reading:
00010CFD 52656164696E672E2E- <1>             db "Reading... ", 0
00010D06 2E2000              <1>
                            <1> msg_writing:
00010D09 57726974696E672E2E- <1>             db "Writing... ", 0
00010D12 2E2000              <1>
                            <1> percentagestr:
00010D15 2020202500          <1>             db "   %", 0  ; "  0%" .. "100%"
                            <1> ; 11/04/2016
                            <1> Msg_No_Set_Space:
00010D1A 496E73756666696369- <1>                 db "Insufficient environment space!"
00010D23 656E7420656E766972- <1>
00010D2C 6F6E6D656E74207370- <1>
00010D35 61636521            <1>
00010D39 0D0A00              <1>                 db 0Dh, 0Ah, 0
                            <1> ; 18/04/2016
                            <1> isc_msg:
00010D3C 0D0A                <1>             db 0Dh, 0Ah
00010D3E 494E56414C49442053- <1>             db "INVALID SYSTEM CALL", 0
00010D47 595354454D2043414C- <1>
00010D50 4C00                <1>
                            <1> usi_msg:
00010D52 0D0A                <1>             db 0Dh, 0Ah
00010D54 554E444546494E4544- <1>             db "UNDEFINED SOFTWARE INTERRUPT", 0
00010D5D 20534F465457415245- <1>
00010D66 20494E544552525550- <1>
00010D6F 5400                <1>
                            <1> ifc_msg:
00010D71 0D0A                <1>             db 0Dh, 0Ah
00010D73 494E56414C49442046- <1>             db "INVALID FUNCTION CALL"
00010D7C 554E4354494F4E2043- <1>
00010D85 414C4C              <1>
                            <1> inv_msg_for_trdos_v2:
00010D88 20                  <1>             db 20h
00010D89 666F72205452444F53- <1>             db "for TRDOS v2!"
00010D92 20763221            <1>
00010D96 07                  <1>             db 07h
00010D97 0D0A                <1>             db 0Dh, 0Ah
00010D99 0D0A                <1>             db 0Dh, 0Ah
00010D9B 494E5420            <1>             db "INT "
00010D9F 303068              <1> int_num_str: db "00h"
00010DA2 0D0A                <1>             db 0Dh, 0Ah
00010DA4 454158203A20        <1>             db "EAX : "
00010DAA 303030303030303068- <1> eax_str:    db "00000000h", 0Dh, 0Ah
00010DB3 0D0A                <1>
00010DB5 454950203A20        <1>             db "EIP : "
00010DBB 303030303030303068- <1> eip_str:    db "00000000h", 0Dh, 0Ah, 0
00010DC4 0D0A00              <1>
                            <1>
                            <1> ; 07/10/2016
                            <1> ; Device names & parameters (for kernel devices)
                            <1>
                            <1> align 2
                            <1> KDEV_NAME:
00010DC8 5454590000000000    <1>             db 'TTY',0,0,0,0,0 ; 1
00010DD0 4D454D0000000000    <1>             db 'MEM',0,0,0,0,0 ; 2
00010DD8 4644300000000000    <1>             db 'FD0',0,0,0,0,0 ; 3
00010DE0 4644310000000000    <1>             db 'FD1',0,0,0,0,0 ; 4
00010DE8 4844300000000000    <1>             db 'HD0',0,0,0,0,0 ; 5
00010DF0 4844310000000000    <1>             db 'HD1',0,0,0,0,0 ; 6
00010DF8 4844320000000000    <1>             db 'HD2',0,0,0,0,0 ; 7
00010E00 4844330000000000    <1>             db 'HD3',0,0,0,0,0 ; 8
00010E08 4C50540000000000    <1>             db 'LPT',0,0,0,0,0 ; 9
00010E10 5454593000000000    <1>             db 'TTY0',0,0,0,0 ; 10
00010E18 5454593100000000    <1>             db 'TTY1',0,0,0,0 ; 11
00010E20 5454593200000000    <1>             db 'TTY2',0,0,0,0 ; 12
00010E28 5454593300000000    <1>             db 'TTY3',0,0,0,0 ; 13
00010E30 5454593400000000    <1>             db 'TTY4',0,0,0,0 ; 14
00010E38 5454593500000000    <1>             db 'TTY5',0,0,0,0 ; 15
00010E40 5454593600000000    <1>             db 'TTY6',0,0,0,0 ; 16
00010E48 5454593700000000    <1>             db 'TTY7',0,0,0,0 ; 17
00010E50 5454593800000000    <1>             db 'TTY8',0,0,0,0 ; 18
00010E58 5454593900000000    <1>             db 'TTY9',0,0,0,0 ; 19
00010E60 434F4D3100000000    <1>             db 'COM1',0,0,0,0 ; 18
00010E68 434F4D3200000000    <1>             db 'COM2',0,0,0,0 ; 19
                            <1>             ;db 'CONSOLE',0       ; 1
                            <1>             ;db 'PRINTER',0   ; 9
                            <1>             ;db 'CDROM'    ; 20
                            <1>             ;db 'CDROM0'   ; 20
                            <1>             ;db 'CDROM1'   ; 21
                            <1>             ;db 'DVD'      ; 22
                            <1>             ;db 'DVD0'     ; 22
```

```
                                <1>             ;db 'DVD1'     ; 23
                                <1>             ;db 'USB'      ; 24
                                <1>             ;db 'USB0'     ; 24
                                <1>             ;db 'USB1'     ; 25
                                <1>             ;db 'USB2'     ; 26
                                <1>             ;db 'USB3'       ; 27
                                <1>             ;db 'KEYBOARD'       ; 1
                                <1>             ;db 'MOUSE'    ; 28
                                <1>             ;db 'SOUND'    ; 29
                                <1>             ;db 'VGA',0,0,0,0 ; 30
                                <1>             ;db 'CGA',0,0,0,0 ; 31
                                <1>             ;db 'AUDIO',0,0,0 ; 29
                                <1>             ;db 'VIDEO',0,0,0 ; 32
                                <1>             ;db 'MUSIC',0,0,0 ; 33
                                <1>             ;db 'ETHERNET'       ; 34
                                <1>             ;db 'SD0',0,0,0,0,0 ; 35
                                <1>             ;db 'SD1',0,0,0,0,0 ; 36
                                <1>             ;db 'SD2',0,0,0,0,0 ; 37
                                <1>             ;db 'SD3',0,0,0,0,0 ; 38
                                <1>             ;db 'SATA0'    ; 35
                                <1>             ;db 'SATA1'    ; 36
                                <1>             ;db 'SATA2'       ; 37
                                <1>             ;db 'SATA3'       ; 38
                                <1>             ;db 'PATA0',0,0,0  ; 5
                                <1>             ;db 'PATA1',0,0,0  ; 6
                                <1>             ;db 'PATA2',0,0,0  ; 7
                                <1>             ;db 'PATA3',0,0,0  ; 8
                                <1>             ;db 'WIRELESS'       ; 39
                                <1>             ;db 'HDMI',0,0,0,0 ; 40
00010E70 4E554C4C00000000       <1>             db 'NULL',0,0,0,0 ; 0
                                <1>
                                <1> NumOfKernelDevNames equ ($-KDEV_NAME) / 8 ; 20 (07/10/2016)
                                <1>
                                <1> KDEV_NUMBER:
00010E78 010203040506070809     <1>             db 1,2,3,4,5,6,7,8,9
00010E81 0A0B0C0D0E0F101112-    <1>             db 10,11,12,13,14,15,16,17,18,19
00010E8A 13                     <1>
00010E8B 121300                 <1>             db 18,19,0
                                <1>
                                <1> NumOfKernelDevices equ $ - KDEV_NUMBER
                                <1>
                                <1> KDEV_OADDR:
00010E8E [AB060100]             <1>             dd otty ;tty  ; 1
00010E92 [AB060100]             <1>             dd sret ;mem  ; 2
00010E96 [AB060100]             <1>             dd sret ;fd0  ; 3
00010E9A [AB060100]             <1>             dd sret ;fd1  ; 4
00010E9E [AB060100]             <1>             dd sret ;hd0  ; 5
00010EA2 [AB060100]             <1>             dd sret ;hd1  ; 6
00010EA6 [AB060100]             <1>             dd sret ;hd2  ; 7
00010EAA [AB060100]             <1>             dd sret ;hd3  ; 8
00010EAE [AB060100]             <1>             dd sret ;lpt  ; 9
00010EB2 [AB060100]             <1>             dd ocvt ;tty0 ; 10
00010EB6 [AB060100]             <1>             dd ocvt ;tty1 ; 11
00010EBA [AB060100]             <1>             dd ocvt ;tty2 ; 12
00010EBE [AB060100]             <1>             dd ocvt ;tty3 ; 13
00010EC2 [AB060100]             <1>             dd ocvt ;tty4 ; 14
00010EC6 [AB060100]             <1>             dd ocvt ;tty5 ; 15
00010ECA [AB060100]             <1>             dd ocvt ;tty6 ; 16
00010ECE [AB060100]             <1>             dd ocvt ;tty7 ; 17
00010ED2 [AB060100]             <1>             dd ocvt ;tty8 ; 18
00010ED6 [AB060100]             <1>             dd ocvt ;tty9 ; 19
                                <1>             ;dd ocvt ;com1 ; 18
                                <1>             ;dd ocvt ;com2 ; 19
00010EDA [AB060100]             <1>             dd sret ;null ; 20
                                <1> KDEV_CADDR:
00010EDE [AB060100]             <1>             dd ctty ;tty  ; 1
00010EE2 [AB060100]             <1>             dd cret ;mem  ; 2
00010EE6 [AB060100]             <1>             dd cret ;fd0  ; 3
00010EEA [AB060100]             <1>             dd cret ;fd1  ; 4
00010EEE [AB060100]             <1>             dd cret ;hd0  ; 5
00010EF2 [AB060100]             <1>             dd cret ;hd1  ; 6
00010EF6 [AB060100]             <1>             dd cret ;hd2  ; 7
00010EFA [AB060100]             <1>             dd cret ;hd3  ; 8
00010EFE [AB060100]             <1>             dd cret ;lpt  ; 9
00010F02 [AB060100]             <1>             dd ocvt ;tty0 ; 10
00010F06 [AB060100]             <1>             dd ccvt ;tty1 ; 11
00010F0A [AB060100]             <1>             dd ccvt ;tty2 ; 12
00010F0E [AB060100]             <1>             dd ccvt ;tty3 ; 13
00010F12 [AB060100]             <1>             dd ccvt ;tty4 ; 14
00010F16 [AB060100]             <1>             dd ccvt ;tty5 ; 15
00010F1A [AB060100]             <1>             dd ccvt ;tty6 ; 16
00010F1E [AB060100]             <1>             dd ccvt ;tty7 ; 17
00010F22 [AB060100]             <1>             dd ccvt ;tty8 ; 18
00010F26 [AB060100]             <1>             dd ccvt ;tty9 ; 19
                                <1>             ;dd ccvt ;com1 ; 18
                                <1>             ;dd ccvt ;com2 ; 19
00010F2A [AB060100]             <1>             dd cret ;null ; 20
                                <1>
                                <1> KDEV_RADDR:
00010F2E [AB060100]             <1>             dd rtty ;tty  ; 1
00010F32 [AB060100]             <1>             dd rmem ;mem  ; 2
00010F36 [AB060100]             <1>             dd rfd  ;fd0  ; 3
00010F3A [AB060100]             <1>             dd rfd  ;fd1  ; 4
00010F3E [AB060100]             <1>             dd rhd  ;hd0  ; 5
00010F42 [AB060100]             <1>             dd rhd  ;hd1  ; 6
00010F46 [AB060100]             <1>             dd rhd  ;hd2  ; 7
00010F4A [AB060100]             <1>             dd rhd  ;hd3  ; 8
00010F4E [AB060100]             <1>             dd rlpt ;lpt  ; 9
00010F52 [AB060100]             <1>             dd rcvt ;tty0 ; 10
00010F56 [AB060100]             <1>             dd rcvt ;tty1 ; 11
00010F5A [AB060100]             <1>             dd rcvt ;tty2 ; 12
00010F5E [AB060100]             <1>             dd rcvt ;tty3 ; 13
```

```
534 00010F62 [AB060100]          <1>             dd rcvt ;tty4 ; 14
535 00010F66 [AB060100]          <1>             dd rcvt ;tty5 ; 15
536 00010F6A [AB060100]          <1>             dd rcvt ;tty6 ; 16
537 00010F6E [AB060100]          <1>             dd rcvt ;tty7 ; 17
538 00010F72 [AB060100]          <1>             dd rcvt ;tty8 ; 18
539 00010F76 [AB060100]          <1>             dd rcvt ;tty9 ; 19
540                              <1>             ;dd rcvt ;com1 ; 18
541                              <1>             ;dd rcvt ;com2 ; 19
542 00010F7A [9CFA0000]          <1>             dd rnull ;null ; 20
543                              <1> KDEV_WADDR:
544 00010F7E [AB060100]          <1>             dd wtty ;tty  ; 1
545 00010F82 [AB060100]          <1>             dd wmem ;mem  ; 2
546 00010F86 [AB060100]          <1>             dd wfd  ;fd0  ; 3
547 00010F8A [AB060100]          <1>             dd wfd  ;fd1  ; 4
548 00010F8E [AB060100]          <1>             dd whd  ;hd0  ; 5
549 00010F92 [AB060100]          <1>             dd whd  ;hd1  ; 6
550 00010F96 [AB060100]          <1>             dd whd  ;hd2  ; 7
551 00010F9A [AB060100]          <1>             dd whd  ;hd3  ; 8
552 00010F9E [AB060100]          <1>             dd wlpt ;lpt  ; 9
553 00010FA2 [AB060100]          <1>             dd xmtt ;tty0 ; 10
554 00010FA6 [AB060100]          <1>             dd xmtt ;tty1 ; 11
555 00010FAA [AB060100]          <1>             dd xmtt ;tty2 ; 12
556 00010FAE [AB060100]          <1>             dd xmtt ;tty3 ; 13
557 00010FB2 [AB060100]          <1>             dd xmtt ;tty4 ; 14
558 00010FB6 [AB060100]          <1>             dd xmtt ;tty5 ; 15
559 00010FBA [AB060100]          <1>             dd xmtt ;tty6 ; 16
560 00010FBE [AB060100]          <1>             dd xmtt ;tty7 ; 17
561 00010FC2 [AB060100]          <1>             dd xmtt ;tty8 ; 18
562 00010FC6 [AB060100]          <1>             dd xmtt ;tty9 ; 19
563                              <1>             ;dd xmtt ;com1 ; 18
564                              <1>             ;dd xmtt ;com2 ; 19
565 00010FCA [9DFA0000]          <1>             dd wnull ;null ; 20
566                              <1>
567                              <1> ; DEV_ACCESS bits:
568                              <1>       ; bit 0 = accessable by normal users
569                              <1>       ; bit 1 = read access permission
570                              <1>       ; bit 2 = write access permission
571                              <1>       ; bit 3 = IOCTL permission to users
572                              <1>       ; bit 4 = block device if it is set
573                              <1>       ; bit 5 = 16 bit or 1024 byte data
574                              <1>       ; bit 6 = 32 bit or 2048 byte data
575                              <1>       ; bit 7 = installable device driver
576                              <1>
577                              <1> KDEV_ACCESS: ; 08/10/2016
578 00010FCE 07                  <1>             db  00000111b; tty, 1
579 00010FCF 07                  <1>             db  00000111b; mem, 2
580 00010FD0 8F                  <1>             db  10001111b; fd0, 3
581 00010FD1 8F                  <1>             db  10001111b; fd1, 4
582 00010FD2 8F                  <1>             db  10001111b; hd0, 5
583 00010FD3 8F                  <1>             db  10001111b; hd1, 6
584 00010FD4 8F                  <1>             db  10001111b; hd2, 7
585 00010FD5 8F                  <1>             db  10001111b; hd3, 8
586 00010FD6 07                  <1>             db  00000111b   ; lpt, 9
587 00010FD7 07                  <1>             db  00000111b; tty0, 10
588 00010FD8 07                  <1>             db  00000111b; tty1, 11
589 00010FD9 07                  <1>             db  00000111b; tty2, 12
590 00010FDA 07                  <1>             db  00000111b; tty3, 13
591 00010FDB 07                  <1>             db  00000111b; tty4, 14
592 00010FDC 07                  <1>             db  00000111b; tty5, 15
593 00010FDD 07                  <1>             db  00000111b; tty6, 16
594 00010FDE 07                  <1>             db  00000111b; tty7, 17
595 00010FDF 07                  <1>             db  00000111b; tty8, 18
596 00010FE0 07                  <1>             db  00000111b; tty9, 19
597                              <1>             ;db 00000111b; com1, 18
598                              <1>             ;db 00000111b; com2, 19
599 00010FE1 00                  <1>             db  00000000b   ; null, 0
600                              <1>
601                              <1> ; 07/10/2016
602                              <1> NumOfInstallableDevices equ 8
603                              <1> NUMIDEV           equ NumOfInstallableDevices ; 8
604                              <1> NUMOFDEVICES equ NumOfKernelDevices + NumOfInstallableDevices
605                              <1>
606                              <1> ; 26/02/2017
607                              <1> ; IRQ Callback (& Signal Response Byte) service availibity
608                              <1> ; 'syscalbac'
609                              <1> ; ************************************************
610                              <1> ; IRQ 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
611                              <1> ; --- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
612                              <1> ; --- 00 00 00 01 02 03 00 04 00 05 06 07 08 09 00 00
613                              <1> ; ************************************************
614                              <1> IRQenum:
615 00010FE2 00000001020300040- <1>       db  0,0,0,1,2,3,0,4,0,5,6,7,8,9,0,0
615 00010FEB 05060708090000     <1>
616                              <1>
617                              <1> ; 28/08/2017
618                              <1> ; 20/08/2017
619                              <1> ; DMA Registers (for 'sysdma')
620                              <1> ; 02/07/2017 (sb16mod.s)
621 00010FF2 00020406C0C4C8CC    <1> dma_adr:    db 0,2,4,6,0C0h,0C4h,0C8h,0CCh
622 00010FFA 01030507C2C6CACE    <1> dma_cnt:    db 1,3,5,7,0C2h,0C6h,0CAh,0CEh
623 00011002 878381828F8B898A    <1> dma_page:   db 87h,83h,81h,82h,8Fh,8Bh,89h,8Ah ; 03/08/2017
624 0001100A 0A0A0A0AD4D4D4D4    <1> dma_mask:   db 0Ah,0Ah,0Ah,0Ah,0D4h,0D4h,0D4h,0D4h
625 00011012 0B0B0B0BD6D6D6D6    <1> dma_mod:    db 0Bh,0Bh,0Bh,0Bh,0D6h,0D6h,0D6h,0D6h
626 0001101A 0C0C0C0CD8D8D8D8    <1> dma_flip:   db 0Ch,0Ch,0Ch,0Ch,0D8h,0D8h,0D8h,0D8h
2313
2314                                 ; 27/08/2014
2315                                 scr_row:
2316 00011022 E0810B00                     dd 0B8000h + 0A0h + 0A0h + 0A0h ; Row 3
2317                                 scr_col:
2318 00011026 00000000                     dd 0
2319
2320 0001102A 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
0001102C [17090000]                     dd    exc0   ; 0h,  Divide-by-zero Error
00011030 [1E090000]                     dd    exc1
00011034 [25090000]                     dd    exc2
00011038 [2C090000]                     dd    exc3
0001103C [30090000]                     dd    exc4
00011040 [34090000]                     dd    exc5
00011044 [38090000]                     dd    exc6   ; 06h,  Invalid Opcode
00011048 [3C090000]                     dd    exc7
0001104C [40090000]                     dd    exc8
00011050 [44090000]                     dd    exc9
00011054 [48090000]                     dd    exc10
00011058 [4C090000]                     dd    exc11
0001105C [50090000]                     dd    exc12
00011060 [54090000]                     dd    exc13  ; 0Dh, General Protection Fault
00011064 [58090000]                     dd    exc14  ; 0Eh, Page Fault
00011068 [5C090000]                     dd    exc15
0001106C [60090000]                     dd    exc16
00011070 [64090000]                     dd    exc17
00011074 [68090000]                     dd    exc18
00011078 [6C090000]                     dd    exc19
0001107C [70090000]                     dd    exc20
00011080 [74090000]                     dd    exc21
00011084 [78090000]                     dd    exc22
00011088 [7C090000]                     dd    exc23
0001108C [80090000]                     dd    exc24
00011090 [84090000]                     dd    exc25
00011094 [88090000]                     dd    exc26
00011098 [8C090000]                     dd    exc27
0001109C [90090000]                     dd    exc28
000110A0 [94090000]                     dd    exc29
000110A4 [98090000]                     dd    exc30
000110A8 [9C090000]                     dd    exc31
                                IRQ_list: ; 28/02/2017 ('syscalbac')
                                        ; Interrupt list
000110AC [8B060000]                     dd    timer_int    ; INT 20h
                                              ;dd   irq0
000110B0 [FF0D0000]                     dd    kb_int       ; 24/01/2016
                                              ;dd   irq1
000110B4 [6D080000]                     dd    irq2
                                              ; COM2 int
000110B8 [71080000]                     dd    irq3
                                              ; COM1 int
000110BC [7C080000]                     dd    irq4
000110C0 [87080000]                     dd    irq5
                                ;DISKETTE_INT: ;06/02/2015
000110C4 [B0410000]                     dd    fdc_int            ; 16/02/2015, IRQ 6 handler
                                              ;dd   irq6
                                ; Default IRQ 7 handler against spurious IRQs (from master PIC)
                                ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
000110C8 [F60B0000]                     dd    default_irq7 ; 25/02/2015
                                              ;dd   irq7
                                ; Real Time Clock Interrupt
000110CC [F6070000]                     dd    rtc_int            ; 23/02/2015, IRQ 8 handler
                                              ;dd   irq8   ; INT 28h
000110D0 [97080000]                     dd    irq9
000110D4 [9B080000]                     dd    irq10
000110D8 [9F080000]                     dd    irq11
000110DC [A3080000]                     dd    irq12
000110E0 [A7080000]                     dd    irq13
                                ;HDISK_INT1:  ;06/02/2015
000110E4 [2C4B0000]                     dd    hdc1_int     ; 21/02/2015, IRQ 14 handler
                                              ;dd   irq14
                                ;HDISK_INT2:  ;06/02/2015
000110E8 [534B0000]                     dd    hdc2_int     ; 21/02/2015, IRQ 15 handler
                                              ;dd   irq15  ; INT 2Fh
                                              ; 14/08/2015
                                        ;dd   sysent       ; INT 30h (system calls)

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

000110EC [49110100]                     dd    int30h       ; Reserved for
                                                           ; !!! Retro UNIX (RUNIX) !!!
                                                           ; !!! SINGLIX !!! System Calls
000110F0 [F2140000]                     dd    int31h       ; Video BIOS (IBM PC/AT, Int 10h)
000110F4 [1E0C0000]                     dd    int32h       ; Keyboard Functions (IBM PC/AT, Int 16h)
000110F8 [66420000]                     dd    int33h       ; DISK I/O (IBM PC/AT, Int 13h)
000110FC [4DF30000]                     dd    int34h       ; #IOCTL# (I/O port access support for ring 3)
00011100 [81590000]                     dd    int35h       ; Time/Date Functions (IBM PC/AT, Int 1Ah)
00011104 [AA0A0000]                     dd    ignore_int   ; INT 36h : Timer Functions
00011108 [AA0A0000]                     dd    ignore_int   ; INT 37h
0001110C [AA0A0000]                     dd    ignore_int   ; INT 38h
00011110 [AA0A0000]                     dd    ignore_int   ; INT 39h
00011114 [AA0A0000]                     dd    ignore_int   ; INT 3Ah
00011118 [AA0A0000]                     dd    ignore_int   ; INT 3Bh
0001111C [AA0A0000]                     dd    ignore_int   ; INT 3Ch
00011120 [AA0A0000]                     dd    ignore_int   ; INT 3Dh
00011124 [AA0A0000]                     dd    ignore_int   ; INT 3Eh
00011128 [AA0A0000]                     dd    ignore_int   ; INT 3Fh
0001112C [61C30000]                     dd    sysent       ; INT 40h : !!! TRDOS 386 System Calls !!!
                                        ;dd   ignore_int
00011130 00000000                       dd    0
```

```
                                ; 20/08/2014
                                  ; /* This is the default interrupt "handler" :-) */
                                  ; Linux v0.12 (head.s)
                                int_msg:
00011134 556E6B6E6F776E2069-          db "Unknown interrupt ! ", 0
0001113D 6E7465727275707420-
00011146 212000
                                ; 15/04/2016
                                ; TRDOS 386 (TRDOS v2.0)
                                ; 29/04/2016
                                int30h:
                                trdos_isc_routine:
                                      ; 02/05/2016
                                      ; 01/05/2016
                                      ; 29/04/2016
                                      ; 18/04/2016
                                      ; 15/04/2016 (TRDOS 386 = TRDOS v2.0)
                                      ; 17/04/2011 (TRDOS v1.0, 'IFC.ASM')
                                      ; 03/02/2011 ('trdos_ifc_routine')
                                      ;
00011149 8B1C24                       mov   ebx, [esp] ; EIP (next)
0001114C 83EB02                       sub   ebx, 2 ; EIP (CD ##h)

0001114F 89C1                         mov   ecx, eax
00011151 8A4301                       mov   al, [ebx+1] ; CDh ##h

00011154 66BA1000                     mov   dx, KDATA
00011158 8EDA                         mov   ds, dx
0001115A 8EC2                         mov   es, dx

0001115C FC                           cld
0001115D 8B15[00520100]               mov   edx, [k_page_dir]
00011163 0F22DA                       mov   cr3, edx

00011166 E85E21FFFF                   call  bytetohex
0001116B 66A3[9F0D0100]               mov   [int_num_str], ax

00011171 89D8                         mov   eax, ebx ; EIP
00011173 E89121FFFF                   call  dwordtohex
00011178 8915[BB0D0100]               mov   [eip_str], edx
0001117E A3[BF0D0100]                 mov   [eip_str+4], eax

00011183 89C8                         mov   eax, ecx
00011185 E87F21FFFF                   call  dwordtohex
0001118A 8915[AA0D0100]               mov   [eax_str], edx
00011190 A3[AE0D0100]                 mov   [eax_str+4], eax

00011195 43                           inc   ebx
00011196 8A03                         mov   al, [ebx] ; Interrupt number

                                trdos_isc_handler:
00011198 80FE30                       cmp   dh, 30h ; Retro UNIX, SINGLIX System calls
0001119B 7507                         jne   short trdos_usi_handler
0001119D BE[3C0D0100]                 mov   esi, isc_msg
000111A2 EB05                         jmp   short loc_write_inv_system_call_msg

                                trdos_usi_handler:
000111A4 BE[520D0100]                 mov   esi, usi_msg

                                loc_write_inv_system_call_msg:
000111A9 E8AF51FFFF                   call  print_msg
                                      ; 29/04/2016
000111AE BE[880D0100]                 mov   esi, inv_msg_for_trdos_v2
000111B3 E8A551FFFF                   call  print_msg

                                loc_ifc_terminate_process:
                                      ; u.uno = process number
                                      ; 29/04/2016
                                      ; 02/05/2016
000111B8 FE05[5B030300]               inc   byte [sysflg] ; 0FFh -> 0

000111BE B801000000                   mov   eax, 1
000111C3 E972B4FFFF                   jmp   sysexit

                                ; 07/03/2015
                                ; Temporary Code
                                display_disks:
000111C8 803D[F65C0000]00             cmp   byte [fd0_type], 0
000111CF 7605                         jna   short ddsks1
000111D1 E87D000000                   call  pdskm
                                ddsks1:
000111D6 803D[F75C0000]00             cmp   byte [fd1_type], 0
000111DD 760C                         jna   short ddsks2
000111DF C605[23130100]31             mov   byte [dskx], '1'
000111E6 E868000000                   call  pdskm
                                ddsks2:
000111EB 803D[F85C0000]00             cmp   byte [hd0_type], 0
000111F2 7654                         jna   short ddsk6
000111F4 66C705[21130100]68-          mov   word [dsktype], 'hd'
000111FC 64
000111FD C605[23130100]30             mov   byte [dskx], '0'
00011204 E84A000000                   call  pdskm
                                ddsks3:
00011209 803D[F95C0000]00             cmp   byte [hd1_type], 0
00011210 7636                         jna   short ddsk6
00011212 C605[23130100]31             mov   byte [dskx], '1'
00011219 E835000000                   call  pdskm
                                ddsks4:
0001121E 803D[FA5C0000]00             cmp   byte [hd2_type], 0
```

```
2522 00011225 7621                      jna    short ddsk6
2523 00011227 C605[23130100]32          mov    byte [dskx], '2'
2524 0001122E E820000000                call   pdskm
2525                                ddsks5:
2526 00011233 803D[FB5C0000]00          cmp    byte [hd3_type], 0
2527 0001123A 760C                      jna    short ddsk6
2528 0001123C C605[23130100]33          mov    byte [dskx], '3'
2529 00011243 E80B000000                call   pdskm
2530                                ddsk6:
2531 00011248 BE[4B130100]              mov    esi, nextline
2532 0001124D E806000000                call   pdskml
2533                                pdskm_ok:
2534 00011252 C3                        retn
2535                                pdskm:
2536 00011253 BE[1F130100]              mov    esi, dsk_ready_msg
2537                                pdskml:
2538 00011258 AC                        lodsb
2539 00011259 08C0                      or     al, al
2540 0001125B 74F5                      jz     short pdskm_ok
2541 0001125D 56                        push   esi
2542                                    ; 13/05/2016
2543 0001125E BB07000000                  mov     ebx, 7  ; Black background,
2544                                                      ; light gray forecolor
2545                                                      ; Video page 0 (bh=0)
2546 00011263 E84A0AFFFF                call   _write_tty
2547 00011268 5E                        pop    esi
2548 00011269 EBED                      jmp    short pdskml
2549
2550 0001126B 90                    Align 2
2551                                    ; 21/08/2014
2552                                exc_msg:
2553 0001126C 435055206578636570-       db "CPU exception ! "
2553 00011275 74696F6E202120
2554                                excnstr:           ; 25/08/2014
2555 0001127C 3F3F68202045495020-       db "??h", "  EIP : "
2555 00011285 3A20
2556                                EIPstr: ; 29/08/2014
2557 00011287 00<rept>                  times 12 db 0
2558
2559                                    ; 23/02/2015
2560                                    ; 25/08/2014
2561                                ;scounter:
2562                                ;     db 5
2563                                ;     db 19
2564
2565                                ; 06/11/2014
2566                                ; Memory Information message
2567                                ; 14/08/2015
2568                                msg_memory_info:
2569 00011293 07                        db     07h
2570 00011294 0D0A                      db     0Dh, 0Ah
2571                                    ;db    "MEMORY ALLOCATION INFO", 0Dh, 0Ah, 0Dh, 0Ah
2572 00011296 546F74616C206D656D-       db     "Total memory : "
2572 0001129F 6F7279203A20
2573                                mem_total_b_str: ; 10 digits
2574 000112A5 303030303030303030-       db     "0000000000 bytes", 0Dh, 0Ah
2574 000112AE 302062797465730D0A
2575 000112B7 2020202020202020-         db     "               ", 20h, 20h, 20h
2575 000112C0 2020202020202020
2576                                mem_total_p_str: ; 7 digits
2577 000112C9 303030303030302070-       db     "0000000 pages", 0Dh, 0Ah
2577 000112D2 616765730D0A
2578 000112D8 0D0A                      db     0Dh, 0Ah
2579 000112DA 46726565206D656D6F-       db     "Free memory  : "
2579 000112E3 727920203A20
2580                                free_mem_b_str:  ; 10 digits
2581 000112E9 3F3F3F3F3F3F3F3F3F-       db     "?????????? bytes", 0Dh, 0Ah
2581 000112F2 3F2062797465730D0A
2582 000112FB 2020202020202020-         db     "               ", 20h, 20h, 20h
2582 00011304 2020202020202020
2583                                free_mem_p_str:  ; 7 digits
2584 0001130D 3F3F3F3F3F3F3F2070-       db     "??????? pages", 0Dh, 0Ah
2584 00011316 616765730D0A
2585 0001131C 0D0A00                    db     0Dh, 0Ah, 0
2586
2587                                dsk_ready_msg:
2588 0001131F 0D0A                      db     0Dh, 0Ah
2589                                dsktype:
2590 00011321 6664                      db     'fd'
2591                                dskx:
2592 00011323 30                        db     '0'
2593 00011324 20                        db     20h
2594 00011325 697320524541445920-       db     'is READY ...'
2594 0001132E 2E2E2E
2595 00011331 00                        db     0
2596
2597                                setup_error_msg:
2598 00011332 0D0A                      db 0Dh, 0Ah
2599 00011334 4469736B2053657475-       db 'Disk Setup Error !'
2599 0001133D 70204572726F722021
2600 00011346 0D0A00                    db 0Dh, 0Ah,0
2601
2602                                next2line: ; 08/02/2016
2603 00011349 0D0A                      db     0Dh, 0Ah
2604                                nextline:
2605 0001134B 0D0A00                    db     0Dh, 0Ah, 0
2606
2607                                ; KERNEL - SYSINIT Messages
2608                                ; 24/08/2015
2609                                ; 13/04/2015 - (Retro UNIX 386 v1 Beginning)
2610                                ; 14/07/2013
2611                                ;kernel_init_err_msg:
```

```
2612                                    ;     db 0Dh, 0Ah
2613                                    ;     db 07h
2614                                    ;     db 'Kernel initialization ERROR !'
2615                                    ;     db 0Dh, 0Ah, 0
2616
2617                                    ;welcome_msg:
2618                                    ;     db 0Dh, 0Ah
2619                                    ;     db 07h
2620                                    ;     db 'Welcome to TRDOS 386 Operating System !'
2621                                    ;     db 0Dh, 0Ah
2622                                    ;     db 'by Erdogan Tan - 22/10/2017 (v2.0.0)'
2623                                    ;     db 0Dh, 0Ah, 0
2624
2625                                    panic_msg:
2626 0001134E 0D0A07                         db 0Dh, 0Ah, 07h
2627 00011351 4552524F523A204B65-           db 'ERROR: Kernel Panic !'
2627 0001135A 726E656C2050616E69-
2627 00011363 632021
2628 00011366 0D0A00                        db 0Dh, 0Ah, 0
2629
2630                                    ;msgl_drv_not_ready:
2631                                    ;     db 07h, 0Dh, 0Ah
2632                                    ;       db 'Drive not ready or read error !'
2633                                    ;       db 0Dh, 0Ah, 0
2634
2635                                    starting_msg:
2636 00011369 5475726B6973682052-           db "Turkish Rational DOS v2.0 [22/10/2017] ...", 0
2636 00011372 6174696F6E616C2044-
2636 0001137B 4F532076322E30205B-
2636 00011384 32322F31302F323031-
2636 0001138D 375D202E2E2E00
2637                                    NextLine:
2638 00011394 0D0A00                        db 0Dh, 0Ah, 0
2639
2640                                    %include 'audio.s' ; 03/04/2017
   1                                <1> ; ****************************************************************************
   2                                <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - audio.s
   3                                <1> ; ----------------------------------------------------------------------------
   4                                <1> ; Last Update: 22/10/2017
   5                                <1> ; ----------------------------------------------------------------------------
   6                                <1> ; Beginning: 03/04/2017
   7                                <1> ; ----------------------------------------------------------------------------
   8                                <1> ; Assembler: NASM version 2.11 (trdos386.s)
   9                                <1> ; ****************************************************************************
  10                                <1>
  11                                <1> ; AUDIO CONTROLLER & CODEC DEFINITIONS & CODE FOR TRDOS 386
  12                                <1>
  13                                <1> ;============================================================================
  14                                <1> ;               EQUATES
  15                                <1> ;============================================================================
  16                                <1>
  17                                <1> ; PCI EQUATES
  18                                <1>
  19                                <1> BIT0  EQU 1
  20                                <1> BIT1  EQU 2
  21                                <1> BIT2  EQU 4
  22                                <1> BIT3  EQU 8
  23                                <1> BIT4  EQU 10h
  24                                <1> BIT5  EQU 20h
  25                                <1> BIT6  EQU 40h
  26                                <1> BIT7  EQU 80h
  27                                <1> BIT8  EQU 100h
  28                                <1> BIT9  EQU 200h
  29                                <1> BIT10 EQU 400h
  30                                <1> BIT11 EQU 800h
  31                                <1> BIT12 EQU 1000h
  32                                <1> BIT13 EQU 2000h
  33                                <1> BIT14 EQU 4000h
  34                                <1> BIT15 EQU 8000h
  35                                <1> BIT16 EQU 10000h
  36                                <1> BIT17 EQU 20000h
  37                                <1> BIT18 EQU 40000h
  38                                <1> BIT19 EQU 80000h
  39                                <1> BIT20 EQU 100000h
  40                                <1> BIT21 EQU 200000h
  41                                <1> BIT22 EQU 400000h
  42                                <1> BIT23 EQU 800000h
  43                                <1> BIT24 EQU 1000000h
  44                                <1> BIT25 EQU 2000000h
  45                                <1> BIT26 EQU 4000000h
  46                                <1> BIT27 EQU 8000000h
  47                                <1> BIT28 EQU 10000000h
  48                                <1> BIT29 EQU 20000000h
  49                                <1> BIT30 EQU 40000000h
  50                                <1> BIT31 EQU 80000000h
  51                                <1> NOT_BIT31 EQU 7FFFFFFFh
  52                                <1>
  53                                <1> ; PCI equates
  54                                <1> ; PCI function address (PFA)
  55                                <1> ; bit 31 = 1
  56                                <1> ; bit 23:16 = bus number     (0-255)
  57                                <1> ; bit 15:11 = device number  (0-31)
  58                                <1> ; bit 10:8 = function number (0-7)
  59                                <1> ; bit 7:0 = register number  (0-255)
  60                                <1>
  61                                <1> IO_ADDR_MASK    EQU     0FFFEh  ; mask off bit 0 for reading BARs
  62                                <1> PCI_INDEX_PORT  EQU     0CF8h
  63                                <1> PCI_DATA_PORT   EQU     0CFCh
  64                                <1> PCI32           EQU     BIT31   ; bitflag to signal 32bit access
  65                                <1> PCI16           EQU     BIT30   ; bitflag for 16bit access
  66                                <1> NOT_PCI32_PCI16    EQU   03FFFFFFh ; NOT BIT31+BIT30 ; 19/03/2017
  67                                <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
   103                  <1> VIA_ACLINK_CTRL_INIT      equ  (VIA_ACLINK_CTRL_ENABLE +
VIA_ACLINK_CTRL_RESET +                                 VIA_ACLINK_CTRL_PCM +                                   VIA_ACLINK_CTRL_VRA)
   104                  <1>
   105                  <1> CODEC_AUX_VOL             equ   04h
   106                  <1> VIA_REG_AC97_BUSY  equ   01000000h ;(1<<24)
   107                  <1> VIA_REG_AC97_CMD_SHIFT    equ   10h ; 16
   108                  <1> VIA_REG_AC97_PRIMARY_VALID equ 02000000h ;(1<<25)
   109                  <1> VIA_REG_AC97_READ  equ   00800000h ;(1<<23)
   110                  <1> VIA_REG_AC97_CODEC_ID_SHIFT   equ  1Eh ; 30
   111                  <1> VIA_REG_AC97_CODEC_ID_PRIMARY equ  0
   112                  <1> VIA_REG_AC97_DATA_SHIFT equ   0
   113                  <1> VIADEV_PLAYBACK         equ   0
   114                  <1> VIA_REG_OFFSET_STATUS   equ   0    ;; byte - channel status
   115                  <1> VIA_REG_OFFSET_CONTROL  equ   01h  ;; byte - channel control
   116                  <1> VIA_REG_CTRL_START equ   80h  ;; WO
   117                  <1> VIA_REG_CTRL_TERMINATE  equ   40h  ;; WO
   118                  <1> VIA_REG_CTRL_PAUSE      equ   08h  ;; RW
   119                  <1> VIA_REG_CTRL_RESET      equ   01h  ;; RW - probably reset? undocumented
   120                  <1> VIA_REG_OFFSET_STOP_IDX equ   08h  ;; dword - stop index, channel type, sample rate
   121                  <1> VIA8233_REG_TYPE_16BIT  equ   200000h ;; RW
   122                  <1> VIA8233_REG_TYPE_STEREO equ   100000h ;; RW
   123                  <1> VIA_REG_OFFSET_CURR_INDEX equ 0Fh ;; byte - channel current index (for via8233 only)
   124                  <1> VIA_REG_OFFSET_TABLE_PTR equ  04h  ;; dword - channel table pointer
   125                  <1> VIA_REG_OFFSET_CURR_PTR equ   04h  ;; dword - channel current pointer
   126                  <1> VIA_REG_OFS_PLAYBACK_VOLUME_L equ  02h ;; byte
   127                  <1> VIA_REG_OFS_PLAYBACK_VOLUME_R equ  03h ;; byte
   128                  <1> VIA_REG_CTRL_AUTOSTART    equ   20h
   129                  <1> VIA_REG_CTRL_INT_EOL      equ   02h
   130                  <1> VIA_REG_CTRL_INT_FLAG     equ   01h
   133                  <1> VIA_REG_CTRL_INT   equ  (VIA_REG_CTRL_INT_FLAG +
VIA_REG_CTRL_INT_EOL +                                  VIA_REG_CTRL_AUTOSTART)
   134                  <1>
   135                  <1> VIA_REG_STAT_STOPPED      equ   04h    ;; RWC
   136                  <1> VIA_REG_STAT_EOL   equ   02h    ;; RWC
   137                  <1> VIA_REG_STAT_FLAG  equ   01h    ;; RWC
   138                  <1> VIA_REG_STAT_ACTIVE       equ   80h    ;; RO
   139                  <1> ; 28/11/2016
   140                  <1> VIA_REG_STAT_LAST  equ   40h    ;; RO
   141                  <1> VIA_REG_STAT_TRIGGER_QUEUED equ 08h  ;; RO
   142                  <1> VIA_REF_CTRL_INT_STOP     equ   04h  ; Interrupt on Current Index = Stop Index
   143                  <1>                              ; and End of Block
   144                  <1>
   145                  <1> VIA_REG_OFFSET_CURR_COUNT equ 0Ch ;; dword - channel current count, index
   146                  <1>
   147                  <1> PORTB       EQU    061h
   148                  <1> REFRESH_STATUS     EQU    010h   ; Refresh signal status
   149                  <1>
   150                  <1> ; AC97 Codec registers.
   151                  <1>
   152                  <1> ; each codec/mixer register is 16bits
   153                  <1>
   154                  <1> CODEC_RESET_REG                 equ     00h  ; reset codec
   155                  <1> CODEC_MASTER_VOL_REG            equ     02h  ; master volume
   156                  <1> CODEC_HP_VOL_REG                equ     04h  ; headphone volume
   157                  <1> CODEC_MASTER_MONO_VOL_REG       equ     06h  ; master mono volume
   158                  <1> CODEC_MASTER_TONE_REG           equ     08h  ; master tone (R+L)
   159                  <1> CODEC_PCBEEP_VOL_REG            equ     0Ah  ; PC beep volume
   160                  <1> CODEC_PHONE_VOL_REG             equ     0Bh  ; phone volume
   161                  <1> CODEC_MIC_VOL_REG               equ     0Eh  ; MIC volume
   162                  <1> CODEC_LINE_IN_VOL_REG           equ     10h  ; line input volume
   163                  <1> CODEC_CD_VOL_REG                equ     12h  ; CD volume
   164                  <1> CODEC_VID_VOL_REG               equ     14h  ; video volume
   165                  <1> CODEC_AUX_VOL_REG               equ     16h  ; aux volume
   166                  <1> CODEC_PCM_OUT_REG               equ     18h  ; PCM output volume
   167                  <1> CODEC_RECORD_SELECT_REG         equ     1Ah  ; record select input
   168                  <1> CODEC_RECORD_VOL_REG            equ     1Ch  ; record volume
   169                  <1> CODEC_RECORD_MIC_VOL_REG        equ     1Eh  ; record mic volume
   170                  <1> CODEC_GP_REG                    equ     20h  ; general purpose
   171                  <1> CODEC_3D_CONTROL_REG            equ     22h  ; 3D control
   172                  <1> ; 24h is reserved
```

```
                        <1> CODEC_POWER_CTRL_REG            equ     26h  ; powerdown control
                        <1> CODEC_EXT_AUDIO_REG             equ     28h  ; extended audio
                        <1> CODEC_EXT_AUDIO_CTRL_REG        equ     2Ah  ; extended audio control
                        <1> CODEC_PCM_FRONT_DACRATE_REG     equ     2Ch  ; PCM out sample rate
                        <1> CODEC_PCM_SURND_DACRATE_REG     equ     2Eh  ; surround sound sample rate
                        <1> CODEC_PCM_LFE_DACRATE_REG       equ     30h  ; LFE sample rate
                        <1> CODEC_LR_ADCRATE_REG            equ     32h  ; PCM in sample rate
                        <1> CODEC_MIC_ADCRATE_REG           equ     34h  ; mic in sample rate
                        <1>
                        <1> ; VT8233 SGD bits (21/04/2017)
                        <1> FLAG  EQU BIT30
                        <1> EOL   EQU BIT31
                        <1>
                        <1> ; INTEL ICH EQUATES
                        <1> ; 28/05/2017
                        <1> INTEL_VID    equ    8086h  ; Intel's PCI vendor ID
                        <1> ICH_DID      equ    2415h  ; ICH (82801AA) device ID
                        <1> NAMBAR_REG      equ      10h    ; native audio mixer Base Address Register
                        <1> NABMBAR_REG     equ      14h    ; native audio bus mastering Base Addr Reg
                        <1>
                        <1> PI_CR_REG       equ     0Bh     ; PCM in Control Register
                        <1> PO_CR_REG    equ     1Bh     ; PCM out Control Register
                        <1> MC_CR_REG    equ     2Bh     ; MIC in Control Register
                        <1>
                        <1> PI_SR_REG    equ     6       ; PCM in Status register
                        <1> PO_SR_REG    equ     16h     ; PCM out Status register
                        <1> MC_SR_REG    equ     26h     ; MIC in Status register
                        <1>
                        <1> IOCE         equ     BIT4    ; interrupt on complete enable.
                        <1> FEIFE        equ     BIT3    ; set if you want an interrupt to fire
                        <1> LVBIE        equ     BIT2    ; last valid buffer interrupt enable.
                        <1> RR           equ     BIT1    ; reset registers.  Nukes all regs
                        <1>                              ; except bits 4:2 of this register.
                        <1>                              ; Only set this bit if BIT 0 is 0
                        <1> RPBM         equ     BIT0    ; Run/Pause
                        <1>                          ; set this bit to start the codec!
                        <1>
                        <1> PI_BDBAR_REG equ     0       ; PCM in buffer descriptor BAR
                        <1> PO_BDBAR_REG equ     10h     ; PCM out buffer descriptor BAR
                        <1> MC_BDBAR_REG equ     20h     ; MIC in buffer descriptor BAR
                        <1>
                        <1> PI_CIV_REG   equ     4       ; PCM in current Index value (RO)
                        <1> PO_CIV_REG   equ     14h     ; PCM out current Index value (RO)
                        <1> MC_CIV_REG   equ     24h     ; MIC in current Index value (RO)
                        <1>
                        <1> PI_LVI_REG   equ     5       ; PCM in Last Valid Index
                        <1> PO_LVI_REG   equ     15h     ; PCM out Last Valid Index
                        <1> MC_LVI_REG   equ     25h     ; MIC in Last Valid Index
                        <1>
                        <1> IOC          equ     BIT31; Fire an interrupt whenever this
                        <1>                          ; buffer is complete.
                        <1> BUP          equ     BIT30; Buffer Underrun Policy.
                        <1>
                        <1> GLOB_CNT_REG equ     2Ch     ; Global Control Register
                        <1> GLOB_STS_REG equ     30h     ; Global Status register (RO)
                        <1>
                        <1> CTRL_ST_CREADY     equ   BIT8+BIT9+BIT28 ; Primary Codec Ready
                        <1>
                        <1> CODEC_REG_POWERDOWN   equ 26h
                        <1> CODEC_REG_ST          equ 26h
                        <1>
                        <1> ; 22/06/2017
                        <1> PO_PICB_REG  equ 18h      ; PCM Out Position In Current Buffer Register
                        <1>
                        <1> ;============================================================================
                        <1> ;               CODE
                        <1> ;============================================================================
                        <1>
                        <1> ; CODE for INTEL ICH AC'97 AUDIO CONTROLLER
                        <1>
                        <1> DetectICH:
                        <1>       ; 10/06/2017
                        <1>       ; 05/06/2017
                        <1>       ; 29/05/2017
                        <1>       ; 28/05/2017
00011397 B886801524     <1>       mov     eax, (ICH_DID << 16) + INTEL_VID
0001139C E876000000     <1>         call    pciFindDevice
000113A1 730D           <1>         jnc     short d_ac97_1
                        <1> d_ac97_0:
                        <1> ; couldn't find the audio device!
000113A3 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
000113A4 B806115930     <1>       mov     eax, (VT8233_DID << 16) + VIA_VID
000113A9 E869000000     <1>         call    pciFindDevice
                        <1> ;       jnc     short d_vt8233_0
                        <1> ; couldn't find the audio device!
                        <1> ;     retn
000113AE 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>
000113B0 A3[8C650100]           <1>       mov   [audio_dev_id], eax
000113B5 8915[90650100]         <1>       mov   [audio_vendor], edx
                                <1>
                                <1>       ; init controller
000113BB B004                   <1>       mov   al, PCI_CMD_REG ; command register (04h)
000113BD E8E2000000             <1>       call  pciRegRead32
                                <1>
                                <1>       ; eax = BUS/DEV/FN/REG
                                <1>       ; edx = STATUS/COMMAND
                                <1>       ;     SSSSSSSSSSSSSSSSCCCCCCCCCCCCCCCC
000113C2 8915[94650100]         <1>       mov   [audio_stats_cmd], edx
                                <1>
000113C8 B010                   <1>       mov   al, PCI_IO_BASE ; IO base address register (10h)
                                <1>       ;mov  al, NAMBAR_REG      ; Native Audio Mixer BAR (10h)
000113CA E8D5000000             <1>       call  pciRegRead32
                                <1>
000113CF 66813D[90650100]86-    <1>       cmp   word [audio_vendor], 8086h ; AC'97 ?
000113D7 80                     <1>
000113D8 751F                   <1>       jne   short d_vt8233_1
                                <1>
000113DA 6683E2FE               <1>       and   dx, 0FFFEh ; Audio Codec IO_ADDR_MASK
000113DE 668915[BC650100]       <1>       mov   [NAMBAR], dx
                                <1>
000113E5 B014                   <1>       mov   al, NABMBAR_REG ; Native Audio Bus Mastering BAR (14h)
000113E7 E8B8000000             <1>       call  pciRegRead32
                                <1>
000113EC 6683E2C0               <1>       and   dx, 0FFC0h ; Audio Controller IO_ADDR_MASK
000113F0 668915[BE650100]       <1>       mov   [NABMBAR], dx
                                <1>         ;mov [audio_io_base], dx
                                <1>
000113F7 EB0B                   <1>       jmp   short d_ac97_2
                                <1>
                                <1> d_vt8233_1:
000113F9 6683E2C0               <1>       and     dx, 0FFC0h ; Audio Controller IO_ADDR_MASK
000113FD 668915[8A650100]       <1>         mov     [audio_io_base], dx
                                <1>
                                <1> d_ac97_2:
                                <1>       ; 10/06/2017
00011404 B03C                   <1>       mov   al, AC97_INT_LINE ; Interrupt Line Register (3Ch)
                                <1>       ;call pciRegRead32
00011406 E886000000             <1>       call  pciRegRead8
                                <1>
                                <1>       ;and  edx, 0FFh
0001140B 6681E2FF00             <1>       and   dx, 0FFh
                                <1>
00011410 8815[87650100]         <1>       mov     [audio_intr], dl
                                <1>
00011416 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
00011417 50             <1>       push   eax
                        <1>       ;push  esi
                        <1>       ;push  edi
                        <1>
00011418 89C6           <1>         mov     esi, eax                ; save off vend+device ID
0001141A BF00FFFF7F     <1>         mov     edi, (80000000h - 100h) ; start with bus 0, dev 0 func 0
                        <1>
                        <1> nextPCIdevice:
0001141F 81C700010000   <1>         add     edi, 100h
00011425 81FF00F8FF80   <1>         cmp     edi, 80FFF800h         ; scanned all devices?
0001142B F9             <1>         stc
0001142C 740C           <1>         je      short PCIScanExit       ; not found
                        <1>
0001142E 89F8           <1>         mov     eax, edi                ; read PCI registers
00011430 E86F000000     <1>         call    pciRegRead32
00011435 39F2           <1>         cmp     edx, esi                ; found device?
00011437 75E6           <1>         jne     short nextPCIdevice
00011439 F8             <1>         clc
                        <1>
                        <1> PCIScanExit:
0001143A 9C             <1>       pushf
0001143B B8FFFFFF7F     <1>       mov    eax, NOT_BIT31      ; 19/03/2017
00011440 21F8           <1>       and    eax, edi     ; return only bus/dev/fn #
00011442 9D             <1>       popf
                        <1>
                        <1>       ;pop   edi
                        <1>       ;pop   esi
00011443 5A             <1>       pop    edx
                        <1>       ;pop   ecx
00011444 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>
00011445 53             <1>       push   ebx
00011446 51             <1>       push   ecx
00011447 89C3           <1>         mov     ebx, eax        ; save eax, dh
00011449 88F1           <1>         mov     cl, dh
                        <1>
0001144B 25FFFFFF3F     <1>         and     eax, NOT_PCI32_PCI16   ; clear out data size request
00011450 0D00000080     <1>         or      eax, BIT31             ; make a PCI access request
00011455 24FC           <1>         and     al, ~3 ; NOT 3         ; force index to be dword
                        <1>
00011457 66BAF80C       <1>         mov     dx, PCI_INDEX_PORT
0001145B EF             <1>         out  dx, eax                   ; write PCI selector
                        <1>
0001145C 66BAFC0C       <1>         mov     dx, PCI_DATA_PORT
00011460 88D8           <1>         mov     al, bl
00011462 2403           <1>         and     al, 3                  ; figure out which port to
00011464 00C2           <1>         add     dl, al                 ; read to
                        <1>
00011466 F7C3000000C0   <1>       test    ebx, PCI32+PCI16
0001146C 7507           <1>         jnz     short _pregr0
0001146E EC             <1>       in     al, dx             ; return 8 bits of data
```

```
                                <1>         mov  dl, al
00011471 88CE                   <1>       mov     dh, cl                  ; restore dh for 8 bit read
00011473 EB12                   <1>       jmp    short _pregr2
                                <1> _pregr0:
00011475 F7C300000080           <1>       test    ebx, PCI32
0001147B 7507                   <1>         jnz  short _pregr1
0001147D 66ED                   <1>       in     ax, dx
0001147F 6689C2                 <1>         mov     dx, ax                ; return 16 bits of data
00011482 EB03                   <1>       jmp    short _pregr2
                                <1> _pregr1:
00011484 ED                     <1>       in     eax, dx                  ; return 32 bits of data
00011485 89C2                   <1>       mov    edx, eax
                                <1> _pregr2:
00011487 89D8                   <1>       mov     eax, ebx          ; restore eax
00011489 25FFFFFF3F             <1>         and     eax, NOT_PCI32_PCI16   ; clear out data size request
0001148E 59                     <1>       pop    ecx
0001148F 5B                     <1>       pop    ebx
00011490 C3                     <1>       retn
                                <1>
                                <1> pciRegRead8:
00011491 25FFFFFF3F             <1>         and     eax, NOT_PCI32_PCI16   ; set up 8 bit read size
00011496 EBAD                   <1>         jmp     short pciRegRead; call generic PCI access
                                <1>
                                <1> pciRegRead16:
00011498 25FFFFFF3F             <1>         and     eax, NOT_PCI32_PCI16   ; set up 16 bit read size
0001149D 0D00000040             <1>         or      eax, PCI16             ; call generic PCI access
000114A2 EBA1                   <1>         jmp     short pciRegRead
                                <1>
                                <1> pciRegRead32:
000114A4 25FFFFFF3F             <1>         and     eax, NOT_PCI32_PCI16   ; set up 32 bit read size
000114A9 0D00000080             <1>         or      eax, PCI32             ; call generic PCI access
000114AE 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>
000114B0 53                     <1>       push   ebx
000114B1 51                     <1>       push   ecx
000114B2 89C3                   <1>         mov     ebx, eax        ; save eax, edx
000114B4 89D1                   <1>         mov     ecx, edx
000114B6 25FFFFFF3F             <1>       and     eax, NOT_PCI32_PCI16     ; clear out data size request
000114BB 0D00000080             <1>         or      eax, BIT31             ; make a PCI access request
000114C0 24FC                   <1>         and     al, ~3 ; NOT 3         ; force index to be dword
                                <1>
000114C2 66BAF80C               <1>         mov     dx, PCI_INDEX_PORT
000114C6 EF                     <1>         out  dx, eax                   ; write PCI selector
                                <1>
000114C7 66BAFC0C               <1>         mov     dx, PCI_DATA_PORT
000114CB 88D8                   <1>         mov     al, bl
000114CD 2403                   <1>         and     al, 3                  ; figure out which port to
000114CF 00C2                   <1>         add     dl, al                 ; write to
                                <1>
000114D1 F7C3000000C0           <1>       test    ebx, PCI32+PCI16
000114D7 7505                   <1>         jnz     short _pregw0
000114D9 88C8                   <1>       mov    al, cl                    ; put data into al
000114DB EE                     <1>       out    dx, al
000114DC EB12                   <1>       jmp    short _pregw2
                                <1> _pregw0:
000114DE F7C300000080           <1>       test    ebx, PCI32
000114E4 7507                   <1>         jnz     short _pregw1
000114E6 6689C8                 <1>       mov    ax, cx              ; put data into ax
000114E9 66EF                   <1>       out    dx, ax
000114EB EB03                   <1>       jmp    short _pregw2
                                <1> _pregw1:
000114ED 89C8                   <1>       mov    eax, ecx            ; put data into eax
000114EF EF                     <1>       out    dx, eax
                                <1> _pregw2:
000114F0 89D8                   <1>         mov     eax, ebx        ; restore eax
000114F2 25FFFFFF3F             <1>         and     eax, NOT_PCI32_PCI16   ; clear out data size request
000114F7 89CA                   <1>         mov     edx, ecx        ; restore dx
000114F9 59                     <1>       pop    ecx
000114FA 5B                     <1>       pop    ebx
000114FB C3                     <1>       retn
                                <1>
                                <1> pciRegWrite8:
000114FC 25FFFFFF3F             <1>         and     eax, NOT_PCI32_PCI16   ; set up 8 bit write size
00011501 EBAD                   <1>         jmp  short pciRegWrite   ; call generic PCI access
                                <1>
                                <1> pciRegWrite16:
00011503 25FFFFFF3F             <1>         and     eax, NOT_PCI32_PCI16   ; set up 16 bit write size
00011508 0D00000040             <1>         or      eax, PCI16             ; call generic PCI access
0001150D EBA1                   <1>         jmp  short pciRegWrite
                                <1>
                                <1> pciRegWrite32:
0001150F 25FFFFFF3F             <1>         and     eax, NOT_PCI32_PCI16   ; set up 32 bit write size
00011514 0D00000080             <1>         or      eax, PCI32             ; call generic PCI access
00011519 EB95                   <1>         jmp  pciRegWrite
```

```
                                <1>
                                <1> init_codec:
                                <1>       ; 05/06/2017
                                <1>       ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)
                                <1>       ;
0001151B A1[8C650100]           <1>       mov    eax, [audio_dev_id]
00011520 B041                   <1>       mov    al, VIA_ACLINK_CTRL
00011522 E86AFFFFFF             <1>       call   pciRegRead8
                                <1>       ; ?
00011527 B040                   <1>       mov    al, VIA_ACLINK_STAT
00011529 E863FFFFFF             <1>       call   pciRegRead8
0001152E F6C201                 <1>       test   dl, VIA_ACLINK_C00_READY
00011531 7508                   <1>         jnz     short _codec_ready_1
00011533 E80E000000             <1>       call   reset_codec
00011538 7306                   <1>       jnc    short _codec_ready_2 ; eax = 1
0001153A C3                     <1>       retn
                                <1> _codec_ready_1:
0001153B B801000000             <1>       mov    eax, 1
                                <1> _codec_ready_2:
00011540 E87A000000             <1>       call   codec_io_w16
                                <1> detect_codec:
00011545 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)
00011546 A1[8C650100]           <1>       mov    eax, [audio_dev_id]
0001154B B041                   <1>       mov    al, VIA_ACLINK_CTRL
0001154D B2E0                   <1>              mov    dl, VIA_ACLINK_CTRL_ENABLE + VIA_ACLINK_CTRL_RESET + VIA_ACLINK_CTRL_SYNC
0001154F E8A8FFFFFF             <1>       call   pciRegWrite8
                                <1>
00011554 E83D000000             <1>       call   delay_100ms  ; wait 100 ms
                                <1> _rc_cold:
00011559 E808000000             <1>         call    cold_reset
0001155E 7301                   <1>         jnc     short _reset_codec_ok
                                <1>
                                <1>       ; 16/04/2017
                                <1>         ;xor     eax, eax         ; timeout error
                                <1>             ;stc
00011560 C3                     <1>       retn
                                <1>
                                <1> _reset_codec_ok:
00011561 31C0                   <1>         xor     eax, eax
                                <1>         ;mov al, VIA_ACLINK_C00_READY ; 1
00011563 FEC0                   <1>         inc  al
00011565 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
00011566 30D2                   <1>       xor    dl, dl ; 0
00011568 E88FFFFFFF             <1>       call   pciRegWrite8
                                <1>
0001156D 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
00011572 B2CC                   <1>       mov    dl, VIA_ACLINK_CTRL_INIT
00011574 E883FFFFFF             <1>       call   pciRegWrite8
                                <1>
00011579 B910000000             <1>       mov    ecx, 16      ; total 2s
                                <1>
                                <1> _crst_wait:
                                <1>       ;mov   eax, [audio_dev_id]
0001157E B040                   <1>       mov    al, VIA_ACLINK_STAT
00011580 E80CFFFFFF             <1>       call   pciRegRead8
                                <1>
00011585 F6C201                 <1>         test    dl, VIA_ACLINK_C00_READY
00011588 750B                   <1>         jnz     short _crst_ok
                                <1>
0001158A 51                     <1>       push   ecx
0001158B E806000000             <1>       call   delay_100ms
00011590 59                     <1>       pop    ecx
                                <1>
00011591 49                     <1>         dec     ecx
00011592 75EA                   <1>         jnz     short _crst_wait
                                <1>
                                <1> _crst_fail:
00011594 F9                     <1>         stc
                                <1> _crst_ok:
00011595 C3                     <1>       retn
                                <1>
                                <1> delay_100ms:
                                <1>       ; 29/05/2017
                                <1>       ; 24/03/2017 ('codec.asm')
                                <1>       ; wait 100 ms
00011596 B990010000             <1>       mov    ecx, 400  ; 400*0.25ms
                                <1> _delay_x_ms:
0001159B E803000000             <1>       call   delay1_4ms
000115A0 E2F9                   <1>         loop _delay_x_ms
000115A2 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:
000115A3 50             <1>         push    eax
000115A4 51             <1>         push    ecx
000115A5 B110           <1>         mov  cl, 16       ; close enough.
                        <1>
000115A7 E461           <1>       in    al, PORTB ; 61h
                        <1>
000115A9 2410           <1>       and   al, REFRESH_STATUS ; 10h
000115AB 88C5           <1>       mov   ch, al      ; Start toggle state
                        <1> _d4ms1:
000115AD E461           <1>       in    al, PORTB   ; Read system control port
                        <1>
000115AF 2410           <1>       and   al, REFRESH_STATUS ; Refresh toggles 15.085 microseconds
000115B1 38C5           <1>       cmp   ch, al
000115B3 74F8           <1>       je    short _d4ms1 ; Wait for state change
                        <1>
000115B5 88C5           <1>       mov   ch, al      ; Update with new state
000115B7 FEC9           <1>       dec   cl
000115B9 75F2           <1>       jnz   short _d4ms1
                        <1>
000115BB F8             <1>       clc   ; 29/05/2017
                        <1>
000115BC 59             <1>         pop     ecx
000115BD 58             <1>         pop     eax
000115BE C3             <1>         retn
                        <1>
                        <1> ; 10/04/2017 (TRDOS 386)
                        <1> ; 12/11/2016
                        <1>
                        <1> codec_io_w16: ;w32
                        <1>       ; ('codec.asm')
000115BF 668B15[8A650100]   <1>         mov  dx, [audio_io_base]
000115C6 6681C28000         <1>         add     dx, VIA_REG_AC97
000115CB EF             <1>       out   dx, eax
000115CC C3             <1>         retn
                        <1>
                        <1> codec_io_r16: ;r32
                        <1>       ; ('codec.asm')
000115CD 668B15[8A650100]   <1>         mov     dx, [audio_io_base]
000115D4 6681C28000         <1>         add     dx, VIA_REG_AC97
000115D9 ED             <1>         in  eax, dx
000115DA C3             <1>         retn
                        <1>
                        <1> ctrl_io_w8:
                        <1>       ; ('codec.asm')
000115DB 660315[8A650100]   <1>         add     dx, [audio_io_base]
000115E2 EE             <1>         out  dx, al
000115E3 C3             <1>         retn
                        <1>
                        <1> ctrl_io_r8:
                        <1>       ; ('codec.asm')
000115E4 660315[8A650100]   <1>         add     dx, [audio_io_base]
000115EB EC             <1>         in  al, dx
000115EC C3             <1>         retn
                        <1>
                        <1> ctrl_io_w32:
                        <1>       ; ('codec.asm')
000115ED 660315[8A650100]   <1>         add     dx, [audio_io_base]
000115F4 EF             <1>         out  dx, eax
000115F5 C3             <1>         retn
                        <1>
                        <1> ctrl_io_r32:
                        <1>       ; ('codec.asm')
000115F6 660315[8A650100]   <1>         add  dx, [audio_io_base]
000115FD ED             <1>       in    eax, dx
000115FE 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
000115FF C1E010         <1>         shl     eax, VIA_REG_AC97_CMD_SHIFT
00011602 0D00008002         <1>         or      eax, VIA_REG_AC97_PRIMARY_VALID + VIA_REG_AC97_READ
                        <1>
00011607 E8B3FFFFFF         <1>       call    codec_io_w16
                        <1>
                        <1>             ; codec_valid
0001160C E831000000         <1>       call  codec_check_ready
00011611 7301           <1>         jnc  short _cr_ok
                        <1>
00011613 C3             <1>       retn
                        <1>
                        <1> _cr_ok:
                        <1>       ; wait 25 ms
00011614 B950000000         <1>       mov   ecx, 80 ; (100*0.25 ms)
                        <1> _cr_wloop:
00011619 E885FFFFFF         <1>       call  delay1_4ms
0001161E E2F9           <1>       loop  _cr_wloop
```

```
                                <1>
00011620 E8A8FFFFFF             <1>         call    codec_io_r16
00011625 25FFFF0000             <1>         and     eax, 0FFFFh
0001162A 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>
0001162B C1E210                 <1>       shl     edx, VIA_REG_AC97_CMD_SHIFT
                                <1>
0001162E C1E000                 <1>         shl     eax, VIA_REG_AC97_DATA_SHIFT ; shl eax, 0
00011631 09C2                   <1>         or      edx, eax
                                <1>
00011633 B800000000             <1>         mov     eax, VIA_REG_AC97_CODEC_ID_PRIMARY
00011638 C1E01E                 <1>         shl     eax, VIA_REG_AC97_CODEC_ID_SHIFT
0001163B 09D0                   <1>         or      eax, edx
                                <1>
0001163D 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:
00011642 B914000000             <1>       mov    ecx, 20      ; total 2s
                                <1> _ccr_wait:
00011647 51                     <1>       push   ecx
                                <1>
00011648 E880FFFFFF             <1>         call    codec_io_r16
0001164D A900000001             <1>         test    eax, VIA_REG_AC97_BUSY
00011652 740B                   <1>         jz      short _ccr_ok
                                <1>
00011654 E83DFFFFFF             <1>       call   delay_100ms
                                <1>
00011659 59                     <1>       pop    ecx
                                <1>
0001165A 49                     <1>       dec     ecx
0001165B 75EA                   <1>         jnz     short _ccr_wait
                                <1>
0001165D F9                     <1>         stc
0001165E C3                     <1>         retn
                                <1>
                                <1> _ccr_ok:
0001165F 59                     <1>       pop    ecx
00011660 25FFFF0000             <1>       and     eax, 0FFFFh
00011665 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>
00011666 B802020000             <1>       mov     eax, 0202h
0001166B 66A3[BA650100]         <1>       mov    [audio_master_volume], ax
00011671 66B81F1F               <1>       mov    ax, 1F1Fh ; 31,31
00011675 BA02000000             <1>       mov    edx, CODEC_MASTER_VOL_REG ; 02h ; Line Out
0001167A E8ACFFFFFF             <1>       call   codec_write
                                <1>       ;jc    short cconfig_error
                                <1>
                                <1>       ;mov    eax, 0202h
0001167F 66B80202               <1>       mov     ax, 0202h
00011683 BA18000000             <1>       mov    edx, CODEC_PCM_OUT_REG ; 18h ; Wave Output (Stereo)
00011688 E89EFFFFFF             <1>       call   codec_write
                                <1>       ;jc    short cconfig_error
                                <1>
                                <1>       ;mov    eax, 0202h
0001168D 66B80202               <1>       mov    ax, 0202h
00011691 BA04000000             <1>       mov    edx, CODEC_AUX_VOL ; 04h ; CODEC_HP_VOL_REG ; HeadPhone
00011696 E890FFFFFF             <1>       call   codec_write
                                <1>       ;jc    short cconfig_error
                                <1>
                                <1>       ;mov    eax, 08h
                                <1>         ;mov    ax,  08h
0001169B 66B80880               <1>       mov    ax, 8008h ; Mute
0001169F BA0C000000             <1>       mov    edx, 0Ch  ; AC97_PHONE_VOL ; TAD Input (Mono)
000116A4 E882FFFFFF             <1>       call   codec_write
                                <1>       ;jc    short cconfig_error
                                <1>
                                <1>       ;mov    eax, 0808h
000116A9 66B80808               <1>         mov     ax,  0808h
000116AD BA10000000             <1>         mov edx, CODEC_LINE_IN_VOL_REG ; 10h ; Line Input (Stereo)
000116B2 E874FFFFFF             <1>       call   codec_write
                                <1>       ;jc    short cconfig_error
                                <1>
                                <1>       ;mov    eax, 0808h
000116B7 66B80808               <1>       mov     ax,  0808h
000116BB BA12000000             <1>         mov edx, CODEC_CD_VOL_REG ; 12h ; CR Input (Stereo)
000116C0 E866FFFFFF             <1>       call   codec_write
```

```
                        <1>       ;jc    short cconfig_error
                        <1>
                        <1>       ;mov    eax, 0808h
000116C5 66B80808       <1>       mov     ax,  0808h
000116C9 BA16000000     <1>         mov  edx, CODEC_AUX_VOL_REG ; 16h ; Aux Input (Stereo)
                        <1>       ;call  codec_write
                        <1>       ;;jc   short cconfig_error
000116CE E958FFFFFF     <1>       jmp    codec_write ; 10/06/2017
                        <1>
                        <1> ;     ; Extended Audio Status (2Ah)
                        <1> ;     mov    eax, CODEC_EXT_AUDIO_CTRL_REG ; 2Ah
                        <1> ;     call   codec_read
                        <1> ;     and     eax, 0FFFFh - 2         ; clear DRA (BIT1)
                        <1> ;     ;or     eax, 1                  ; set VRA (BIT0)
                        <1> ;     or     eax, 5       ; VRA (BIT0) & S/PDIF (BIT2) ; 14/11/2016
                        <1> ;     mov    edx, CODEC_EXT_AUDIO_CTRL_REG
                        <1> ;     call   codec_write
                        <1> ;     ;jc    short cconfig_error
                        <1> ;
                        <1> ;set_sample_rate:
                        <1> ;     ;movzx eax, word [audio_freq]
                        <1> ;     mov    ax, [audio_freq]
                        <1> ;     mov    edx, CODEC_PCM_FRONT_DACRATE_REG ; 2Ch ; PCM Front DAC Rate
                        <1> ;     ;call  codec_write
                        <1> ;     ;retn
                        <1> ;     jmp    codec_write
                        <1>
                        <1> ;cconfig_error:
                        <1> ;     retn
                        <1>
                        <1> vt8233_int_handler:
                        <1>       ; Interrupt Handler for VIA VT8237R Audio Controller
                        <1>       ; Note: called by 'dev_IRQ_service'
                        <1>       ; 14/10/2017
                        <1>       ; 09/10/2017, 10/10/2017, 12/10/2017
                        <1>       ; 13/06/2017
                        <1>       ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')
                        <1>       ; 24/03/2017 - 'PLAYER.COM' ('player.asm')
                        <1>
                        <1>       ;push  eax ; * must be saved !
                        <1>       ;push  edx
                        <1>       ;push  ecx
                        <1>       ;push  ebx ; * must be saved !
                        <1>       ;push  esi
                        <1>       ;push  edi
                        <1>
                        <1>       ;cmp   byte [audio_busy], 1
                        <1>       ;jnb   short _ih0 ; 09/10/2017
                        <1>
                        <1>       ;mov   byte [audio_flag_eol], 0
                        <1>
000116D3 66BA0000       <1>         mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STATUS
000116D7 E808FFFFFF     <1>         call    ctrl_io_r8
                        <1>
000116DC A880           <1>       test    al, VIA_REG_STAT_ACTIVE
000116DE 7417           <1>         jz      short _ih0 ; 09/10/2017
                        <1>
000116E0 2407           <1>         and     al, VIA_REG_STAT_EOL + VIA_REG_STAT_FLAG + VIA_REG_STAT_STOPPED
000116E2 A2[B9650100]   <1>       mov    [audio_flag_eol], al
000116E7 740E           <1>         jz   short _ih0 ; 09/10/2017
                        <1>
                        <1>       ; 09/10/2017
                        <1>       ;mov   byte [audio_busy], 1
                        <1>
000116E9 803D[B8650100]01 <1>     cmp    byte [audio_play_cmd], 1
000116F0 7315           <1>       jnb    short _ih1 ; 10/10/2017
                        <1>
000116F2 E860000000     <1>       call   channel_reset
                        <1> _ih0:
                        <1>       ; 09/10/2017
000116F7 A0[B9650100]   <1>         mov     al, [audio_flag_eol]   ;; ack ;;
000116FC 66BA0000       <1>         mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STATUS
00011700 E8D6FEFFFF     <1>         call    ctrl_io_w8
00011705 EB4F           <1>       jmp    short _ih4
                        <1> _ih1:
                        <1> vt8233_tuneLoop:
00011707 A0[B9650100]   <1>         mov     al, [audio_flag_eol]   ;; ack ;;
0001170C 66BA0000       <1>         mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STATUS
00011710 E8C6FEFFFF     <1>         call    ctrl_io_w8
                        <1>
                        <1>       ; 12/10/2017
00011715 C605[AC650100]00 <1>     mov    byte [audio_flag], 0 ; Reset
                        <1>
                        <1>       ; 10/10/2017
                        <1>       ; 09/10/2017
                        <1>       ;test  byte [audio_flag_eol], VIA_REG_STAT_FLAG
                        <1>       ;jz    short _ih2 ; EOL
                        <1>
                        <1>       ; 14/10/2017
0001171C F605[B9650100]02 <1>     test   byte [audio_flag_eol], VIA_REG_STAT_EOL
00011723 7506           <1>       jnz    short _ih2 ; EOL
                        <1>                       ; (Half Buffer 2 has been completed
                        <1>                       ; and Half Buffer 1 will be played.)
                        <1>       ; FLAG
                        <1>       ; (Half Buffer 1 has been completed
                        <1>       ;  and Half Buffer 2 will be played.)
                        <1>
                        <1>       ; 14/10/2017
                        <1>       ;; (Continue to play.)
                        <1>       ;mov   al, VIA_REG_CTRL_INT
                        <1>             ;or    al, VIA_REG_CTRL_START
                        <1>             ;mov   dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_CONTROL
```

```
 988                            <1>         ;call      ctrl_io_w8
 989                            <1>       ; 12/10/2017
 990                            <1>       ;mov   byte [audio_flag], 1
 991 00011725 FE05[AC650100]    <1>       inc    byte [audio_flag] ; = 1
 992                            <1> _ih2:
 993                            <1>       ; 10/10/2017
 994 0001172B 8B3D[A4650100]    <1>       mov    edi, [audio_dma_buff]
 995 00011731 8B0D[A8650100]    <1>       mov    ecx, [audio_dmabuff_size]
 996 00011737 D1E9              <1>       shr    ecx, 1 ; dma buff size / 2 = half buffer size
 997                            <1>
 998                            <1>       ; 12/10/2017
 999 00011739 803D[AC650100]00  <1>       cmp    byte [audio_flag], 0
1000 00011740 7702              <1>       ja     short _ih3 ; Playing Half Buffer 2 (Current: FLAG)
1001                            <1>       ; Playing Half Buffer 1 (Current: EOL)
1002 00011742 01CF              <1>       add    edi, ecx
1003                            <1> _ih3:
1004                            <1>       ; Update half buffer 2 while playing half buffer 1 (FLAG)
1005                            <1>       ; Update half buffer 1 while playing half buffer 2 (EOL)
1006                            <1>
1007 00011744 8B35[9C650100]    <1>       mov    esi, [audio_p_buffer] ; phy addr of audio buff
1008 0001174A C1E902            <1>       shr    ecx, 2 ; half buff size / 4
1009 0001174D F3A5              <1>       rep    movsd
1010                            <1>       ; switch flag value ;
1011 0001174F 8035[AC650100]01  <1>       xor    byte [audio_flag], 1 ; 10/10/2017
1012                            <1>       ; 12/10/2017
1013                            <1>       ; [audio_flag] = 0 : Playing dma half buffer 2 (just after FLAG)
1014                            <1>                          ; Next buffer (to update) is dma half buff 1
1015                            <1>       ;             = 1 : Playing dma half buffer 1 (just after EOL)
1016                            <1>                          ; Next buffer (to update) is dma half buff 2
1017                            <1> _ih4:
1018                            <1>       ; 28/05/2017
1019                            <1>       ;mov   byte [audio_busy], 0 ; 09/10/2017
1020                            <1>       ;
1021                            <1>       ;pop   edi
1022                            <1>       ;pop   esi
1023                            <1>       ;pop   ebx ; * must be restored !
1024                            <1>       ;pop   ecx
1025                            <1>       ;pop   edx
1026                            <1>       ;pop   eax ; * must be restored !
1027                            <1>
1028 00011756 C3                <1>       retn
1029                            <1>
1030                            <1> channel_reset:
1031                            <1>       ; 24/06/2017
1032                            <1>       ; 29/05/2017
1033                            <1>       ; 23/03/2017
1034                            <1>       ; 14/11/2016 - Erdogan Tan
1035                            <1>       ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)
1036 00011757 BA01000000      <1>         mov  edx, VIA_REG_OFFSET_CONTROL
1037                            <1>         ;mov eax, VIA_REG_CTRL_PAUSE + VIA_REG_CTRL_TERMINATE + VIA_REG_CTRL_RESET
1038 0001175C B848000000      <1>         mov  eax, VIA_REG_CTRL_PAUSE + VIA_REG_CTRL_TERMINATE ; 24/06/2017
1039 00011761 E875FEFFFF      <1>       call    ctrl_io_w8
1040                            <1>
1041                            <1>         ;mov edx, VIA_REG_OFFSET_CONTROL
1042                            <1>         ;call   ctrl_io_r8
1043                            <1>
1044                            <1>       ; wait for 50 ms
1045 00011766 B9A0000000        <1>       mov    ecx, 160 ; (200*0.25 ms) ; 29/05/2017
1046                            <1> _ch_rst_wait:
1047 0001176B E833FEFFFF        <1>       call   delay1_4ms
1048 00011770 49                <1>       dec    ecx
1049 00011771 75F8              <1>       jnz    short _ch_rst_wait
1050                            <1>
1051                            <1>         ; disable interrupts
1052 00011773 BA01000000        <1>         mov  edx, VIA_REG_OFFSET_CONTROL
1053 00011778 31C0              <1>         xor     eax, eax
1054 0001177A E85CFEFFFF        <1>         call    ctrl_io_w8
1055                            <1>
1056                            <1>         ; clear interrupts
1057 0001177F BA00000000        <1>         mov  edx, VIA_REG_OFFSET_STATUS
1058 00011784 B803000000        <1>       mov    eax, 3
1059 00011789 E84DFEFFFF        <1>         call ctrl_io_w8
1060                            <1>
1061                            <1>       ;mov   edx, VIA_REG_OFFSET_CURR_PTR
1062                            <1>       ;xor   eax, eax
1063                            <1>       ;call  ctrl_io_w32
1064                            <1>
1065 0001178E C3                <1>         retn
1066                            <1>
1067                            <1> vt8233_stop: ; 22/04/2017
1068 0001178F C605[B8650100]00  <1>       mov    byte [audio_play_cmd], 0 ; stop !
1069                            <1> _tlp2:
1070                            <1>       ; 24/06/2017
1071                            <1>         ; finished with song, stop everything
1072                            <1>       ;mov   al, VIA_REG_CTRL_INT
1073                            <1>         ;or  al, VIA_REG_CTRL_TERMINATE
1074                            <1>       ;mov   dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_CONTROL
1075                            <1>         ;call      ctrl_io_w8
1076                            <1>
1077                            <1>         ;call      channel_reset
1078                            <1>       ;retn
1079 00011796 EBBF              <1>       jmp    short channel_reset
1080                            <1>
1081                            <1> set_vt8233_bdl: ; Set VT8237R Buffer Descriptor List
1082                            <1>       ; 28/05/2017
1083                            <1>       ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')
1084                            <1>       ; 24/03/2017 - 'PLAYER.COM' ('via_wav.asm' - 29/11/2016)
1085                            <1>
1086                            <1>       ; eax = dma buffer address = [audio_DMA_buff]
1087                            <1>       ; ecx = dma buffer buffer size = [audio_dmabuff_size]
1088                            <1>
1089 00011798 D1E9              <1>       shr    ecx, 1 ; dma half buffer size
```

```
1090 0001179A 89CE              <1>       mov    esi, ecx
1091                            <1>
1092 0001179C BF[C0650100]      <1>         mov     edi, audio_bdl_buff    ; get BDL address
1093 000117A1 B910000000        <1>         mov     ecx, 32 / 2            ; make 32 entries in BDL
1094                            <1>
1095 000117A6 EB05              <1>       jmp    short s_vt8233_bdl1
1096                            <1>
1097                            <1> s_vt8233_bdl0:
1098                            <1>       ; set buffer descriptor 0 to start of data file in memory
1099                            <1>
1100 000117A8 A1[A4650100]      <1>       mov    eax, [audio_dma_buff]     ; Physical address of DMA buffer
1101                            <1>
1102                            <1> s_vt8233_bdl1:
1103 000117AD AB                <1>       stosd                    ; store dmabuffer1 address
1104                            <1>
1105 000117AE 89C2              <1>       mov    edx, eax
1106                            <1>
1107                            <1> ; VIA VT8235.PDF: (Page 110) (Erdogan Tan, 29/11/2016)
1108                            <1>       ;
1109                            <1>       ;      Audio SGD Table Format
1110                            <1>       ;      -------------------------------
1111                            <1>       ;      63   62    61-56    55-32  31-0
1112                            <1>       ;      --   --   --------  -----  ----
1113                            <1>       ;      EOL FLAG -reserved- Base   Base
1114                            <1>       ;                         Count  Address
1115                            <1>       ;                           [23:0] [31:0]
1116                            <1>       ;      EOL: End Of Link.
1117                            <1>       ;           1 indicates this block is the last of the link.
1118                            <1>       ;           If the channel “Interrupt on EOL” bit is set, then
1119                            <1>       ;           an interrupt is generated at the end of the transfer.
1120                            <1>       ;
1121                            <1>       ;      FLAG: Block Flag. If set, transfer pauses at the end of this
1122                            <1>       ;            block. If the channel “Interrupt on FLAG” bit is set,
1123                            <1>       ;            then an interrupt is generated at the end of this block.
1124                            <1>
1125 000117B0 89F0              <1>       mov    eax, esi ; DMA half buffer size
1126 000117B2 01C2              <1>       add    edx, eax
1127 000117B4 0D00000040        <1>       or     eax, FLAG
1128                            <1>       ;or    eax, EOL
1129 000117B9 AB                <1>       stosd
1130                            <1>
1131                            <1> ; 2nd buffer:
1132                            <1>
1133 000117BA 89D0              <1>         mov  eax, edx ; Physical address of the 2nd half of DMA buffer
1134 000117BC AB                <1>       stosd         ; store dmabuffer2 address
1135                            <1>
1136                            <1> ; set length to [audio_dmabuff_size]/2
1137                            <1> ; Set control (bits 31:16) to BUP, bits 15:0=number of samples
1138                            <1> ;
1139 000117BD 89F0              <1>       mov    eax, esi ; DMA half buffer size
1140 000117BF 0D00000080        <1>       or     eax, EOL
1141                            <1>       ;or    eax, FLAG
1142 000117C4 AB                <1>       stosd
1143                            <1>
1144 000117C5 E2E1              <1>       loop    s_vt8233_bdl0
1145                            <1>
1146 000117C7 C3                <1>       retn
1147                            <1>
1148                            <1> vt8233_start_play:
1149                            <1>       ; start to play audio data via VT8233 audio controller
1150                            <1>       ; 13/06/2017
1151                            <1>       ; 10/06/2017
1152                            <1>       ; 24/04/2017
1153                            <1>       ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')
1154                            <1>       ; 24/03/2017 - 'PLAYER.COM' ('via_wav.asm' - 29/11/2016)
1155                            <1>       ; write buffer descriptor list address
1156                            <1>       ;
1157                            <1>
1158                            <1>       ; Extended Audio Status (2Ah)
1159 000117C8 B82A000000        <1>       mov    eax, CODEC_EXT_AUDIO_CTRL_REG ; 2Ah
1160 000117CD E82DFEFFFF        <1>       call   codec_read
1161 000117D2 25FDFF0000        <1>       and     eax, 0FFFFh - 2          ; clear DRA (BIT1)
1162                            <1>       ;or     eax, 1                   ; set VRA (BIT0)
1163 000117D7 83C805            <1>       or     eax, 5       ; VRA (BIT0) & S/PDIF (BIT2) ; 14/11/2016
1164 000117DA BA2A000000        <1>       mov    edx, CODEC_EXT_AUDIO_CTRL_REG
1165 000117DF E847FEFFFF        <1>       call   codec_write
1166                            <1>       ;jc    short cconfig_error
1167                            <1>
1168                            <1> set_sample_rate:
1169                            <1>       ;movzx eax, word [audio_freq]
1170 000117E4 66A1[B6650100]    <1>       mov    ax, [audio_freq]
1171 000117EA BA2C000000        <1>       mov    edx, CODEC_PCM_FRONT_DACRATE_REG ; 2Ch ; PCM Front DAC Rate
1172 000117EF E837FEFFFF        <1>       call   codec_write
1173                            <1>
1174 000117F4 B8[C0650100]      <1>         mov  eax, audio_bdl_buff
1175                            <1>
1176                            <1>       ; 12/11/2016 - Erdogan Tan
1177                            <1>       ; (Ref: KolibriOS, vt823x.asm, 'create_primary_buff')
1178 000117F9 BA04000000        <1>       mov    edx, VIADEV_PLAYBACK + VIA_REG_OFFSET_TABLE_PTR
1179 000117FE E8EAFDFFFF        <1>         call ctrl_io_w32
1180                            <1>
1181                            <1>       ;call  codec_check_ready
1182                            <1>
1183 00011803 66BA0200          <1>       mov    dx, VIADEV_PLAYBACK + VIA_REG_OFS_PLAYBACK_VOLUME_L
1184                            <1>         ;mov eax, 2 ; 31
1185 00011807 B01F              <1>       mov    al, 31
1186 00011809 2A05[BA650100]    <1>         sub  al, [audio_master_volume_l]
1187 0001180F E8C7FDFFFF        <1>       call   ctrl_io_w8
1188                            <1>
1189                            <1>       ;call  codec_check_ready
1190                            <1>
1191 00011814 66BA0300          <1>         mov     dx, VIADEV_PLAYBACK + VIA_REG_OFS_PLAYBACK_VOLUME_R
```

```
1192                                    <1>         ;mov ax, 2  ; 31
1193 00011818 B01F                      <1>       mov   al, 31
1194 0001181A 2A05[BB650100]            <1>         sub  al, [audio_master_volume_r]
1195 00011820 E8B6FDFFFF                <1>       call    ctrl_io_w8
1196                                    <1>
1197                                    <1>       ;call  codec_check_ready
1198                                    <1> ;
1199                                    <1> ;
1200                                    <1> ; All set.  Let's play some music.
1201                                    <1> ;
1202                                    <1> ;
1203                                    <1>             ;mov    dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STOP_IDX
1204                                    <1>         ;mov    ax, VIA8233_REG_TYPE_16BIT or VIA8233_REG_TYPE_STEREO or 0xfffff or 0xff000000
1205                                    <1>         ;call   ctrl_io_w32
1206                                    <1>
1207                                    <1>       ;call  codec_check_ready
1208                                    <1>
1209                                    <1>       ; 08/12/2016
1210                                    <1>       ; 07/10/2016
1211                                    <1>         ;mov    al, 1
1212 00011825 B01F                     <1>         mov  al, 31
1213 00011827 E815000000               <1>       call    set_VT8233_LastValidIndex
1214                                    <1>
1215 0001182C C605[B8650100]01         <1>       mov   byte [audio_play_cmd], 1 ; play command (do not stop) !
1216                                    <1>
1217                                    <1> vt8233_play: ; continue to play
1218                                    <1>       ; 22/04/2017
1219 00011833 B023                     <1>         mov  al, VIA_REG_CTRL_INT
1220 00011835 0C80                     <1>             or     al, VIA_REG_CTRL_START
1221                                    <1>         ;mov al, VIA_REG_CTRL_AUTOSTART + VIA_REG_CTRL_START
1222                                    <1>       ;mov   al, VIA_REG_CTRL_AUTOSTART + VIA_REG_CTRL_START + VIA_REG_CTRL_INT_FLAG
1223 00011837 66BA0100                 <1>       mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_CONTROL
1224 0001183B E89BFDFFFF               <1>         call    ctrl_io_w8
1225                                    <1>       ;call  codec_check_ready
1226                                    <1>       ;retn
1227                                    <1>       ;jmp   codec_check_ready
1228 00011840 C3                       <1>       retn
1229                                    <1>
1230                                    <1> ;input AL = index # to stop on
1231                                    <1> set_VT8233_LastValidIndex:
1232                                    <1>       ; 10/06/2017
1233                                    <1>       ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')
1234                                    <1>       ; 24/03/2017 - 'PLAYER.COM' ('via_wav.asm' - 29/11/2016)
1235                                    <1>       ; 19/11/2016
1236                                    <1>       ; 14/11/2016 - Erdogan Tan (Ref: VIA VT8235.PDF, Page 110)
1237                                    <1>       ; 12/11/2016 - Erdogan Tan
1238                                    <1>       ; (Ref: KolibriOS, vt823x.asm, 'create_primary_buff')
1239                                    <1>       ;push  edx
1240 00011841 6650                     <1>       push   ax
1241                                    <1>       ;push  ecx
1242 00011843 0FB705[B6650100]         <1>       movzx  eax, word [audio_freq] ; Hertz
1243 0001184A BA00001000               <1>       mov    edx, 100000h ; 2^20 = 1048576
1244 0001184F F7E2                     <1>       mul    edx
1245 00011851 B980BB0000               <1>       mov    ecx, 48000
1246 00011856 F7F1                     <1>       div    ecx
1247                                    <1>       ;and   eax, 0FFFFFh
1248                                    <1>       ;pop   ecx
1249 00011858 665A                     <1>       pop    dx
1250 0001185A C1E218                   <1>       shl    edx, 24  ; STOP Index Setting: Bit 24 to 31
1251 0001185D 09D0                     <1>       or     eax, edx
1252                                    <1>       ; 19/11/2016
1253 0001185F 803D[B4650100]10         <1>       cmp    byte [audio_bps], 16
1254 00011866 7505                     <1>       jne    short sLVI_1
1255 00011868 0D00002000               <1>       or     eax, VIA8233_REG_TYPE_16BIT
1256                                    <1> sLVI_1:
1257 0001186D 803D[B5650100]02         <1>       cmp    byte [audio_stmo], 2
1258 00011874 7505                     <1>       jne    short sLVI_2
1259 00011876 0D00001000               <1>       or     eax, VIA8233_REG_TYPE_STEREO
1260                                    <1> sLVI_2:
1261 0001187B BA08000000               <1>       mov     edx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STOP_IDX
1262 00011880 E868FDFFFF               <1>         call    ctrl_io_w32
1263                                    <1>       ;call  codec_check_ready
1264                                    <1>       ;pop   edx
1265 00011885 C3                       <1>       retn
1266                                    <1>
1267                                    <1> vt8233_pause: ; pause
1268                                    <1>       ; 10/06/2017
1269                                    <1>       ; 22/04/2017
1270 00011886 B023                     <1>       mov     al, VIA_REG_CTRL_INT
1271 00011888 0C08                     <1>         or      al, VIA_REG_CTRL_PAUSE
1272 0001188A 66BA0100                 <1>       mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_CONTROL
1273 0001188E E848FDFFFF               <1>         call    ctrl_io_w8
1274                                    <1>       ;call  codec_check_ready
1275                                    <1>       ;retn
1276                                    <1>       ;jmp   codec_check_ready
1277 00011893 C3                       <1>       retn
1278                                    <1>
1279                                    <1> vt8233_reset:
1280                                    <1>       ; 22/04/2017
1281                                    <1>       ; reset VT8237R (vt8233) Audio Controller
1282                                    <1>       ;cmp   byte [audio_play_cmd], 1
1283                                    <1>       ;jna   short vt8233_rst_0
1284 00011894 C605[B8650100]00         <1>       mov    byte [audio_play_cmd], 0 ; stop !
1285                                    <1> vt8233_rst_0:
1286 0001189B E8A6FCFFFF               <1>       call   reset_codec
1287 000118A0 720A                     <1>       jc     short vt8233_rst_1 ; codec error !
1288                                    <1>       ; eax = 1
1289 000118A2 E818FDFFFF               <1>       call   codec_io_w16 ; w32
1290 000118A7 E8ABFEFFFF               <1>       call   channel_reset
1291                                    <1> vt8233_rst_1:
1292 000118AC C3                       <1>       retn
1293                                    <1>
```

```
1294                                    <1> vt8233_volume:
1295                                    <1>       ; set VT8237R (vt8233) sound volume level
1296                                    <1>       ; 24/04/2017
1297                                    <1>       ; 22/04/2017
1298                                    <1>       ; bl = component (0 = master/playback/lineout volume)
1299                                    <1>       ; cl = left channel volume level (0 to 31)
1300                                    <1>       ; ch = right channel volume level (0 to 31)
1301                                    <1>
1302 000118AD 08DB                      <1>       or    bl, bl
1303 000118AF 7520                      <1>       jnz   short vt8233_vol_1 ; temporary !
1304 000118B1 66B81F1F                  <1>       mov   ax, 1F1Fh ; 31,31
1305 000118B5 38C1                      <1>       cmp   cl, al
1306 000118B7 7718                      <1>       ja    short vt8233_vol_1 ; temporary !
1307 000118B9 38E5                      <1>       cmp   ch, ah
1308 000118BB 7714                      <1>       ja    short vt8233_vol_1 ; temporary !
1309 000118BD 66890D[BA650100]          <1>       mov   [audio_master_volume], cx
1310 000118C4 6629C8                    <1>       sub   ax, cx
1311 000118C7 BA02000000                <1>       mov   edx, CODEC_MASTER_VOL_REG ; 02h ; Line Out
1312 000118CC E85AFDFFFF                <1>       call  codec_write
1313                                    <1> vt8233_vol_1:
1314 000118D1 C3                        <1>       retn
1315                                    <1>
1316                                    <1> ; CODE for SOUND BLASTER 16
1317                                    <1>
1318                                    <1> DetectSB:
1319                                    <1>       ; 24/04/2017
1320                                    <1>       ;pushad
1321                                    <1> ScanPort:
1322 000118D2 66BB1002                  <1>       mov     bx, 210h    ; start scanning ports
1323                                    <1>                         ; 210h, 220h, .. 260h
1324                                    <1> ResetDSP:
1325 000118D6 6689DA                    <1>       mov     dx, bx            ; try to reset the DSP.
1326 000118D9 6683C206                  <1>       add     dx, 06h
1327 000118DD B001                      <1>       mov   al, 1
1328 000118DF EE                        <1>       out   dx, al
1329                                    <1>
1330 000118E0 EC                        <1>       in    al, dx
1331 000118E1 EC                        <1>       in    al, dx
1332 000118E2 EC                        <1>       in    al, dx
1333 000118E3 EC                        <1>       in    al, dx
1334                                    <1>
1335 000118E4 30C0                      <1>       xor     al, al
1336 000118E6 EE                        <1>       out   dx, al
1337                                    <1>
1338 000118E7 6683C208                  <1>       add     dx, 08h
1339 000118EB 66B96400                  <1>       mov   cx, 100
1340                                    <1> WaitID:
1341 000118EF EC                        <1>       in    al, dx
1342 000118F0 08C0                      <1>       or      al, al
1343 000118F2 7804                      <1>       js      short GetID
1344 000118F4 E2F9                      <1>       loop    WaitID
1345 000118F6 EB0F                      <1>       jmp     short NextPort
1346                                    <1> GetID:
1347 000118F8 6683EA04                  <1>       sub     dx, 04h
1348 000118FC EC                        <1>       in    al, dx
1349 000118FD 3CAA                      <1>       cmp     al, 0AAh
1350 000118FF 7413                      <1>       je      short Found
1351 00011901 6683C204                  <1>       add     dx, 04h
1352 00011905 E2E8                      <1>       loop    WaitID
1353                                    <1> NextPort:
1354 00011907 6683C310                  <1>       add     bx, 10h           ; if not response,
1355 0001190B 6681FB6002                <1>       cmp     bx, 260h    ; try the next port.
1356 00011910 76C4                      <1>       jbe     short ResetDSP
1357 00011912 F9                        <1>       stc
1358 00011913 C3                        <1>       retn
1359                                    <1> Found:
1360 00011914 66891D[8A650100]          <1>       mov     [audio_io_base], bx      ; SB Port Address Found!
1361                                    <1> ScanIRQ:
1362                                    <1> SetIrqs:
1363 0001191B 28C0                      <1>       sub   al, al ; 0
1364 0001191D A2[82650100]              <1>       mov   [IRQnum], al ; reset
1365 00011922 A2[87650100]              <1>       mov   [audio_intr], al ; reset
1366                                    <1>
1367                                    <1>       ; ah > 0 -> set IRQ vector
1368                                    <1>       ; al = IRQ number
1369                                    <1>       ;mov   ax, 103h ; IRQ 3
1370                                    <1>       ;call  set_hardware_int_vector
1371                                    <1>       ;mov   ax, 104h ; IRQ 4
1372                                    <1>       ;call  set_hardware_int_vector
1373 00011927 66B80501                  <1>       mov   ax, 105h ; IRQ 5
1374 0001192B E8EEDDFFFF                <1>       call  set_hardware_int_vector
1375 00011930 66B80701                  <1>       mov   ax, 107h ; IRQ 7
1376 00011934 E8E5DDFFFF                <1>       call  set_hardware_int_vector
1377                                    <1>
1378 00011939 668B15[8A650100]          <1>       mov     dx, [audio_io_base] ; tells to the SB to
1379 00011940 6683C20C                  <1>       add     dx, 0Ch               ; generate a IRQ!
1380                                    <1> WaitSb:
1381 00011944 EC                        <1>       in    al, dx
1382 00011945 08C0                      <1>       or      al, al
1383 00011947 78FB                      <1>       js      short WaitSb
1384 00011949 B0F2                      <1>       mov     al, 0F2h
1385 0001194B EE                        <1>       out   dx, al
1386                                    <1>
1387 0001194C 31C9                      <1>       xor     ecx, ecx    ; wait until IRQ level
1388                                    <1> WaitIRQ:
1389 0001194E A0[82650100]              <1>       mov   al, [IRQnum]
1390 00011953 3C00                      <1>       cmp     al, 0 ; is changed or timeout.
1391 00011955 7706                      <1>       ja    short IrqOk
1392 00011957 6649                      <1>       dec   cx
1393 00011959 75F3                      <1>       jnz   short WaitIRQ
1394 0001195B EB15                      <1>       jmp   short RestoreIrqs
1395                                    <1> IrqOk:
```

```
1396 0001195D A2[87650100]        <1>       mov    [audio_intr], al ; set
1397 00011962 668B15[8A650100]    <1>       mov     dx, [audio_io_base]
1398 00011969 6683C20E            <1>       add     dx, 0Eh
1399 0001196D EC                  <1>       in     al, dx ; SB acknowledge.
1400 0001196E B020                <1>       mov    al, 20h
1401 00011970 E620                <1>       out    20h, al      ; Hardware acknowledge.
1402                              <1>
1403                              <1> RestoreIrqs:
1404                              <1>       ; ah = 0 -> reset IRQ vector
1405                              <1>       ; al = IRQ number
1406                              <1>       ;mov   ax, 3 ; IRQ 3
1407                              <1>       ;call  set_hardware_int_vector
1408                              <1>       ;mov   ax, 4 ; IRQ 4
1409                              <1>       ;call  set_hardware_int_vector
1410 00011972 66B80500            <1>       mov    ax, 5 ; IRQ 5
1411 00011976 E8A3DDFFFF          <1>       call   set_hardware_int_vector
1412 0001197B 66B80700            <1>       mov    ax, 7 ; IRQ 7
1413 0001197F E89ADDFFFF          <1>       call   set_hardware_int_vector
1414                              <1>
1415 00011984 31D2                <1>       xor    edx, edx
1416 00011986 8915[8C650100]      <1>       mov    [audio_dev_id], edx ; 0
1417 0001198C 8915[90650100]      <1>       mov    [audio_vendor], edx ; 0
1418 00011992 8915[94650100]      <1>       mov    [audio_stats_cmd], edx ; 0
1419                              <1>
1420                              <1>       ;popad
1421                              <1>
1422 00011998 803D[87650100]01    <1>       cmp     byte [audio_intr], 1 ; IRQ level was changed?
1423                              <1>
1424 0001199F C3                  <1>       retn
1425                              <1>
1426                              <1> %macro      SbOut  1
1427                              <1> %%Wait:
1428                              <1>       in     al, dx
1429                              <1>       or     al, al
1430                              <1>       js     short %%Wait
1431                              <1>       mov    al, %1
1432                              <1>       out    dx, al
1433                              <1> %endmacro
1434                              <1>
1435                              <1> SbInit_play:
1436                              <1>       ; 22/10/2017
1437                              <1>       ; 20/10/2017
1438                              <1>       ; 06/10/2017
1439                              <1>       ; 13/07/2017, 09/08/2017
1440                              <1>       ; 24/04/2017, 15/05/2017, 24/06/2017
1441                              <1>       ;pushad
1442                              <1> SetBuffer:
1443                              <1>       ;mov   byte [DmaFlag], 0
1444                              <1>
1445 000119A0 8B1D[A4650100]      <1>       mov    ebx, [audio_dma_buff] ; physical addr of DMA buff
1446 000119A6 89DF                <1>       mov    edi, ebx
1447 000119A8 8B0D[A8650100]      <1>       mov     ecx, [audio_dmabuff_size]
1448                              <1>
1449 000119AE 803D[B4650100]10    <1>       cmp    byte [audio_bps], 16
1450 000119B5 7531                <1>       jne    short sbInit_0 ; set 8 bit DMA buffer
1451                              <1>
1452                              <1>       ; 09/08/2017
1453                              <1>       ; convert byte count to word count
1454 000119B7 D1E9                <1>       shr    ecx, 1
1455 000119B9 49                  <1>       dec    ecx ; word count - 1
1456                              <1>       ; convert byte offset to word offset
1457 000119BA D1EB                <1>       shr    ebx, 1
1458                              <1>
1459                              <1>       ; 16 bit DMA buffer setting (DMA channel 5)
1460 000119BC B005                <1>       mov     al, 05h  ; set mask bit for channel 5  (4+1)
1461 000119BE E6D4                <1>       out    0D4h, al
1462                              <1>
1463 000119C0 30C0                <1>       xor     al, al   ; stops all DMA processes on selected channel
1464 000119C2 E6D8                <1>       out    0D8h, al ; clear selected channel register
1465                              <1>
1466 000119C4 88D8                <1>       mov     al, bl      ; byte 0 of DMA buffer offset in words (physical)
1467 000119C6 E6C4                <1>       out    0C4h, al ; DMA channel 5 port number
1468                              <1>
1469 000119C8 88F8                <1>       mov     al, bh   ; byte 1 of DMA buffer offset in words (physical)
1470 000119CA E6C4                <1>       out    0C4h, al
1471                              <1>
1472                              <1>       ; 09/08/2017
1473 000119CC C1EB0F              <1>       shr    ebx, 15       ; complete 16 bit shift
1474 000119CF 80E3FE              <1>       and    bl, 0FEh ; clear bit 0 (not necessary, it will be ignored)
1475                              <1>
1476 000119D2 88D8                <1>       mov     al, bl   ; byte 2 of DMA buffer address (physical)
1477 000119D4 E68B                <1>       out    8Bh, al  ; page register port addr for channel 5 ; 13/07/2017
1478                              <1>
1479 000119D6 88C8                <1>       mov     al, cl   ; low byte of DMA count - 1
1480 000119D8 E6C6                <1>       out    0C6h, al ; count register port addr for channel 1
1481                              <1>
1482 000119DA 88E8                <1>       mov     al, ch   ; high byte of DMA count - 1
1483 000119DC E6C6                <1>       out    0C6h, al
1484                              <1>
1485                              <1>       ; channel 5, read, autoinitialized, single mode
1486                              <1>       ;mov   al, 49h
1487 000119DE B059                <1>       mov    al, 59h  ; 06/10/2017
1488 000119E0 E6D6                <1>       out    0D6h, al ; DMA mode register port address
1489                              <1>
1490 000119E2 B001                <1>       mov     al, 01h  ; clear mask bit for channel 1
1491 000119E4 E6D4                <1>       out    0D4h, al ; DMA mask register port address
1492                              <1>
1493 000119E6 EB28                <1>       jmp    short ClearBuffer
1494                              <1>
1495                              <1> sbInit_0:
1496 000119E8 49                  <1>       dec     ecx ; 09/08/2017
1497                              <1>
```

```
1498                                    <1>     ; 8 bit DMA buffer setting (DMA channel 1)
1499 000119E9 B005                      <1>     mov     al, 05h ; set mask bit for channel 1  (4+1)
1500 000119EB E60A                      <1>     out    0Ah, al ; DMA mask register
1501                                    <1>
1502 000119ED 30C0                      <1>     xor     al, al  ; stops all DMA processes on selected channel
1503 000119EF E60C                      <1>     out    0Ch, al ; clear selected channel register
1504                                    <1>
1505 000119F1 88D8                      <1>     mov     al, bl      ; byte 0 of DMA buffer address (physical)
1506 000119F3 E602                      <1>     out    02h, al ; DMA channel 1 port number
1507                                    <1>
1508 000119F5 88F8                      <1>     mov     al, bh  ; byte 1 of DMA buffer address (physical)
1509 000119F7 E602                      <1>     out    02h, al
1510                                    <1>
1511 000119F9 C1EB10                    <1>     shr    ebx, 16
1512                                    <1>
1513 000119FC 88D8                      <1>     mov     al, bl  ; byte 2 of DMA buffer address (physical)
1514 000119FE E683                      <1>     out    83h, al ; page register port addr for channel 1
1515                                    <1>
1516 00011A00 88C8                      <1>     mov     al, cl  ; low byte of DMA count - 1
1517 00011A02 E603                      <1>     out    03h, al ; count register port addr for channel 1
1518                                    <1>
1519 00011A04 88E8                      <1>     mov     al, ch  ; high byte of DMA count - 1
1520 00011A06 E603                      <1>     out    03h, al
1521                                    <1>
1522                                    <1>     ; channel 1, read, autoinitialized, single mode
1523                                    <1>     ;mov   al, 49h
1524 00011A08 B059                      <1>     mov    al, 59h ; 06/10/2017
1525 00011A0A E60B                      <1>     out    0Bh, al ; DMA mode register port address
1526                                    <1>
1527 00011A0C B001                      <1>     mov     al, 01h ; clear mask bit for channel 1
1528 00011A0E E60A                      <1>     out    0Ah, al ; DMA mask register port address
1529                                    <1>
1530                                    <1> ClearBuffer:
1531                                    <1>     ;;mov  edi, [audio_dma_buff]
1532                                    <1>     ;;mov  ecx, [audio_dmabuff_size]
1533                                    <1>     ;inc   ecx
1534                                    <1>     ;mov     al, 80h
1535                                    <1>     ;;cld
1536                                    <1>     ;rep     stosb
1537                                    <1> SetIrq:
1538                                    <1>     ;mov   ebx, SbIrqhandler
1539                                    <1>     ;mov   al, [audio_intr] ; IRQ number
1540                                    <1>     ;call  set_dev_IRQ_service
1541                                    <1>     ;; SETUP (audio) INTERRUPT CALLBACK SERVICE
1542                                    <1>     ;mov   bl, [audio_intr] ; IRQ number
1543                                    <1>     ;mov   bh, [audio_cb_mode]
1544                                    <1>     ;inc   bh  ; 1 = Signal Response Byte method (fixed value)
1545                                    <1>     ;          ; 2 = Callback service method
1546                                    <1>     ;          ; 3 = Auto Increment S.R.B. method
1547                                    <1>     ;mov   cl, [audio_srb]
1548                                    <1>     ;mov   edx, [audio_cb_addr]
1549                                    <1>     ;mov   al, [audio_user]
1550                                    <1>     ;call  set_irq_callback_service
1551                                    <1> ResetDsp:
1552 00011A10 668B15[8A650100]          <1>     mov     dx, [audio_io_base]
1553 00011A17 6683C206                  <1>     add     dx, 06h
1554 00011A1B B001                      <1>     mov     al, 1
1555 00011A1D EE                        <1>     out    dx, al
1556                                    <1>
1557 00011A1E EC                        <1>     in     al, dx
1558 00011A1F EC                        <1>     in     al, dx
1559 00011A20 EC                        <1>     in     al, dx
1560 00011A21 EC                        <1>     in     al, dx
1561                                    <1>
1562 00011A22 30C0                      <1>     xor     al, al
1563 00011A24 EE                        <1>     out    dx, al
1564                                    <1>
1565 00011A25 66B96400                  <1>     mov     cx, 100
1566 00011A29 28E4                      <1>     sub    ah, ah ; 0
1567                                    <1> WaitId:
1568 00011A2B 668B15[8A650100]          <1>     mov     dx, [audio_io_base]
1569 00011A32 6683C20E                  <1>     add     dx, 0Eh
1570 00011A36 EC                        <1>     in     al, dx
1571 00011A37 08C0                      <1>     or      al, al
1572 00011A39 7807                      <1>     js      short sb_GetId
1573 00011A3B E2EE                      <1>     loop    WaitId
1574 00011A3D E9B4000000                <1>     jmp     sb_Exit
1575                                    <1> sb_GetId:
1576 00011A42 668B15[8A650100]          <1>     mov     dx, [audio_io_base]
1577 00011A49 6683C20A                  <1>     add     dx, 0Ah
1578 00011A4D EC                        <1>     in     al, dx
1579 00011A4E 3CAA                      <1>     cmp     al, 0AAh
1580 00011A50 7407                      <1>     je      short SbOk
1581 00011A52 E2D7                      <1>     loop    WaitId
1582 00011A54 E99D000000                <1>     jmp    sb_Exit
1583                                    <1> SbOk:
1584 00011A59 668B15[8A650100]          <1>     mov     dx, [audio_io_base]
1585 00011A60 6683C20C                  <1>     add     dx, 0Ch
1586                                    <1>     SbOut   0D1h ; Turn on speaker
1586                                    <2> %%Wait:
1586 00011A64 EC                        <2>  in al, dx
1586 00011A65 08C0                      <2>  or al, al
1586 00011A67 78FB                      <2>  js short %%Wait
1586 00011A69 B0D1                      <2>  mov al, %1
1586 00011A6B EE                        <2>  out dx, al
1587                                    <1>     SbOut   41h ; 8 bit or 16 bit transfer
1587                                    <2> %%Wait:
1587 00011A6C EC                        <2>  in al, dx
1587 00011A6D 08C0                      <2>  or al, al
1587 00011A6F 78FB                      <2>  js short %%Wait
1587 00011A71 B041                      <2>  mov al, %1
1587 00011A73 EE                        <2>  out dx, al
```

```
1588 00011A74 668B1D[B6650100]    <1>       mov    bx, [audio_freq] ; sampling rate (Hz)
1589                              <1>       SbOut  bh ; sampling rate high byte
1589                              <2> %%Wait:
1589 00011A7B EC                  <2>  in al, dx
1589 00011A7C 08C0                <2>  or al, al
1589 00011A7E 78FB                <2>  js short %%Wait
1589 00011A80 88F8                <2>  mov al, %1
1589 00011A82 EE                  <2>  out dx, al
1590                              <1>       SbOut  bl ; sampling rate low byte
1590                              <2> %%Wait:
1590 00011A83 EC                  <2>  in al, dx
1590 00011A84 08C0                <2>  or al, al
1590 00011A86 78FB                <2>  js short %%Wait
1590 00011A88 88D8                <2>  mov al, %1
1590 00011A8A EE                  <2>  out dx, al
1591                              <1>
1592                              <1>       ; 22/05/2017
1593 00011A8B E8C0000000          <1>       call   sb16_volume_initial ; 15/05/2017
1594                              <1>       ; 20/05/2017
1595                              <1>       ;call  sb16_volume
1596                              <1>
1597                              <1> StartDma:
1598                              <1>       ; autoinitialized mode
1599 00011A90 803D[B4650100]10    <1>       cmp    byte [audio_bps], 16 ; 16 bit samples
1600 00011A97 7411                <1>       je     short sb_play_1
1601                              <1>       ; 8 bit samples
1602 00011A99 66BBC600            <1>       mov    bx, 0C6h ; 8 bit output (0C6h)
1603 00011A9D 803D[B5650100]02    <1>       cmp    byte [audio_stmo], 2 ; 1 = mono, 2 = stereo
1604 00011AA4 7214                <1>       jb     short sb_play_2
1605 00011AA6 B720                <1>       mov    bh, 20h      ; 8 bit stereo (20h)
1606 00011AA8 EB10                <1>       jmp    short sb_play_2
1607                              <1> sb_play_1:
1608                              <1>       ; 16 bit samples
1609 00011AAA 66BBB610            <1>       mov    bx, 10B6h ; 16 bit output (0B6h)
1610 00011AAE 803D[B5650100]02    <1>       cmp    byte [audio_stmo], 2 ; 1 = mono, 2 = stereo
1611 00011AB5 7203                <1>       jb     short sb_play_2
1612 00011AB7 80C720              <1>       add    bh, 20h      ; 16 bit stereo (30h)
1613                              <1> sb_play_2:
1614                              <1>       ; PCM output (8/16 bit mono autoinitialized transfer)
1615                              <1>       SbOut   bl ; bCommand
1615                              <2> %%Wait:
1615 00011ABA EC                  <2>  in al, dx
1615 00011ABB 08C0                <2>  or al, al
1615 00011ABD 78FB                <2>  js short %%Wait
1615 00011ABF 88D8                <2>  mov al, %1
1615 00011AC1 EE                  <2>  out dx, al
1616                              <1>       SbOut  bh ; bMode
1616                              <2> %%Wait:
1616 00011AC2 EC                  <2>  in al, dx
1616 00011AC3 08C0                <2>  or al, al
1616 00011AC5 78FB                <2>  js short %%Wait
1616 00011AC7 88F8                <2>  mov al, %1
1616 00011AC9 EE                  <2>  out dx, al
1617 00011ACA 8B1D[A8650100]      <1>       mov    ebx, [audio_dmabuff_size]  ; 15/05/2017
1618 00011AD0 D1EB                <1>       shr    ebx, 1 ; half buffer size
1619                              <1>       ; 20/10/2017
1620 00011AD2 803D[B4650100]10    <1>       cmp    byte [audio_bps], 16 ; 16 bit DMA
1621 00011AD9 7502                <1>       jne    short sb_play_3
1622 00011ADB D1EB                <1>       shr    ebx, 1 ; byte count to word count
1623                              <1> sb_play_3:
1624 00011ADD 664B                <1>       dec    bx  ; wBlkSize is one less than the actual size
1625                              <1>       SbOut   bl
1625                              <2> %%Wait:
1625 00011ADF EC                  <2>  in al, dx
1625 00011AE0 08C0                <2>  or al, al
1625 00011AE2 78FB                <2>  js short %%Wait
1625 00011AE4 88D8                <2>  mov al, %1
1625 00011AE6 EE                  <2>  out dx, al
1626                              <1>       SbOut   bh
1626                              <2> %%Wait:
1626 00011AE7 EC                  <2>  in al, dx
1626 00011AE8 08C0                <2>  or al, al
1626 00011AEA 78FB                <2>  js short %%Wait
1626 00011AEC 88F8                <2>  mov al, %1
1626 00011AEE EE                  <2>  out dx, al
1627                              <1>
1628 00011AEF C605[B8650100]01    <1>       mov    byte [audio_play_cmd], 1 ; playing !
1629                              <1>
1630                              <1>       ;; Set Voice and master volumes
1631                              <1>       ;mov   dx, [audio_io_base]
1632                              <1>       ;add   dl, 4 ; Mixer chip Register Address Port
1633                              <1>       ;SbOut 30h   ; select Master Volume Register (L)
1634                              <1>       ;inc   dl    ; Mixer chip Register Data Port
1635                              <1>       ;SbOut 0F8h  ; Max. volume value is 31 (31*8)
1636                              <1>       ;dec   dl
1637                              <1>       ;SbOut 31h   ; select Master Volume Register (R)
1638                              <1>       ;inc   dl
1639                              <1>       ;SbOut 0F8h  ; Max. volume value is 31 (31*8)
1640                              <1>       ;dec   dl
1641                              <1>       ;SbOut 32h   ; select Voice Volume Register (L)
1642                              <1>       ;inc   dl
1643                              <1>       ;SbOut 0F8h  ; Max. volume value is 31 (31*8)
1644                              <1>       ;dec   dl
1645                              <1>       ;SbOut 33h   ; select Voice Volume Register (R)
1646                              <1>       ;inc   dl
1647                              <1>       ;SbOut 0F8h  ; Max. volume value is 31 (31*8)
1648                              <1>       ;;
1649                              <1>       ;dec   dl
1650                              <1>       ;SbOut 44h   ; select Treble Register (L)
1651                              <1>       ;inc   dl
1652                              <1>       ;SbOut 0F0h  ; Max. Treble value is 15 (15*16)
1653                              <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
00011AF6 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>
00011AF7 668B15[8A650100]       <1>       mov     dx, [audio_io_base]
                                <1>       ; 20/10/2017
00011AFE 80C20F                 <1>       add     dl, 0Fh ; 2xFh (DSP 16 bit intr ack)
00011B01 803D[B4650100]10       <1>       cmp    byte [audio_bps], 16
00011B08 7402                   <1>       je     short sb_irq_16bit_ack
                                <1> sb_irq_8bit_ack:
00011B0A FECA                   <1>       dec    dl  ; 2xEh (DSP 8 bit intr ack)
                                <1> sb_irq_16bit_ack:
00011B0C 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>
00011B0D 803D[B8650100]01       <1>       cmp    byte [audio_play_cmd], 1
00011B14 7307                   <1>       jnb    short sb_irq_h1
                                <1> sb_irq_h0:
00011B16 E8A9000000             <1>       call   sb16_stop
00011B1B EB2B                   <1>       jmp    short sb_irq_h3
                                <1> sb_irq_h1:
                                <1>       ;call  sb16_tuneloop
                                <1>       ; 09/10/2017
                                <1> sb16_tuneloop:
00011B1D 8B3D[A4650100]         <1>       mov    edi, [audio_dma_buff]
00011B23 8B0D[A8650100]         <1>       mov    ecx, [audio_dmabuff_size]
00011B29 D1E9                   <1>       shr    ecx, 1 ; dma buff size / 2 = half buffer size
                                <1>
                                <1>       ; 22/05/2017
00011B2B F605[AC650100]01       <1>       test   byte [audio_flag], 1  ; Current flag value
00011B32 7402                   <1>       jz     short sb_tlp1 ; EOL (Half Buffer 1 must be filled)
                                <1>       ; FLAG (Half Buffer 2 must be filled)
00011B34 01CF                   <1>       add    edi, ecx
                                <1>       ; 15/05/2017
                                <1> sb_tlp1:
00011B36 8B35[9C650100]         <1>       mov    esi, [audio_p_buffer] ; phy addr of audio buff
                                <1>       ;rep   movsb
00011B3C C1E902                 <1>       shr    ecx, 2 ; half buff size / 4
00011B3F F3A5                   <1>       rep    movsd
                                <1>       ;retn
                                <1>
                                <1>       ; 10/10/2017
                                <1>       ; switch flag value
00011B41 8035[AC650100]01       <1>       xor    byte [audio_flag], 1
                                <1>
                                <1>       ; 12/10/2017
                                <1>       ; [audio_flag] = 0 : Playing dma half buffer 2 (odd intr count)
                                <1>                          ; Next buffer (to update) is dma half buff 1
                                <1>       ;             = 1 : Playing dma half buffer 1 (even intr count)
                                <1>                          ; Next buffer (to update) is dma half buff 2
                                <1>
                                <1> sb_irq_h3:
                                <1>       ;mov   byte [audio_busy], 0
                                <1>
                                <1>       ;pop   edi
                                <1>       ;pop   esi
                                <1>       ;pop   edx
                                <1>       ;pop   ecx
                                <1>       ;pop   ebx ; * must be restored !
                                <1>       ;pop   eax ; * must be restored !
                                <1>
00011B48 C3                     <1>       retn
                                <1>
                                <1> sb16_volume:
                                <1>       ; 22/10/2017
                                <1>       ; mov [audio_master_volume_l], cl
                                <1>       ; mov [audio_master_volume_h], ch
00011B49 66890D[BA650100]       <1>       mov    [audio_master_volume], cx
                                <1> sb16_volume_initial:
```

```
1756 00011B50 6652                <1>       push   dx ; DX (port address) must be saved
1757 00011B52 668B15[8A650100]    <1>       mov    dx, [audio_io_base]
1758 00011B59 6683C204            <1>       add    dx, 4 ; Mixer chip address port
1759 00011B5D B022                <1>       mov    al, 22h ; master volume
1760 00011B5F EE                  <1>       out    dx, al
1761 00011B60 6642                <1>       inc    dx
1762 00011B62 8A25[BA650100]      <1>       mov    ah, [audio_master_volume_l]
1763 00011B68 C0EC02              <1>       shr    ah, 2 ; 32 -> 8 level
1764 00011B6B C0E405              <1>       shl    ah, 5 ; bit 5 to 7
1765 00011B6E A0[BB650100]        <1>       mov    al, [audio_master_volume_r]
1766 00011B73 C0E802              <1>       shr    al, 2 ; 32 -> 8 level
1767                              <1>       ;and   al, 0Fh
1768 00011B76 D0E0                <1>       shl    al, 1 ; bit 1 to 3
1769 00011B78 08E0                <1>       or     al, ah
1770 00011B7A EE                  <1>       out    dx, al
1771 00011B7B 665A                <1>       pop    dx ; DX (port address) must be restored
1772 00011B7D C3                  <1>       retn
1773                              <1>
1774                              <1> sb16_pause:
1775 00011B7E 668B15[8A650100]    <1>       mov    dx, [audio_io_base]
1776 00011B85 6683C20C            <1>       add    dx, 0Ch ; Command & Data Port
1777 00011B89 803D[B4650100]10    <1>       cmp    byte [audio_bps], 16 ; 16 bit samples
1778 00011B90 7404                <1>       je     short sb_pause_1
1779                              <1>       ; 8 bit samples
1780 00011B92 B3D0                <1>       mov    bl, 0D0h ; 8 bit DMA mode
1781 00011B94 EB02                <1>       jmp    short sb_pause_2
1782                              <1> sb_pause_1:
1783                              <1>       ; 16 bit samples
1784 00011B96 B3D5                <1>       mov    bl, 0D5h ; 16 bit DMA mode
1785                              <1> sb_pause_2:
1786                              <1>       SbOut   bl ; bCommand
1786                              <2> %%Wait:
1786 00011B98 EC                  <2>  in al, dx
1786 00011B99 08C0                <2>  or al, al
1786 00011B9B 78FB                <2>  js short %%Wait
1786 00011B9D 88D8                <2>  mov al, %1
1786 00011B9F EE                  <2>  out dx, al
1787                              <1> sb_pause_3:
1788 00011BA0 C3                  <1>       retn
1789                              <1>
1790                              <1> sb16_continue:
1791 00011BA1 668B15[8A650100]    <1>       mov    dx, [audio_io_base]
1792 00011BA8 6683C20C            <1>       add    dx, 0Ch ; Command & Data Port
1793 00011BAC 803D[B4650100]10    <1>       cmp    byte [audio_bps], 16 ; 16 bit samples
1794 00011BB3 7404                <1>       je     short sb_cont_1
1795                              <1>       ; 8 bit samples
1796 00011BB5 B3D4                <1>       mov    bl, 0D4h ; 8 bit DMA mode
1797 00011BB7 EB02                <1>       jmp    short sb_cont_2
1798                              <1> sb_cont_1:
1799                              <1>       ; 16 bit samples
1800 00011BB9 B3D6                <1>       mov    bl, 0D6h ; 16 bit DMA mode
1801                              <1> sb_cont_2:
1802                              <1>       SbOut   bl ; bCommand
1802                              <2> %%Wait:
1802 00011BBB EC                  <2>  in al, dx
1802 00011BBC 08C0                <2>  or al, al
1802 00011BBE 78FB                <2>  js short %%Wait
1802 00011BC0 88D8                <2>  mov al, %1
1802 00011BC2 EE                  <2>  out dx, al
1803                              <1> sb_cont_3:
1804 00011BC3 C3                  <1>       retn
1805                              <1>
1806                              <1> sb16_stop:
1807                              <1>       ; 24/04/2017
1808 00011BC4 803D[B8650100]00    <1>       cmp    byte [audio_play_cmd], 0
1809 00011BCB 7648                <1>       jna    short sb16_stop_4
1810                              <1>
1811                              <1>       ; 22/05/2017
1812 00011BCD 668B15[8A650100]    <1>       mov    dx, [audio_io_base]
1813 00011BD4 6683C20C            <1>       add    dx, 0Ch
1814                              <1>
1815 00011BD8 B3D9                <1>       mov    bl, 0D9h ; exit auto-initialize 16 bit transfer
1816                              <1>       ; stop  autoinitialized DMA transfer mode
1817 00011BDA 803D[B4650100]10    <1>       cmp    byte [audio_bps], 16 ; 16 bit samples
1818 00011BE1 7402                <1>       je     short sb16_stop_1
1819                              <1>       ;mov   bl, 0DAh ; exit auto-initialize 8 bit transfer
1820 00011BE3 FEC3                <1>       inc    bl
1821                              <1> sb16_stop_1:
1822                              <1>       SbOut  bl ; exit auto-initialize transfer command
1822                              <2> %%Wait:
1822 00011BE5 EC                  <2>  in al, dx
1822 00011BE6 08C0                <2>  or al, al
1822 00011BE8 78FB                <2>  js short %%Wait
1822 00011BEA 88D8                <2>  mov al, %1
1822 00011BEC EE                  <2>  out dx, al
1823                              <1>
1824 00011BED 30C0                <1>       xor     al, al ; stops all DMA processes on selected channel
1825                              <1>
1826 00011BEF 803D[B4650100]10    <1>       cmp    byte [audio_bps], 16 ; 16 bit samples
1827 00011BF6 7404                <1>       je     short sb16_stop_2
1828 00011BF8 E60C                <1>       out    0Ch, al ; clear selected channel register
1829 00011BFA EB02                <1>       jmp    short sb16_stop_3
1830                              <1>
1831                              <1> sb16_stop_2:
1832 00011BFC E6D8                <1>       out    0D8h, al ; clear selected channel register
1833                              <1>
1834                              <1> sb16_stop_3:
1835 00011BFE C605[B8650100]00    <1>       mov    byte [audio_play_cmd], 0 ; stop !
1836                              <1> SbDone:
1837                              <1>       ;mov   dx, [audio_io_base]
1838                              <1>       ;add   dx, 0Ch
1839                              <1>       SbOut   0D0h
```

```
1839                                    <2> %%Wait:
1839 00011C05 EC                        <2>  in al, dx
1839 00011C06 08C0                      <2>  or al, al
1839 00011C08 78FB                      <2>  js short %%Wait
1839 00011C0A B0D0                      <2>  mov al, %1
1839 00011C0C EE                        <2>  out dx, al
1840                                    <1>       SbOut   0D3h
1840                                    <2> %%Wait:
1840 00011C0D EC                        <2>  in al, dx
1840 00011C0E 08C0                      <2>  or al, al
1840 00011C10 78FB                      <2>  js short %%Wait
1840 00011C12 B0D3                      <2>  mov al, %1
1840 00011C14 EE                        <2>  out dx, al
1841                                    <1> sb16_stop_4:
1842 00011C15 C3                        <1>       retn
1843                                    <1>
1844                                    <1> sb16_reset:
1845                                    <1>       ; 24/04/2017
1846 00011C16 668B15[8A650100]          <1>       mov     dx, [audio_io_base] ; try to reset the DSP.
1847 00011C1D 6683C206                  <1>       add     dx, 06h
1848 00011C21 B001                      <1>       mov    al, 1
1849 00011C23 EE                        <1>       out    dx, al
1850                                    <1>
1851 00011C24 EC                        <1>       in     al, dx
1852 00011C25 EC                        <1>       in     al, dx
1853 00011C26 EC                        <1>       in     al, dx
1854 00011C27 EC                        <1>       in     al, dx
1855                                    <1>
1856 00011C28 30C0                      <1>       xor     al, al
1857 00011C2A EE                        <1>       out    dx, al
1858                                    <1>
1859 00011C2B 6683C208                  <1>       add     dx, 08h
1860 00011C2F 66B96400                  <1>       mov    cx, 100
1861                                    <1> sbrstWaitID:
1862 00011C33 EC                        <1>       in     al, dx
1863 00011C34 08C0                      <1>       or      al, al
1864 00011C36 7804                      <1>       js      short sbrstGetID
1865 00011C38 E2F9                      <1>       loop    sbrstWaitID
1866 00011C3A F9                        <1>       stc
1867 00011C3B C3                        <1>       retn
1868                                    <1> sbrstGetID:
1869 00011C3C 6683EA04                  <1>       sub     dx, 04h
1870 00011C40 EC                        <1>       in     al, dx
1871 00011C41 3CAA                      <1>       cmp     al, 0AAh
1872 00011C43 7406                      <1>       je      short sb_rst_retn
1873 00011C45 6683C204                  <1>       add     dx, 04h
1874 00011C49 E2E8                      <1>       loop    sbrstWaitID
1875                                    <1> sb_rst_retn:
1876 00011C4B C3                        <1>       retn
1877                                    <1>
1878                                    <1> ac97_codec_config:
1879                                    <1>       ; 10/06/2017
1880                                    <1>       ; 05/06/2017
1881                                    <1>       ; 29/05/2017
1882                                    <1>       ; 28/05/2017 (TRDOS 386, 'audio.s')
1883                                    <1>       ; 07/11/2016 (Erdogan Tan)
1884                                    <1>       ; Derived from 'codecConfig' procedure in 'CODEC.ASM'
1885                                    <1>       ; .wav player for DOS by Jeff Leyda (02/09/2002)
1886                                    <1>
1887                                    <1>       ;; 'PLAYER.ASM'
1888                                    <1>       ;; get ICH base address regs for mixer and bus master
1889                                    <1>
1890                                    <1> init_ac97_controller: ; 10/06/2017
1891 00011C4C A1[8C650100]              <1>       mov    eax, [audio_dev_id]
1892                                    <1>         ;mov al, NAMBAR_REG
1893                                    <1>         ;;call  pciRegRead16                ; read PCI registers 10-11
1894                                    <1>         ;call      pciRegRead32
1895                                    <1>       ;and   dx, IO_ADDR_MASK          ; mask off BIT0
1896                                    <1>       ;;and  edx, IO_ADDR_MASK
1897                                    <1>
1898                                    <1>         ;mov [NAMBAR], dx              ; save audio mixer base addr
1899                                    <1>
1900                                    <1>         ;mov    al, NABMBAR_REG
1901                                    <1>         ;;call     pciRegRead16
1902                                    <1>         ;call      pciRegRead32
1903                                    <1>       ;and   dx, 0FFC0h ; IO_ADDR_MASK
1904                                    <1>       ;;and  edx, 0FFC0h
1905                                    <1>
1906                                    <1>         ;mov    [NABMBAR], dx               ; save bus master base addr
1907                                    <1>
1908                                    <1>       ;mov   eax, [audio_dev_id]
1909 00011C51 B004                      <1>         mov     al, PCI_CMD_REG
1910                                    <1>         ;call      pciRegRead8              ; read PCI command register
1911 00011C53 E840F8FFFF                <1>         call pciRegRead16
1912 00011C58 80CA05                    <1>       or      dl, IO_ENA+BM_ENA             ; enable IO and bus master
1913                                    <1>         ;call      pciRegWrite8
1914 00011C5B E8A3F8FFFF                <1>       call   pciRegWrite16
1915                                    <1>
1916                                    <1>       ; 'CODEC.ASM'
1917                                    <1>
1918                                    <1>       ; enable codec, unmute stuff, set output rate
1919                                    <1> ;     ; entry: [audio_freq] = desired sample rate
1920                                    <1>
1921                                    <1> ;     mov          dx, [NAMBAR]
1922                                    <1> ;     add          dx, CODEC_EXT_AUDIO_CTRL_REG     ; 2Ah
1923                                    <1> ;     in           ax, dx
1924                                    <1> ;     or     ax, 1
1925                                    <1> ;     out    dx, ax                          ; Enable variable rate audio
1926                                    <1>
1927                                    <1> ;       ;call    delay1_4ms
1928                                    <1> ;       ;call    delay1_4ms
1929                                    <1> ;       ;call    delay1_4ms
```

```
1930                                    <1> ;       ;call    delay1_4ms
1931                                    <1>
1932                                    <1> ;     mov    ax, [audio_freq]          ; sample rate
1933                                    <1>
1934                                    <1> ;     mov          dx, [NAMBAR]
1935                                    <1> ;     add          dx, CODEC_PCM_FRONT_DACRATE_REG  ; 2Ch
1936                                    <1> ;     out    dx, ax                          ; out sample rate
1937                                    <1>
1938                                    <1> ;       ;call      delay1_4ms
1939                                    <1> ;       ;call      delay1_4ms
1940                                    <1> ;       ;call      delay1_4ms
1941                                    <1> ;       ;call      delay1_4ms
1942                                    <1>
1943                                    <1>       ;mov   dx, [NAMBAR]              ; mixer base address
1944                                    <1>         ;add dx, CODEC_RESET_REG             ; reset register
1945                                    <1>         ;mov ax, 42
1946                                    <1>       ;out   dx, ax                         ; reset
1947                                    <1>
1948                                    <1>       ;mov    dx, [NABMBAR]                ; bus master base address
1949                                    <1>         ;add dx, GLOB_STS_REG
1950                                    <1>         ;mov ax, 2
1951                                    <1>       ;out   dx, ax
1952                                    <1>
1953 00011C60 E831F9FFFF               <1>         call    delay_100ms ; 29/05/2017
1954                                    <1>
1955                                    <1> init_ac97_codec:
1956                                    <1>       ; 10/06/2017
1957                                    <1>       ; 29/05/2017
1958                                    <1>       ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, intelac97.asm)
1959                                    <1>       ;
1960 00011C65 66BA2C00                 <1>       mov    dx, GLOB_CNT_REG ; 2Ch
1961 00011C69 660315[BE650100]         <1>       add    dx, [NABMBAR]
1962 00011C70 ED                       <1>       in     eax, dx
1963                                    <1>       ; ?
1964 00011C71 66BA3000                 <1>       mov    dx, GLOB_STS_REG ; 30h
1965 00011C75 660315[BE650100]         <1>       add    dx, [NABMBAR]
1966 00011C7C ED                       <1>       in     eax, dx
1967                                    <1>
1968 00011C7D 83F8FF                   <1>       cmp    eax, 0FFFFFFFFh ; -1
1969 00011C80 744B                     <1>       je     short init_ac97_codec_err1
1970                                    <1>
1971 00011C82 A900030010               <1>       test   eax, CTRL_ST_CREADY
1972 00011C87 7507                     <1>       jnz    short _ac97_codec_ready
1973                                    <1>
1974 00011C89 E8EF020000               <1>       call   reset_ac97_codec
1975 00011C8E 723E                     <1>       jc     short init_ac97_codec_err2
1976                                    <1>
1977                                    <1> _ac97_codec_ready:
1978 00011C90 668B15[BC650100]         <1>       mov    dx, [NAMBAR]
1979                                    <1>       ;add   dx, 0 ; ac_reg_0 ; reset register
1980 00011C97 66EF                     <1>       out    dx, ax
1981                                    <1>
1982 00011C99 31C0                     <1>       xor    eax, eax ; 0
1983 00011C9B 668B15[BC650100]         <1>       mov    dx, [NAMBAR]
1984 00011CA2 6683C226                 <1>       add    dx, CODEC_REG_POWERDOWN
1985 00011CA6 66EF                     <1>       out    dx, ax
1986                                    <1>
1987                                    <1>       ; 10/06/2017
1988                                    <1>       ; 29/05/2017
1989                                    <1>       ; wait for 1 second
1990 00011CA8 B9E8030000               <1>       mov    ecx, 1000 ; 1000*0.25ms = 1s
1991                                    <1> _ac97_codec_rloop:
1992 00011CAD E8F1F8FFFF               <1>       call   delay1_4ms
1993 00011CB2 E8ECF8FFFF               <1>       call   delay1_4ms
1994 00011CB7 E8E7F8FFFF               <1>       call   delay1_4ms
1995 00011CBC E8E2F8FFFF               <1>       call   delay1_4ms
1996                                    <1>       ;mov   dx, [NAMBAR]
1997                                    <1>       ;add   dx, CODEC_REG_POWERDOWN
1998 00011CC1 66ED                     <1>       in     ax, dx
1999 00011CC3 6683E00F                 <1>       and    ax, 0Fh
2000 00011CC7 3C0F                     <1>       cmp    al, 0Fh
2001 00011CC9 7404                     <1>       je     short _ac97_codec_init_ok
2002 00011CCB E2E0                     <1>       loop   _ac97_codec_rloop
2003                                    <1>
2004                                    <1> init_ac97_codec_err1:
2005 00011CCD F9                       <1>       stc
2006                                    <1> init_ac97_codec_err2:
2007 00011CCE C3                       <1>       retn
2008                                    <1>
2009                                    <1> _ac97_codec_init_ok:
2010 00011CCF B002                     <1>         mov     al, 2 ; force set 16-bit 2-channel PCM
2011 00011CD1 66BA2C00                 <1>       mov    dx, GLOB_CNT_REG ; 2Ch
2012 00011CD5 660315[BE650100]         <1>       add    dx, [NABMBAR]
2013 00011CDC EF                       <1>       out    dx, eax
2014                                    <1>
2015                                    <1>       ;call  delay1_4ms
2016                                    <1>
2017                                    <1>       ; 10/06/2017
2018 00011CDD E849020000               <1>       call   reset_ac97_controller
2019                                    <1>
2020                                    <1> ;     call   setup_ac97_codec
2021                                    <1> ;
2022                                    <1> ;detect_ac97_codec:
2023                                    <1> ;     retn
2024                                    <1>
2025                                    <1> setup_ac97_codec:
2026                                    <1>       ; 10/06/2017
2027                                    <1>       ; 29/05/2017
2028 00011CE2 B802020000               <1>       mov     eax, 0202h
2029 00011CE7 66A3[BA650100]           <1>       mov    [audio_master_volume], ax
2030 00011CED 66B81F1F                 <1>       mov    ax, 1F1Fh ; 31, 31
2031                                    <1>
```

```
2032 00011CF1 668B15[BC650100]    <1>       mov     dx, [NAMBAR]
2033 00011CF8 6683C202            <1>       add     dx, CODEC_MASTER_VOL_REG        ;02h
2034 00011CFC 6631C0              <1>       xor    ax, ax       ; volume attenuation = 0 (max. volume)
2035 00011CFF 66EF                <1>       out     dx, ax
2036                              <1>
2037 00011D01 668B15[BC650100]    <1>       mov     dx, [NAMBAR]
2038 00011D08 6683C206            <1>       add     dx, CODEC_MASTER_MONO_VOL_REG   ;06h
2039                              <1>       ;xor   ax, ax
2040 00011D0C 66EF                <1>       out     dx, ax
2041                              <1>
2042 00011D0E 668B15[BC650100]    <1>       mov     dx, [NAMBAR]
2043 00011D15 6683C20A            <1>       add     dx, CODEC_PCBEEP_VOL_REG        ;0Ah
2044                              <1>       ;xor    ax, ax
2045 00011D19 66EF                <1>       out     dx, ax
2046                              <1>
2047 00011D1B 668B15[BC650100]    <1>       mov     dx, [NAMBAR]
2048 00011D22 6683C218            <1>       add     dx, CODEC_PCM_OUT_REG           ;18h
2049                              <1>       ;xor    ax, ax
2050 00011D26 66EF                <1>       out     dx, ax
2051                              <1>
2052 00011D28 66B80880            <1>       mov     ax,  8008h ; Mute
2053 00011D2C 668B15[BC650100]    <1>       mov     dx, [NAMBAR]
2054 00011D33 6683C20C            <1>       add    dx, 0Ch       ; AC97_PHONE_VOL ; TAD Input (Mono)
2055 00011D37 66EF                <1>       out     dx, ax
2056                              <1>
2057 00011D39 66B80808            <1>         mov  ax,  0808h
2058 00011D3D 668B15[BC650100]    <1>       mov     dx, [NAMBAR]
2059 00011D44 6683C210            <1>         add  dx, CODEC_LINE_IN_VOL_REG ; 10h ; Line Input (Stereo)
2060 00011D48 66EF                <1>       out     dx, ax
2061                              <1>
2062                              <1>       ;mov   ax,  0808h
2063 00011D4A 668B15[BC650100]    <1>       mov     dx, [NAMBAR]
2064 00011D51 6683C212            <1>         add  dx, CODEC_CD_VOL_REG ; 12h ; CR Input (Stereo)
2065 00011D55 66EF                <1>       out     dx, ax
2066                              <1>
2067                              <1>       ;mov   ax,  0808h
2068 00011D57 668B15[BC650100]    <1>       mov     dx, [NAMBAR]
2069 00011D5E 6683C216            <1>         add  dx, CODEC_AUX_VOL_REG ; 16h ; Aux Input (Stereo)
2070 00011D62 66EF                <1>       out     dx, ax
2071                              <1>
2072                              <1>         ;call    delay1_4ms
2073                              <1>         ;call    delay1_4ms
2074                              <1>         ;call    delay1_4ms
2075                              <1>         ;call    delay1_4ms
2076                              <1>
2077                              <1> detect_ac97_codec:
2078 00011D64 C3                  <1>         retn
2079                              <1>
2080                              <1> set_ac97_bdl: ; Set AC97 (ICH) Buffer Descriptor List
2081                              <1>       ; 17/06/2017
2082                              <1>       ; 11/06/2017
2083                              <1>       ; 28/05/2017
2084                              <1>       ; eax = dma buffer address = [audio_DMA_buff]
2085                              <1>       ; ecx = dma buffer buffer size = [audio_dmabuff_size]
2086                              <1>
2087 00011D65 D1E9                <1>       shr    ecx, 1 ; dma half buffer size
2088 00011D67 89CE                <1>       mov    esi, ecx
2089                              <1>
2090 00011D69 BF[C0650100]        <1>         mov     edi, audio_bdl_buff    ; get BDL address
2091 00011D6E B910000000          <1>         mov     ecx, 32 / 2            ; make 32 entries in BDL
2092                              <1>
2093 00011D73 EB05                <1>       jmp    short s_ac97_bdl1
2094                              <1>
2095                              <1> s_ac97_bdl0:
2096                              <1>       ; set buffer descriptor 0 to start of data file in memory
2097                              <1>
2098 00011D75 A1[A4650100]        <1>       mov    eax, [audio_dma_buff]     ; Physical address of DMA buffer
2099                              <1>
2100                              <1> s_ac97_bdl1:
2101 00011D7A AB                  <1>       stosd                     ; store dmabuffer1 address
2102                              <1>
2103 00011D7B 89C2                <1>         mov    edx, eax
2104                              <1>
2105                              <1> ;
2106                              <1> ; Buffer Descriptors List
2107                              <1> ; As stated earlier, each buffer descriptor list is a set of (up to) 32
2108                              <1> ; descriptors, each 8 bytes in length.  Bytes 0-3 of a descriptor entry point
2109                              <1> ; to a chunk of memory to either play from or record to.  Bytes 4-7 of an
2110                              <1> ; entry describe various control things detailed below.
2111                              <1> ;
2112                              <1> ; Buffer pointers must always be aligned on a Dword boundry.
2113                              <1> ;
2114                              <1> ;
2115                              <1>
2116                              <1> ;IOC                     equ     BIT31 ; Fire an interrupt whenever this
2117                              <1>                                        ; buffer is complete.
2118                              <1>
2119                              <1> ;BUP                     equ     BIT30  ; Buffer Underrun Policy.
2120                              <1>                                        ; if this buffer is the last buffer
2121                              <1>                                        ; in a playback, fill the remaining
2122                              <1>                                        ; samples with 0 (silence) or not.
2123                              <1>                                        ; It's a good idea to set this to 1
2124                              <1>                                        ; for the last buffer in playback,
2125                              <1>                                        ; otherwise you're likely to get a lot
2126                              <1>                                        ; of noise at the end of the sound.
2127                              <1>
2128                              <1> ;
2129                              <1> ; Bits 15:0 contain the length of the buffer, in number of samples, which
2130                              <1> ; are 16 bits each, coupled in left and right pairs, or 32bits each.
2131                              <1> ; Luckily for us, that's the same format as .wav files.
2132                              <1> ;
2133                              <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>
00011D7D 89F0           <1>       mov    eax, esi ; DMA half buffer size
00011D7F 01C2           <1>       add    edx, eax
00011D81 D1E8           <1>       shr    eax, 1 ; count of 16 bit samples
                        <1>       ;or    eax, IOC+BUP
00011D83 0D00000080     <1>       or     eax, IOC ; 11/06/2017
00011D88 AB             <1>       stosd
                        <1>
                        <1> ; 2nd buffer:
                        <1>
00011D89 89D0           <1>         mov  eax, edx ; Physical address of the 2nd half of DMA buffer
00011D8B 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> ;
00011D8C 89F0           <1>       mov    eax, esi ; DMA half buffer size
00011D8E D1E8           <1>       shr    eax, 1 ; count of 16 bit samples
                        <1>       ;or    eax, IOC+BUP
00011D90 0D00000080     <1>       or     eax, IOC ; 11/06/2017
00011D95 AB             <1>       stosd
                        <1>
00011D96 E2DD           <1>       loop    s_ac97_bdl0
                        <1>
00011D98 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>
00011D99 668B15[BC650100]   <1>       mov          dx, [NAMBAR]
00011DA0 6683C22A           <1>       add          dx, CODEC_EXT_AUDIO_CTRL_REG     ; 2Ah
00011DA4 66ED               <1>       in           ax, dx
00011DA6 6683C801           <1>       or     ax, 1
00011DAA 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>
00011DAC 66A1[B6650100]     <1>       mov    ax, [audio_freq]          ; sample rate
                        <1>
00011DB2 668B15[BC650100]   <1>       mov          dx, [NAMBAR]
00011DB9 6683C22C           <1>       add          dx, CODEC_PCM_FRONT_DACRATE_REG  ; 2Ch
00011DBD 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> ;
00011DBF 668B15[BE650100]   <1>         mov     dx, [NABMBAR]
00011DC6 6683C21B           <1>         add     dx, PO_CR_REG                 ; set pointer to Cntl reg
00011DCA B002               <1>         mov     al, RR                        ; set reset
00011DCC 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> ;
00011DCD B8[C0650100]       <1>         mov  eax, audio_bdl_buff
00011DD2 668B15[BE650100]   <1>       mov    dx, [NABMBAR]
00011DD9 6683C210           <1>       add    dx, PO_BDBAR_REG
00011DDD EF                 <1>       out    dx, eax
                        <1> ;
                        <1> ; All set.  Let's play some music.
```

```
2236                                <1> ;
2237                                <1> ;
2238 00011DDE B81F000000          <1>       mov    eax, 31
2239 00011DE3 E816000000          <1>       call    set_ac97_LastValidIndex
2240                                <1>
2241 00011DE8 C605[B8650100]01    <1>       mov    byte [audio_play_cmd], 1 ; play command (do not stop) !
2242                                <1>
2243                                <1> ac97_play: ; continue to play (after pause)
2244                                <1>       ; 11/06/2017
2245                                <1>       ; 29/05/2017
2246                                <1>       ; 28/05/2017
2247 00011DEF 668B15[BE650100]    <1>         mov     dx, [NABMBAR]
2248 00011DF6 6683C21B            <1>         add     dx, PO_CR_REG          ; PCM out control register
2249 00011DFA B011                <1>         mov  al, IOCE+RPBM ; 29/05/2017
2250                                <1>         ;mov al, 1Dh ; (Ref: KolibriOS, intelac97.asm, 'play:')
2251 00011DFC EE                  <1>       out     dx, al                   ; set start!
2252                                <1>
2253                                <1>       ;mov   byte [audio_play_cmd], 1 ; play command (do not stop) !
2254                                <1>
2255 00011DFD C3                  <1>       retn
2256                                <1>
2257                                <1> ;input AL = index # to stop on
2258                                <1> set_ac97_LastValidIndex:
2259                                <1>       ; 28/05/2017
2260                                <1>       ; Derived from 'setLastValidIndex' procedure in 'ICHWAV.ASM'
2261                                <1>       ; .wav player for DOS by Jeff Leyda (02/09/2002)
2262 00011DFE 668B15[BE650100]    <1>       mov    dx, [NABMBAR]
2263 00011E05 6683C215            <1>       add    dx, PO_LVI_REG
2264 00011E09 EE                  <1>         out     dx, al
2265                                <1>       ;mov   [audio_lvi], al ; for ac97_int_handler
2266 00011E0A C3                  <1>       retn
2267                                <1>
2268                                <1> ac97_volume:
2269                                <1>       ; 28/05/2017
2270                                <1>       ; bl = component (0 = master/playback/lineout volume)
2271                                <1>       ; cl = left channel volume level (0 to 31)
2272                                <1>       ; ch = right channel volume level (0 to 31)
2273                                <1>
2274 00011E0B 08DB                <1>       or     bl, bl
2275 00011E0D 7523                <1>       jnz    short ac97_vol_1 ; temporary !
2276 00011E0F 66B81F1F            <1>       mov    ax, 1F1Fh ; 31,31
2277 00011E13 38C1                <1>       cmp    cl, al
2278 00011E15 771B                <1>       ja     short ac97_vol_1 ; temporary !
2279 00011E17 38E5                <1>       cmp    ch, ah
2280 00011E19 7717                <1>       ja     short ac97_vol_1 ; temporary !
2281 00011E1B 66890D[BA650100]    <1>       mov    [audio_master_volume], cx
2282 00011E22 6629C8              <1>       sub    ax, cx
2283 00011E25 668B15[BC650100]    <1>       mov     dx, [NAMBAR]
2284 00011E2C 6683C202            <1>       add     dx, CODEC_MASTER_VOL_REG  ; 02h ; Line Out
2285 00011E30 66EF                <1>       out     dx, ax
2286                                <1> ac97_vol_1:
2287 00011E32 C3                  <1>       retn
2288                                <1>
2289                                <1> ac97_int_handler:
2290                                <1>       ; 12/10/2017
2291                                <1>       ; 10/10/2017
2292                                <1>       ; 09/10/2017
2293                                <1>       ; 13/06/2017, 13/06/2017
2294                                <1>       ; 10/06/2017, 11/06/2017
2295                                <1>       ; Interrupt Handler for AC97 (ICH) Audio Controller
2296                                <1>       ; Note: called by 'dev_IRQ_service'
2297                                <1>       ; 28/05/2017
2298                                <1>
2299                                <1>       ;push  eax ; * must be saved !
2300                                <1>       ;push  edx
2301                                <1>       ;push  ecx
2302                                <1>       ;push  ebx ; * must be saved !
2303                                <1>       ;push  esi
2304                                <1>       ;push  edi
2305                                <1>
2306                                <1>       ;cmp   byte [audio_busy], 1
2307                                <1>       ;jnb   _ac97_ih2    ; busy !
2308                                <1>
2309 00011E33 66BA3000            <1>         mov     dx, GLOB_STS_REG
2310 00011E37 660315[BE650100]    <1>         add  dx, [NABMBAR]
2311 00011E3E ED                  <1>       in     eax, dx
2312                                <1>
2313 00011E3F 83F8FF              <1>         cmp     eax, 0FFFFFFFFh ; -1
2314 00011E42 0F849A000000        <1>         je      _ac97_ih3 ; exit
2315                                <1>
2316 00011E48 A940000000          <1>         test    eax, 40h ; PCM Out Interrupt
2317 00011E4D 750E                <1>         jnz     short _ac97_ih0
2318                                <1>
2319 00011E4F 85C0                <1>       test   eax, eax
2320 00011E51 0F848B000000        <1>       jz     _ac97_ih3 ; exit
2321                                <1>
2322                                <1>       ;mov   dx, GLOB_STS_REG
2323                                <1>         ;add dx, [NABMBAR]
2324 00011E57 EF                  <1>       out    dx, eax
2325                                <1>
2326 00011E58 E985000000          <1>       jmp    _ac97_ih3 ; exit
2327                                <1>
2328                                <1> _ac97_ih0:
2329 00011E5D 50                  <1>       push   eax
2330                                <1>       ; 09/10/2017
2331 00011E5E 803D[B8650100]01    <1>       cmp    byte [audio_play_cmd], 1
2332 00011E65 727C                <1>       jb     short _ac97_ih4 ; stop command !
2333                                <1>
2334                                <1>       ;mov   byte [audio_busy], 1
2335                                <1>
2336                                <1>       ;mov   al, 10h
2337                                <1>       ;mov   dx, PO_CR_REG
```

```
2338                                    <1>         ;add dx, [NABMBAR]
2339                                    <1>       ;out   dx, al
2340                                    <1>
2341 00011E67 66B81C00                 <1>       mov    ax, 1Ch ; FIFOE(=16)+BCIS(=8)+LVBCI(=4)
2342 00011E6B 66BA1600                 <1>       mov    dx, PO_SR_REG
2343 00011E6F 660315[BE650100]         <1>         add  dx, [NABMBAR]
2344 00011E76 66EF                     <1>       out    dx, ax
2345                                    <1>
2346 00011E78 66BA1400                 <1>       mov    dx, PO_CIV_REG
2347 00011E7C 660315[BE650100]         <1>         add  dx, [NABMBAR]
2348 00011E83 EC                       <1>       in     al, dx
2349                                    <1>
2350                                    <1>       ;cmp   al, [audio_civ] ; [audio_flag]
2351                                    <1>       ;je    short _ac97_ih2
2352                                    <1>
2353 00011E84 A2[B9650100]             <1>       mov    [audio_civ], al
2354 00011E89 FEC8                     <1>       dec    al
2355                                    <1>       ;inc   al ; 11/06/2017
2356 00011E8B 241F                     <1>       and    al, 1Fh
2357                                    <1>
2358 00011E8D 66BA1500                 <1>         mov     dx, PO_LVI_REG
2359 00011E91 660315[BE650100]         <1>         add  dx, [NABMBAR]
2360 00011E98 EE                       <1>         out  dx, al
2361                                    <1>
2362                                    <1>       ; 12/10/2017
2363 00011E99 A0[B9650100]             <1>       mov    al, [audio_civ]
2364 00011E9E FEC0                     <1>       inc    al
2365 00011EA0 2401                     <1>       and    al, 1
2366 00011EA2 A2[AC650100]             <1>       mov    [audio_flag], al
2367                                    <1>       ;; [audio_flag] : 0 = Buffer 1, 1 = Buffer 2
2368                                    <1>       ;
2369 00011EA7 58                       <1>       pop    eax
2370                                    <1>       ;
2371 00011EA8 83E040                   <1>       and     eax, 40h
2372 00011EAB 668B15[BE650100]         <1>         mov  dx, [NABMBAR]
2373 00011EB2 6683C230                 <1>       add    dx, GLOB_STS_REG
2374 00011EB6 EF                       <1>       out    dx, eax
2375                                    <1>
2376                                    <1>       ;; 13/06/2017
2377                                    <1>       ;mov   al, 11h ; IOCE + RPBM
2378                                    <1>       ;mov   dx, PO_CR_REG
2379                                    <1>         ;add dx, [NABMBAR]
2380                                    <1>       ;out   dx, al
2381                                    <1>
2382                                    <1> ac97_tuneloop:
2383                                    <1>       ; 09/10/2017
2384 00011EB7 8B3D[A4650100]           <1>       mov    edi, [audio_dma_buff]
2385 00011EBD 8B0D[A8650100]           <1>       mov    ecx, [audio_dmabuff_size]
2386 00011EC3 D1E9                     <1>       shr    ecx, 1 ; dma buff size / 2 = half buffer size
2387                                    <1>
2388                                    <1>       ; 12/10/2017
2389 00011EC5 803D[AC650100]00         <1>       cmp    byte [audio_flag], 0
2390 00011ECC 7702                     <1>       ja     short _ac97_ih1  ; Playing Half Buffer 2 (Current: FLAG)
2391                                    <1>       ; Playing Half Buffer 1 (Current: EOL)
2392 00011ECE 01CF                     <1>       add    edi, ecx
2393                                    <1> _ac97_ih1:
2394                                    <1>       ; Update half buffer 2 while playing half buffer 1 (next: FLAG)
2395                                    <1>       ; Update half buffer 1 while playing half buffer 2 (next: EOL)
2396                                    <1>
2397 00011ED0 8B35[9C650100]           <1>       mov    esi, [audio_p_buffer] ; phy addr of audio buff
2398 00011ED6 C1E902                   <1>       shr    ecx, 2 ; half buff size / 4
2399 00011ED9 F3A5                     <1>       rep    movsd
2400                                    <1>
2401                                    <1>       ; 10/10/2017
2402                                    <1>       ; switch flag value
2403 00011EDB 8035[AC650100]01         <1>       xor    byte [audio_flag], 1
2404                                    <1>       ; 12/10/2017
2405                                    <1>       ; [audio_flag] = 0 : Playing dma half buffer 2 (even index value)
2406                                    <1>                          ; Next buffer (to update) is dma half buff 1
2407                                    <1>       ;             = 1 : Playing dma half buffer 1 (odd index value)
2408                                    <1>                          ; Next buffer (to update) is dma half buff 2
2409                                    <1>
2410                                    <1> _ac97_ih2:
2411                                    <1>       ;mov   byte [audio_busy], 0
2412                                    <1> _ac97_ih3:
2413                                    <1>       ;pop   edi
2414                                    <1>       ;pop   esi
2415                                    <1>       ;pop   ebx ; * must be restored !
2416                                    <1>       ;pop   ecx
2417                                    <1>       ;pop   edx
2418                                    <1>       ;pop   eax ; * must be restored !
2419                                    <1>
2420 00011EE2 C3                       <1>       retn
2421                                    <1>
2422                                    <1> _ac97_ih4:
2423                                    <1>       ; 09/10/2017
2424 00011EE3 E818000000               <1>       call   _ac97_stop
2425                                    <1>       ;
2426 00011EE8 58                       <1>       pop    eax
2427                                    <1>       ;
2428 00011EE9 83E040                   <1>       and     eax, 40h
2429 00011EEC 668B15[BE650100]         <1>         mov  dx, [NABMBAR]
2430 00011EF3 6683C230                 <1>       add    dx, GLOB_STS_REG
2431 00011EF7 EF                       <1>       out    dx, eax
2432                                    <1>
2433                                    <1>       ;; 13/06/2017
2434                                    <1>       ;mov   al, 11h ; IOCE + RPBM
2435                                    <1>       ;dx, PO_CR_REG
2436                                    <1>         ;add dx, [NABMBAR]
2437                                    <1>       ;out   dx, al
2438                                    <1>
2439                                    <1>       ; 10/10/2017
```

```
2440                                    <1>       ;jmp   short _ac97_ih3  ; exit
2441 00011EF8 C3                       <1>       retn
2442                                    <1>
2443                                    <1> ac97_stop:
2444                                    <1>       ; 28/05/2017
2445 00011EF9 C605[B8650100]00          <1>       mov    byte [audio_play_cmd], 0 ; stop !
2446                                    <1> _ac97_stop: ; 09/10/2017
2447                                    <1>       ; 29/05/2017
2448                                    <1>       ;mov   dx, [NABMBAR]
2449                                    <1>       ;add   dx, PO_CR_REG
2450                                    <1>       ;mov   al, 0
2451                                    <1>       ;out   dx, al
2452                                    <1>
2453                                    <1>       ; 11/06/2017
2454 00011F00 30C0                     <1>       xor    al, al ; 0
2455 00011F02 E813000000               <1>       call   ac97_po_cmd
2456                                    <1>
2457                                    <1>       ; (Ref: KolibriOS, intelac97.asm, 'stop:')
2458                                    <1>       ; Clear FIFOE, BCIS, LVBCI (Ref: Intel ICH hub manual)
2459 00011F07 66B81C00                 <1>       mov     ax, 1Ch
2460 00011F0B 668B15[BE650100]         <1>       mov     dx, [NABMBAR]
2461 00011F12 6683C216                 <1>       add     dx, PO_SR_REG
2462 00011F16 66EF                     <1>       out     dx, ax
2463                                    <1>
2464                                    <1>       ;retn
2465                                    <1>
2466                                    <1>       ; 11/06/2017
2467 00011F18 B002                     <1>       mov     al, RR
2468                                    <1> ac97_po_cmd:
2469                                    <1>        ;11/06/2017
2470                                    <1>       ; 29/05/2017
2471 00011F1A 668B15[BE650100]         <1>       mov     dx, [NABMBAR]
2472 00011F21 6683C21B                 <1>         add     dx, PO_CR_REG          ; PCM out control register
2473 00011F25 EE                       <1>       out    dx, al
2474 00011F26 C3                       <1>       retn
2475                                    <1>
2476                                    <1> ac97_pause:
2477                                    <1>       ; 11/06/2017
2478                                    <1>       ; 29/05/2017
2479 00011F27 B010                     <1>       mov    al, IOCE
2480 00011F29 EBEF                     <1>       jmp    short ac97_po_cmd
2481                                    <1>
2482                                    <1> reset_ac97_controller:
2483                                    <1>       ; 10/06/2017
2484                                    <1>       ; 29/05/2017
2485                                    <1>       ; 28/05/2017
2486                                    <1>       ; reset AC97 audio controller registers
2487 00011F2B 31C0                     <1>       xor     eax, eax
2488 00011F2D 66BA0B00                 <1>         mov  dx, PI_CR_REG
2489 00011F31 660315[BE650100]         <1>       add    dx, [NABMBAR]
2490 00011F38 EE                       <1>       out     dx, al
2491                                    <1>
2492 00011F39 66BA1B00                 <1>         mov     dx, PO_CR_REG
2493 00011F3D 660315[BE650100]         <1>       add    dx, [NABMBAR]
2494 00011F44 EE                       <1>       out     dx, al
2495                                    <1>
2496 00011F45 66BA2B00                 <1>         mov     dx, MC_CR_REG
2497 00011F49 660315[BE650100]         <1>       add    dx, [NABMBAR]
2498 00011F50 EE                       <1>       out     dx, al
2499                                    <1>
2500 00011F51 B002                     <1>         mov     al, RR
2501 00011F53 66BA0B00                 <1>         mov     dx, PI_CR_REG
2502 00011F57 660315[BE650100]         <1>       add    dx, [NABMBAR]
2503 00011F5E EE                       <1>       out     dx, al
2504                                    <1>
2505 00011F5F 66BA1B00                 <1>         mov     dx, PO_CR_REG
2506 00011F63 660315[BE650100]         <1>       add    dx, [NABMBAR]
2507 00011F6A EE                       <1>       out     dx, al
2508                                    <1>
2509 00011F6B 66BA2B00                 <1>         mov     dx, MC_CR_REG
2510 00011F6F 660315[BE650100]         <1>       add    dx, [NABMBAR]
2511 00011F76 EE                       <1>       out     dx, al
2512                                    <1>
2513 00011F77 C3                       <1>       retn
2514                                    <1>
2515                                    <1> ac97_reset:
2516                                    <1>       ; 10/06/2017
2517                                    <1>       ; 29/05/2017
2518                                    <1>       ; 28/05/2017
2519 00011F78 E8AEFFFFFF               <1>       call   reset_ac97_controller
2520                                    <1>       ; 29/05/2017
2521                                    <1>       ;jmp   reset_ac97_codec
2522                                    <1> reset_ac97_codec:
2523                                    <1>       ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, intelac97.asm)
2524 00011F7D 66BA2C00                 <1>       mov    dx, GLOB_CNT_REG ; 2Ch
2525 00011F81 660315[BE650100]         <1>       add    dx, [NABMBAR]
2526 00011F88 ED                       <1>       in     eax, dx
2527                                    <1>
2528 00011F89 A902000000               <1>       test   eax, 2
2529 00011F8E 7407                     <1>       jz     short _r_ac97codec_cold
2530                                    <1>
2531 00011F90 E80F000000               <1>       call   warm_ac97codec_reset
2532 00011F95 7308                     <1>       jnc    short _r_ac97codec_ok
2533                                    <1> _r_ac97codec_cold:
2534 00011F97 E83D000000               <1>         call    cold_ac97codec_reset
2535 00011F9C 7301                     <1>         jnc     short _r_ac97codec_ok
2536                                    <1>
2537                                    <1>       ; 16/04/2017
2538                                    <1>         ;xor     eax, eax         ; timeout error
2539                                    <1>             ;stc
2540 00011F9E C3                       <1>       retn
2541                                    <1>
```

```
2542                                    <1> _r_ac97codec_ok:
2543 00011F9F 31C0                     <1>         xor     eax, eax
2544                                    <1>         ;mov al, VIA_ACLINK_C00_READY ; 1
2545 00011FA1 FEC0                     <1>         inc  al
2546 00011FA3 C3                       <1>       retn
2547                                    <1>
2548                                    <1> warm_ac97codec_reset:
2549                                    <1>       ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, intelac97.asm)
2550 00011FA4 B806000000               <1>         mov     eax, 6
2551 00011FA9 66BA2C00                 <1>       mov    dx, GLOB_CNT_REG ; 2Ch
2552 00011FAD 660315[BE650100]         <1>       add    dx, [NABMBAR]
2553 00011FB4 EF                       <1>       out    dx, eax
2554                                    <1>
2555 00011FB5 B90A000000               <1>       mov    ecx, 10      ; total 1s
2556                                    <1> _warm_ac97c_rst_wait:
2557 00011FBA 51                       <1>       push   ecx
2558 00011FBB E8D6F5FFFF               <1>       call   delay_100ms
2559 00011FC0 59                       <1>       pop    ecx
2560                                    <1>
2561 00011FC1 66BA3000                 <1>       mov    dx, GLOB_STS_REG ; 30h
2562 00011FC5 660315[BE650100]         <1>       add    dx, [NABMBAR]
2563 00011FCC ED                       <1>       in     eax, dx
2564                                    <1>
2565 00011FCD A900030010               <1>       test   eax, CTRL_ST_CREADY
2566 00011FD2 7504                     <1>       jnz    short _warm_ac97c_rst_ok
2567                                    <1>
2568 00011FD4 49                       <1>         dec     ecx
2569 00011FD5 75E3                     <1>         jnz     short _warm_ac97c_rst_wait
2570                                    <1>
2571                                    <1> _warm_ac97c_rst_fail:
2572 00011FD7 F9                       <1>         stc
2573                                    <1> _warm_ac97c_rst_ok:
2574 00011FD8 C3                       <1>       retn
2575                                    <1>
2576                                    <1> cold_ac97codec_reset:
2577                                    <1>       ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, intelac97.asm)
2578 00011FD9 B802000000               <1>         mov     eax, 2
2579 00011FDE 66BA2C00                 <1>       mov    dx, GLOB_CNT_REG ; 2Ch
2580 00011FE2 660315[BE650100]         <1>       add    dx, [NABMBAR]
2581 00011FE9 EF                       <1>       out    dx, eax
2582                                    <1>
2583 00011FEA E8A7F5FFFF               <1>       call   delay_100ms  ; wait 100 ms
2584 00011FEF E8A2F5FFFF               <1>       call   delay_100ms  ; wait 100 ms
2585 00011FF4 E89DF5FFFF               <1>       call   delay_100ms  ; wait 100 ms
2586 00011FF9 E898F5FFFF               <1>       call   delay_100ms  ; wait 100 ms
2587                                    <1>
2588 00011FFE B910000000               <1>       mov    ecx, 16      ; total 20*100 ms = 2s
2589                                    <1> _cold_ac97c_rst_wait:
2590 00012003 66BA3000                 <1>       mov    dx, GLOB_STS_REG ; 30h
2591 00012007 660315[BE650100]         <1>       add    dx, [NABMBAR]
2592 0001200E ED                       <1>       in     eax, dx
2593                                    <1>
2594 0001200F A900030010               <1>       test   eax, CTRL_ST_CREADY
2595 00012014 750B                     <1>       jnz    short _cold_ac97c_rst_ok
2596                                    <1>
2597 00012016 51                       <1>       push   ecx
2598 00012017 E87AF5FFFF               <1>       call   delay_100ms
2599 0001201C 59                       <1>       pop    ecx
2600                                    <1>
2601 0001201D 49                       <1>         dec     ecx
2602 0001201E 75E3                     <1>         jnz     short _cold_ac97c_rst_wait
2603                                    <1>
2604                                    <1> _cold_ac97c_rst_fail:
2605 00012020 F9                       <1>         stc
2606                                    <1> _cold_ac97c_rst_ok:
2607 00012021 C3                       <1>       retn
2608                                    <1>
2609                                    <1> sb16_current_sound_data:
2610                                    <1>       ; 20/08/2017
2611                                    <1>       ; 24/06/2017
2612                                    <1>       ; 22/06/2017
2613                                    <1>       ; get current sound (PCM out) data for graphics
2614                                    <1>       ; (for Sound Blaster 16)
2615                                    <1>       ; ebx = Physical address (on page boundary)
2616                                    <1>       ; ecx = Byte count
2617                                    <1>       ; [audio_buff_size]
2618                                    <1>
2619                                    <1>       ;;mov  edi, [audio_buff_size]
2620                                    <1>       ;mov   edi, [audio_dmabuff_size]
2621                                    <1>       ;mov   esi, [audio_dma_buff]
2622 00012022 39CF                     <1>       cmp    edi, ecx
2623 00012024 7302                     <1>       jnb    short sb16_gcd_0
2624 00012026 89F9                     <1>       mov    ecx, edi
2625                                    <1> sb16_gcd_0:
2626                                    <1>       ; 20/08/2017
2627 00012028 803D[B4650100]10         <1>       cmp    byte [audio_bps], 16
2628 0001202F 750F                     <1>       jne    short sb16_gcd_1 ; 8 bit DMA channel
2629 00012031 E4C6                     <1>       in     al, 0C6h ; DMA channel 5 count register
2630 00012033 88C2                     <1>       mov    dl, al
2631 00012035 E4C6                     <1>       in     al, 0C6h
2632 00012037 88C6                     <1>       mov     dh, al
2633 00012039 0FB7C2                   <1>       movzx  eax, dx
2634 0001203C D1E0                     <1>       shl    eax, 1 ; word count -> byte count
2635 0001203E EB4E                     <1>       jmp    short sb16_gcd_2
2636                                    <1> sb16_gcd_1:
2637 00012040 E403                     <1>       in     al, 03h ; DMA channel 1 count register
2638 00012042 88C2                     <1>       mov    dl, al
2639 00012044 E403                     <1>       in     al, 03h
2640 00012046 88C6                     <1>       mov     dh, al
2641 00012048 0FB7C2                   <1>       movzx  eax, dx
2642 0001204B EB41                     <1>       jmp    short sb16_gcd_2
2643                                    <1> ;sb16_gcd_2:
```

```
2644                                    <1> ;       cmp     eax, ecx
2645                                    <1> ;       jnb     short sb16_gcd_3
2646                                    <1> ;       ; remain count < graphics bytes
2647                                    <1> ;       mov     eax, ecx ; fix remain count to data size
2648                                    <1> ;sb16_gcd_3:
2649                                    <1> ;       sub     edi, eax
2650                                    <1> ;       jna     short sb16_gcd_4
2651                                    <1> ;       add     esi, edi ; dma buffer offset
2652                                    <1> ;sb16_gcd_4:
2653                                    <1> ;       mov     edi, ebx ; buffer address (for graphics)
2654                                    <1> ;       mov     [u.r0], ecx
2655                                    <1> ;       rep     movsb
2656                                    <1> ;       retn
2657                                    <1>
2658                                    <1> get_current_sound_data:
2659                                    <1>         ; 24/06/2017
2660                                    <1>         ; 22/06/2017
2661                                    <1>         ; get current sound (PCM out) data for graphics
2662                                    <1>         ;
2663                                    <1>         ; ebx = Physical address (on page boundary)
2664                                    <1>         ; ecx = Byte count
2665                                    <1>         ; [audio_buff_size]
2666                                    <1>
2667                                    <1>         ;mov   edi, [audio_buff_size]
2668 0001204D 8B3D[A8650100]           <1>         mov     edi, [audio_dmabuff_size]
2669 00012053 8B35[A4650100]           <1>         mov     esi, [audio_dma_buff]
2670 00012059 803D[85650100]02         <1>         cmp     byte [audio_device], 2
2671 00012060 72C0                     <1>         jb      short sb16_current_sound_data ; = 1
2672 00012062 D1EF                     <1>         shr     edi, 1
2673 00012064 39CF                     <1>         cmp     edi, ecx
2674 00012066 7302                     <1>         jnb     short gcd_0
2675 00012068 89F9                     <1>         mov     ecx, edi
2676                                    <1> gcd_0:
2677 0001206A 803D[85650100]03         <1>         cmp     byte [audio_device], 3
2678 00012071 7232                     <1>         jb      short ac97_current_sound_data ; = 2
2679                                    <1>         ; = 3
2680                                    <1> vt8233_current_sound_data:
2681                                    <1>         ; 22/06/2017
2682                                    <1>         ; 21/06/2017
2683                                    <1>         ; get current sound (PCM out) data for graphics
2684                                    <1>         ; (for VT 8233, VT 8237R)
2685                                    <1>         ; ebx = Physical address (on page boundary)
2686                                    <1>         ; ecx = Byte count
2687                                    <1>         ; [audio_buff_size]
2688                                    <1>
2689                                    <1>         ;;mov  edi, [audio_buff_size]
2690                                    <1>         ;mov   edi, [audio_dmabuff_size]
2691                                    <1>         ;mov   esi, [audio_dma_buff]
2692                                    <1>         ;shr   edi, 1
2693                                    <1>         ;cmp   edi, ecx
2694                                    <1>         ;jnb   short vt8233_gcd_1
2695                                    <1>         ;mov   ecx, edi
2696                                    <1> vt8233_gcd_1:
2697 00012073 BA0C000000               <1>         mov     edx, VIA_REG_OFFSET_CURR_COUNT
2698 00012078 E879F5FFFF               <1>         call    ctrl_io_r32
2699 0001207D 89C2                     <1>         mov     edx, eax ; remain count (bits 23-0),
2700                                    <1>                          ; SGD index (bits 31-24)
2701 0001207F 81E200000001             <1>         and     edx, 1000000h ; SGD index (0 = 1st half)
2702 00012085 7402                     <1>         jz      short vt8233_gcd_2
2703                                    <1>         ; the second half of DMA buffer
2704 00012087 01FE                     <1>         add     esi, edi
2705                                    <1> vt8233_gcd_2:
2706 00012089 25FFFFFF00               <1>         and     eax, 0FFFFFFh ; bits 23-0
2707                                    <1> ac97_gcd_2:
2708                                    <1> sb16_gcd_2:
2709 0001208E 39C8                     <1>         cmp     eax, ecx
2710 00012090 7302                     <1>         jnb     short vt8233_gcd_3
2711                                    <1>         ; remain count < graphics bytes
2712 00012092 89C8                     <1>         mov     eax, ecx ; fix remain count to data size
2713                                    <1> vt8233_gcd_3:
2714 00012094 29C7                     <1>         sub     edi, eax
2715 00012096 7602                     <1>         jna     short vt8233_gcd_4
2716 00012098 01FE                     <1>         add     esi, edi ; dma buffer offset
2717                                    <1> vt8233_gcd_4:
2718 0001209A 89DF                     <1>         mov     edi, ebx ; buffer address (for graphics)
2719 0001209C 890D[64030300]           <1>         mov     [u.r0], ecx
2720 000120A2 F3A4                     <1>         rep     movsb
2721                                    <1> vt8233_gcd_5:
2722 000120A4 C3                       <1>         retn
2723                                    <1>
2724                                    <1> ac97_current_sound_data:
2725                                    <1>         ; 23/06/2017
2726                                    <1>         ; 22/06/2017
2727                                    <1>         ; get current sound (PCM out) data for graphics
2728                                    <1>         ; (for AC'97, ICH)
2729                                    <1>         ; ebx = Physical address (on page boundary)
2730                                    <1>         ; ecx = Byte count
2731                                    <1>         ; [audio_buff_size]
2732                                    <1>
2733                                    <1>         ;;mov  edi, [audio_buff_size]
2734                                    <1>         ;mov   edi, [audio_dmabuff_size]
2735                                    <1>         ;mov   esi, [audio_dma_buff]
2736                                    <1>         ;shr   edi, 1
2737                                    <1>         ;cmp   edi, ecx
2738                                    <1>         ;jnb   short ac97_gcd_0
2739                                    <1>         ;mov   ecx, edi
2740                                    <1> ac97_gcd_0:
2741 000120A5 66BA1400                 <1>         mov     dx, PO_CIV_REG ; Position In Current Buff Reg
2742 000120A9 660315[BE650100]         <1>         add     dx, [NABMBAR]
2743 000120B0 EC                       <1>         in      al, dx ; current index value
2744 000120B1 A801                     <1>         test    al, 1
2745 000120B3 7402                     <1>         jz      short ac97_gcd_1
```

```
2746 000120B5 01FE                <1>       add    esi, edi
2747                              <1> ac97_gcd_1:
2748 000120B7 31C0                <1>       xor    eax, eax
2749 000120B9 66BA1800            <1>       mov    dx, PO_PICB_REG ; Position In Current Buff Reg
2750 000120BD 660315[BE650100]    <1>       add    dx, [NABMBAR]
2751 000120C4 66ED                <1>       in     ax, dx ; remain dwords
2752 000120C6 C1E002              <1>       shl    eax, 2 ; remain bytes ; 23/06/2017
2753 000120C9 EBC3                <1>       jmp    short ac97_gcd_2
2754                              <1> ;     cmp    eax, ecx
2755                              <1> ;     jnb    short ac97_gcd_2
2756                              <1> ;     ; remain count < graphics bytes
2757                              <1> ;     mov    eax, ecx ; fix remain count to data size
2758                              <1> ;ac97_gcd_2:
2759                              <1> ;     sub    edi, eax
2760                              <1> ;     jna    short ac97_gcd_3
2761                              <1> ;     add    esi, edi ; dma buffer offset
2762                              <1> ;ac97_gcd_3:
2763                              <1> ;     mov    edi, ebx ; buffer address (for graphics)
2764                              <1> ;     mov    [u.r0], ecx
2765                              <1> ;     rep    movsb
2766                              <1> ;     retn
2767                              <1>
2768                              <1> sb16_get_dma_buff_off:
2769                              <1>       ; 24/06/2017
2770                              <1>       ; 22/06/2017
2771                              <1>       ; get current (PCM OUT DMA buffer) pointer
2772                              <1>       ; (for Sound Blaster 16)
2773                              <1>
2774                              <1>       ;mov   ecx, [audio_dmabuff_size]
2775                              <1>       ;xor   ebx, ebx
2776                              <1>       ;shr   ecx, 1
2777                              <1> sb16_gdmabo_0:
2778 000120CB E403                <1>       in     al, 03h
2779 000120CD 88C2                <1>       mov    dl, al
2780 000120CF E403                <1>       in     al, 03h
2781 000120D1 88C6                <1>       mov     dh, al
2782 000120D3 0FB7C2              <1>       movzx  eax, dx
2783 000120D6 EB30                <1>       jmp    short sb16_gdmabo_1
2784                              <1>
2785                              <1> get_dma_buffer_offset:
2786                              <1>       ; 24/06/2017
2787                              <1>       ; 22/06/2017
2788                              <1>       ; get current sound (PCM out) data for graphics
2789                              <1>       ;
2790                              <1>       ; ebx = Physical address (on page boundary)
2791                              <1>       ; ecx = Byte count
2792                              <1>       ; [audio_buff_size]
2793                              <1>
2794 000120D8 8B0D[A8650100]      <1>       mov    ecx, [audio_dmabuff_size]
2795 000120DE 31DB                <1>       xor    ebx, ebx
2796                              <1> gdmabo_0:
2797 000120E0 803D[85650100]02    <1>       cmp    byte [audio_device], 2
2798 000120E7 72E2                <1>       jb     short sb16_get_dma_buff_off
2799 000120E9 742A                <1>       je     short ac97_get_dma_buff_off
2800                              <1>
2801                              <1> vt8233_get_dma_buff_off:
2802                              <1>       ; 24/06/2017
2803                              <1>       ; 22/06/2017
2804                              <1>       ; get current (PCM OUT DMA buffer) pointer
2805                              <1>       ; (for VT 8233, VT 8237R)
2806                              <1>
2807                              <1>       ;mov   ecx, [audio_dmabuff_size]
2808                              <1>       ;xor   ebx, ebx
2809 000120EB D1E9                <1>       shr    ecx, 1
2810                              <1> vt8233_gdmabo_0:
2811 000120ED BA0C000000          <1>       mov    edx, VIA_REG_OFFSET_CURR_COUNT
2812 000120F2 E8FFF4FFFF          <1>       call   ctrl_io_r32
2813 000120F7 89C2                <1>       mov    edx, eax ; remain count (bits 23-0),
2814                              <1>                       ; SGD index (bits 31-24)
2815 000120F9 81E200000001        <1>       and    edx, 1000000h ; SGD index (0 = 1st half)
2816 000120FF 7402                <1>       jz     short vt8233_gdmabo_1
2817                              <1>       ; the second half of DMA buffer
2818 00012101 89CB                <1>       mov    ebx, ecx
2819                              <1> vt8233_gdmabo_1:
2820 00012103 25FFFFFF00          <1>       and    eax, 0FFFFFFh ; bits 23-0
2821                              <1> sb16_gdmabo_1:
2822                              <1> ac97_gdmabo_2:
2823 00012108 29C1                <1>       sub    ecx, eax
2824 0001210A 7602                <1>       jna    short vt8233_gdmabo_2
2825 0001210C 01CB                <1>       add    ebx, ecx ; dma buffer offset
2826                              <1> vt8233_gdmabo_2:
2827 0001210E 891D[64030300]      <1>       mov    [u.r0], ebx
2828 00012114 C3                  <1>       retn
2829                              <1>
2830                              <1> ac97_get_dma_buff_off:
2831                              <1>       ; 24/06/2017
2832                              <1>       ; 22/06/2017
2833                              <1>       ; get current (PCM OUT DMA buffer) pointer
2834                              <1>       ; (for AC'97, ICH)
2835                              <1>       ; ebx = Physical address (on page boundary)
2836                              <1>       ; ecx = Byte count
2837                              <1>       ; [audio_buff_size]
2838                              <1>
2839                              <1>       ;mov   ecx, [audio_dmabuff_size]
2840                              <1>       ;xor   ebx, ebx
2841 00012115 D1E9                <1>       shr    ecx, 1
2842                              <1> ac97_gdmabo_0:
2843 00012117 66BA1400            <1>       mov    dx, PO_CIV_REG ; Position In Current Buff Reg
2844 0001211B 660315[BE650100]    <1>       add    dx, [NABMBAR]
2845 00012122 EC                  <1>       in     al, dx ; current index value
2846 00012123 A801                <1>       test   al, 1
2847 00012125 7402                <1>       jz     short ac97_gdmabo_1
```

```
  2848 00012127 89CB                <1>       mov    ebx, ecx
  2849                              <1> ac97_gdmabo_1:
  2850 00012129 31C0                <1>       xor    eax, eax
  2851 0001212B 66BA1800            <1>       mov    dx, PO_PICB_REG ; Position In Current Buff Reg
  2852 0001212F 660315[BE650100]    <1>       add    dx, [NABMBAR]
  2853 00012136 66ED                <1>       in     ax, dx ; remain dwords
  2854 00012138 EBCE                <1>       jmp    short ac97_gdmabo_2
  2641
  2642 0001213A 90<rept>                align 4
  2643
  2644                                  %include 'vgadata.s' ; 04/07/2016
    1                               <1> ;
********************************************************************************************
     2                              <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - vgadata.s  (palette and fond data)
     3                              <1> ; ----------------------------------------------------------------------------
     4                              <1> ; Last Update: 04/07/2016
     5                              <1> ; ----------------------------------------------------------------------------
     6                              <1> ; Beginning: 16/01/2016
     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 '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> ; 04/07/2016
    25                              <1> ; COLOR DATA
    26                              <1>
    27                            <1> palette0:
    28 0001213C 00000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
    28 00012145 00000000000000      <1>
    29 0001214C 0000000000000002A- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
    29 00012155 2A2A2A2A2A2A2A      <1>
    30 0001215C 2A2A2A2A2A2A2A2A2A- <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
    30 00012165 2A2A2A2A2A2A2A      <1>
    31 0001216C 2A2A2A2A2A2A2A2A2A- <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
    31 00012175 2A2A2A2A2A2A2A      <1>
    32 0001217C 2A2A2A2A2A2A2A3F- <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh
    32 00012185 3F3F3F3F3F3F3F      <1>
    33 0001218C 3F3F3F3F3F3F3F3F3F- <1>     db  03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh
    33 00012195 3F3F3F3F3F3F3F      <1>
    34 0001219C 00000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
    34 000121A5 00000000000000      <1>
    35 000121AC 0000000000000002A- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
    35 000121B5 2A2A2A2A2A2A2A      <1>
    36 000121BC 2A2A2A2A2A2A2A2A2A- <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
    36 000121C5 2A2A2A2A2A2A2A      <1>
    37 000121CC 2A2A2A2A2A2A2A2A2A- <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
    37 000121D5 2A2A2A2A2A2A2A      <1>
    38 000121DC 2A2A2A2A2A2A2A3F- <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh
    38 000121E5 3F3F3F3F3F3F3F      <1>
    39 000121EC 3F3F3F3F3F3F3F3F3F- <1>     db  03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh
    39 000121F5 3F3F3F3F3F3F3F      <1>
    40                            <1> palette1:
    41 000121FC 00000000002A002A00- <1>     db  000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah
    41 00012205 002A2A2A00002A      <1>
    42 0001220C 002A2A15002A2A2A00- <1>     db  000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah
    42 00012215 000000002A002A      <1>
    43 0001221C 00002A2A2A00002A00- <1>     db  000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah, 000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah
    43 00012225 2A2A15002A2A2A      <1>
    44 0001222C 15151515153F153F15- <1>     db  015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh, 015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh
    44 00012235 153F3F3F15153F      <1>
    45 0001223C 153F3F3F153F3F3F15- <1>     db  015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh
    45 00012245 151515153F153F      <1>
    46 0001224C 15153F3F3F15153F15- <1>     db  015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh
    46 00012255 3F3F3F153F3F3F      <1>
    47 0001225C 000000000002A002A00- <1>     db  000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah
    47 00012265 002A2A2A00002A      <1>
    48 0001226C 002A2A15002A2A2A00- <1>     db  000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah
    48 00012275 000000002A002A      <1>
    49 0001227C 00002A2A2A00002A00- <1>     db  000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah, 000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah
    49 00012285 2A2A15002A2A2A      <1>
    50 0001228C 15151515153F153F15- <1>     db  015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh, 015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh
    50 00012295 153F3F3F15153F      <1>
    51 0001229C 153F3F3F153F3F3F15- <1>     db  015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh
    51 000122A5 151515153F153F      <1>
    52 000122AC 15153F3F3F15153F15- <1>     db  015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh
    52 000122B5 3F3F3F153F3F3F      <1>
    53                            <1> palette2:
    54 000122BC 00000000002A002A00- <1>     db  000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah
    54 000122C5 002A2A2A00002A      <1>
    55 000122CC 002A2A2A002A2A2A00- <1>     db  000h, 02ah, 02ah, 02ah, 000h, 02ah, 02ah, 02ah, 000h, 000h, 015h, 000h, 000h, 03fh, 000h, 02ah
    55 000122D5 001500003F002A      <1>
    56 000122DC 15002A3F2A00152A00- <1>     db  015h, 000h, 02ah, 03fh, 02ah, 000h, 015h, 02ah, 000h, 03fh, 02ah, 02ah, 015h, 02ah, 02ah, 03fh
    56 000122E5 3F2A2A152A2A3F      <1>
    57 000122EC 00150000152A003F00- <1>     db  000h, 015h, 000h, 000h, 015h, 02ah, 000h, 03fh, 000h, 000h, 03fh, 02ah, 02ah, 015h, 000h, 02ah
    57 000122F5 003F2A2A15002A      <1>
    58 000122FC 152A2A3F002A3F2A00- <1>     db  015h, 02ah, 02ah, 03fh, 000h, 02ah, 03fh, 02ah, 000h, 015h, 015h, 000h, 015h, 03fh, 000h, 03fh
    58 00012305 151500153F003F      <1>
    59 0001230C 15003F3F2A15152A15- <1>     db  015h, 000h, 03fh, 03fh, 02ah, 015h, 015h, 02ah, 015h, 03fh, 02ah, 03fh, 015h, 02ah, 03fh, 03fh
    59 00012315 3F2A3F152A3F3F      <1>
    60 0001231C 150000150002A152A00- <1>     db  015h, 000h, 000h, 015h, 000h, 02ah, 015h, 02ah, 000h, 015h, 02ah, 02ah, 03fh, 000h, 000h, 03fh
    60 00012325 152A2A3F00003F      <1>
    61 0001232C 002A3F2A003F2A2A15- <1>     db  000h, 02ah, 03fh, 02ah, 000h, 03fh, 02ah, 02ah, 015h, 000h, 015h, 015h, 000h, 03fh, 015h, 02ah
    61 00012335 001515003F152A      <1>
```

```
62 0001233C 15152A3F3F00153F00- <1>     db  015h, 015h, 02ah, 03fh, 03fh, 000h, 015h, 03fh, 000h, 03fh, 03fh, 02ah, 015h, 03fh, 02ah, 03fh
62 00012345 3F3F2A153F2A3F      <1>
63 0001234C 15150015152A153F00- <1>     db  015h, 015h, 000h, 015h, 015h, 02ah, 015h, 03fh, 000h, 015h, 03fh, 02ah, 03fh, 015h, 000h, 03fh
63 00012355 153F2A3F15003F      <1>
64 0001235C 152A3F3F003F3F2A15- <1>     db  015h, 02ah, 03fh, 03fh, 000h, 03fh, 03fh, 02ah, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh
64 00012365 151515153F153F      <1>
65 0001236C 15153F3F3F15153F15- <1>     db  015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh
65 00012375 3F3F3F153F3F3F      <1>

66                              <1> palette3:
67 0001237C 00000000002A002A00- <1>     db  000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah
67 00012385 002A2A2A00002A      <1>
68 0001238C 002A2A15002A2A2A15- <1>     db  000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh
68 00012395 151515153F153F      <1>
69 0001239C 15153F3F3F15153F15- <1>     db  015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh
69 000123A5 3F3F3F153F3F3F      <1>
70 000123AC 000000050505080808- <1>     db  000h, 000h, 000h, 005h, 005h, 005h, 008h, 008h, 008h, 00bh, 00bh, 00bh, 00eh, 00eh, 00eh, 011h
70 000123B5 0B0B0B0E0E0E11      <1>
71 000123BC 111114141418181C- <1>     db  011h, 011h, 014h, 014h, 014h, 018h, 018h, 018h, 01ch, 01ch, 01ch, 020h, 020h, 020h, 024h, 024h
71 000123C5 1C1C2020202424      <1>
72 000123CC 242828282D2D2D3232- <1>     db  024h, 028h, 028h, 028h, 02dh, 02dh, 02dh, 032h, 032h, 032h, 038h, 038h, 038h, 03fh, 03fh, 03fh
72 000123D5 323838383F3F3F      <1>
73 000123DC 00003F10003F1F003F- <1>     db  000h, 000h, 03fh, 010h, 000h, 03fh, 01fh, 000h, 03fh, 02fh, 000h, 03fh, 03fh, 000h, 03fh, 03fh
73 000123E5 2F003F3F003F3F      <1>
74 000123EC 002F3F001F3F00103F- <1>     db  000h, 02fh, 03fh, 000h, 01fh, 03fh, 000h, 010h, 03fh, 000h, 000h, 03fh, 010h, 000h, 03fh, 01fh
74 000123F5 00003F10003F1F      <1>
75 000123FC 003F2F003F3F002F3F- <1>     db  000h, 03fh, 02fh, 000h, 03fh, 03fh, 000h, 02fh, 03fh, 000h, 01fh, 03fh, 000h, 010h, 03fh, 000h
75 00012405 001F3F00103F00      <1>
76 0001240C 003F00003F10003F1F- <1>     db  000h, 03fh, 000h, 000h, 03fh, 010h, 000h, 03fh, 01fh, 000h, 03fh, 02fh, 000h, 03fh, 03fh, 000h
76 00012415 003F2F003F3F00      <1>
77 0001241C 2F3F001F3F00103F1F- <1>     db  02fh, 03fh, 000h, 01fh, 03fh, 000h, 010h, 03fh, 01fh, 01fh, 03fh, 027h, 01fh, 03fh, 02fh, 01fh
77 00012425 1F3F271F3F2F1F      <1>
78 0001242C 3F371F3F3F1F3F3F1F- <1>     db  03fh, 037h, 01fh, 03fh, 03fh, 01fh, 03fh, 03fh, 01fh, 037h, 03fh, 01fh, 02fh, 03fh, 01fh, 027h
78 00012435 373F1F2F3F1F27      <1>
79 0001243C 3F1F1F3F271F3F2F1F- <1>     db  03fh, 01fh, 01fh, 03fh, 027h, 01fh, 03fh, 02fh, 01fh, 03fh, 037h, 01fh, 03fh, 03fh, 01fh, 037h
79 00012445 3F371F3F3F1F37      <1>
80 0001244C 3F1F2F3F1F273F1F1F- <1>     db  03fh, 01fh, 02fh, 03fh, 01fh, 027h, 03fh, 01fh, 01fh, 03fh, 01fh, 01fh, 03fh, 027h, 01fh, 03fh
80 00012455 3F1F1F3F271F3F      <1>
81 0001245C 2F1F3F371F3F3F1F37- <1>     db  02fh, 01fh, 03fh, 037h, 01fh, 03fh, 03fh, 01fh, 037h, 03fh, 01fh, 02fh, 03fh, 01fh, 027h, 03fh
81 00012465 3F1F2F3F1F273F      <1>
82 0001246C 2D2D3F312D3F362D3F- <1>     db  02dh, 02dh, 03fh, 031h, 02dh, 03fh, 036h, 02dh, 03fh, 03ah, 02dh, 03fh, 03fh, 02dh, 03fh, 03fh
82 00012475 3A2D3F3F2D3F3F      <1>
83 0001247C 2D3A3F2D363F2D313F- <1>     db  02dh, 03ah, 03fh, 02dh, 036h, 03fh, 02dh, 031h, 03fh, 02dh, 02dh, 03fh, 031h, 02dh, 03fh, 036h
83 00012485 2D2D3F312D3F36      <1>
84 0001248C 2D3F3A2D3F3F2D3A3F- <1>     db  02dh, 03fh, 03ah, 02dh, 03fh, 03fh, 02dh, 03ah, 03fh, 02dh, 036h, 03fh, 02dh, 031h, 03fh, 02dh
84 00012495 2D363F2D313F2D      <1>
85 0001249C 2D3F2D2D3F312D3F36- <1>     db  02dh, 03fh, 02dh, 02dh, 03fh, 031h, 02dh, 03fh, 036h, 02dh, 03fh, 03ah, 02dh, 03fh, 03fh, 02dh
85 000124A5 2D3F3A2D3F3F2D      <1>
86 000124AC 3A3F2D363F2D313F00- <1>     db  03ah, 03fh, 02dh, 036h, 03fh, 02dh, 031h, 03fh, 000h, 000h, 01ch, 007h, 000h, 01ch, 00eh, 000h
86 000124B5 001C07001C0E00      <1>
87 000124BC 1C15001C1C001C1C00- <1>     db  01ch, 015h, 000h, 01ch, 01ch, 000h, 01ch, 01ch, 000h, 015h, 01ch, 000h, 00eh, 01ch, 000h, 007h
87 000124C5 151C000E1C0007      <1>
88 000124CC 1C00001C07001C0E00- <1>     db  01ch, 000h, 000h, 01ch, 007h, 000h, 01ch, 00eh, 000h, 01ch, 015h, 000h, 01ch, 01ch, 000h, 015h
88 000124D5 1C15001C1C0015      <1>
89 000124DC 1C000E1C00071C0000- <1>     db  01ch, 000h, 00eh, 01ch, 000h, 007h, 01ch, 000h, 000h, 01ch, 000h, 000h, 01ch, 007h, 000h, 01ch
89 000124E5 1C00001C07001C      <1>
90 000124EC 0E001C15001C1C0015- <1>     db  00eh, 000h, 01ch, 015h, 000h, 01ch, 01ch, 000h, 015h, 01ch, 000h, 00eh, 01ch, 000h, 007h, 01ch
90 000124F5 1C000E1C00071C      <1>
91 000124FC 0E0E1C110E1C150E1C- <1>     db  00eh, 00eh, 01ch, 011h, 00eh, 01ch, 015h, 00eh, 01ch, 018h, 00eh, 01ch, 01ch, 00eh, 01ch, 01ch
91 00012505 180E1C1C0E1C1C      <1>
92 0001250C 0E181C0E151C0E111C- <1>     db  00eh, 018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 00eh, 00eh, 01ch, 011h, 00eh, 01ch, 015h
92 00012515 0E0E1C110E1C15      <1>
93 0001251C 0E1C180E1C1C0E181C- <1>     db  00eh, 01ch, 018h, 00eh, 01ch, 01ch, 00eh, 018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 00eh
93 00012525 0E151C0E111C0E      <1>
94 0001252C 0E1C0E0E1C110E1C15- <1>     db  00eh, 01ch, 00eh, 00eh, 01ch, 011h, 00eh, 01ch, 015h, 00eh, 01ch, 018h, 00eh, 01ch, 01ch, 00eh
94 00012535 0E1C180E1C1C0E      <1>
95 0001253C 181C0E151C0E111C14- <1>     db  018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 014h, 014h, 01ch, 016h, 014h, 01ch, 018h, 014h
95 00012545 141C16141C1814      <1>
96 0001254C 1C1A141C1C141C1C14- <1>     db  01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ch, 01ch, 014h, 01ah, 01ch, 014h, 018h, 01ch, 014h, 016h
96 00012555 1A1C14181C1416      <1>
97 0001255C 1C14141C16141C1814- <1>     db  01ch, 014h, 014h, 01ch, 016h, 014h, 01ch, 018h, 014h, 01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ah
97 00012565 1C1A141C1C141A      <1>
98 0001256C 1C14181C14161C1414- <1>     db  01ch, 014h, 018h, 01ch, 014h, 016h, 01ch, 014h, 014h, 01ch, 014h, 014h, 01ch, 016h, 014h, 01ch
98 00012575 1C14141C16141C      <1>
99 0001257C 18141C1A141C1C141A- <1>     db  018h, 014h, 01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ah, 01ch, 014h, 018h, 01ch, 014h, 016h, 01ch
99 00012585 1C14181C14161C      <1>
100 0001258C 000010040010080010- <1>     db  000h, 000h, 010h, 004h, 000h, 010h, 008h, 000h, 010h, 00ch, 000h, 010h, 010h, 000h, 010h, 010h
100 00012595 0C001010001010      <1>
101 0001259C 000C10000810000410- <1>     db  000h, 00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 000h, 000h, 010h, 004h, 000h, 010h, 008h
101 000125A5 00001004001008      <1>
102 000125AC 00100C001010000C10- <1>     db  000h, 010h, 00ch, 000h, 010h, 010h, 000h, 00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 000h
102 000125B5 00081000041000      <1>
103 000125BC 001000001004001008- <1>     db  000h, 010h, 000h, 000h, 010h, 004h, 000h, 010h, 008h, 000h, 010h, 00ch, 000h, 010h, 010h, 000h
103 000125C5 00100C00101000      <1>
104 000125CC 0C1000081000041008- <1>     db  00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 008h, 008h, 010h, 00ah, 008h, 010h, 00ch, 008h
104 000125D5 08100A08100C08      <1>
105 000125DC 100E08101008101008- <1>     db  010h, 00eh, 008h, 010h, 010h, 008h, 010h, 010h, 008h, 00eh, 010h, 008h, 00ch, 010h, 008h, 00ah
105 000125E5 0E10080C10080A      <1>
106 000125EC 100808100A08100C08- <1>     db  010h, 008h, 008h, 010h, 00ah, 008h, 010h, 00ch, 008h, 010h, 00eh, 008h, 010h, 010h, 008h, 00eh
106 000125F5 100E081010080E      <1>
107 000125FC 10080C10080A100808- <1>     db  010h, 008h, 00ch, 010h, 008h, 00ah, 010h, 008h, 008h, 010h, 008h, 008h, 010h, 00ah, 008h, 010h
107 00012605 100808100A0810      <1>
108 0001260C 0C08100E081010080E- <1>     db  00ch, 008h, 010h, 00eh, 008h, 010h, 010h, 008h, 00eh, 010h, 008h, 00ch, 010h, 008h, 00ah, 010h
108 00012615 10080C10080A10      <1>
109 0001261C 0B0B100C0B100D0B10- <1>     db  00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh, 00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh, 010h, 010h
109 00012625 0F0B10100B1010      <1>
110 0001262C 0B0F100B0D100B0C10- <1>     db  00bh, 00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh
110 00012635 0B0B100C0B100D      <1>
111 0001263C 0B100F0B10100B0F10- <1>     db  00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh, 00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 00bh
111 00012645 0B0D100B0C100B      <1>
112 0001264C 0B100B0B100C0B100D- <1>     db  00bh, 010h, 00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh, 00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh
112 00012655 0B100F0B10100B      <1>
113 0001265C 0F100B0D100B0C1000- <1>     db  00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
113 00012665 00000000000000      <1>
114 0001266C 00000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
114 00012675 00000000000000      <1>
115                              <1>
116                              <1>
```

```
                            <1> ; 04/07/2016
                            <1> ; FONT DATA
                            <1>
                            <1> CRT_CHAR_GEN:
                            <1> vgafont8:
0001267C 0000000000000007E- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 081h, 0a5h, 081h, 0bdh, 099h, 081h, 07eh
00012685 81A581BD99817E     <1>
0001268C 7EFFDBFFC3E7FF7E6C- <1>     db  07eh, 0ffh, 0dbh, 0ffh, 0c3h, 0e7h, 0ffh, 07eh, 06ch, 0feh, 0feh, 0feh, 07ch, 038h, 010h, 000h
00012695 FEFEFE7C381000     <1>
0001269C 10387CFE7C38100038- <1>     db  010h, 038h, 07ch, 0feh, 07ch, 038h, 010h, 000h, 038h, 07ch, 038h, 0feh, 0feh, 07ch, 038h, 07ch
000126A5 7C38FEFE7C387C     <1>
000126AC 1010387CFE7C387C00- <1>     db  010h, 010h, 038h, 07ch, 0feh, 07ch, 038h, 07ch, 000h, 000h, 018h, 03ch, 03ch, 018h, 000h, 000h
000126B5 00183C3C180000     <1>
000126BC FFFFE7C3C3E7FFFF00- <1>     db  0ffh, 0ffh, 0e7h, 0c3h, 0c3h, 0e7h, 0ffh, 0ffh, 000h, 03ch, 066h, 042h, 042h, 066h, 03ch, 000h
000126C5 3C664242663C00     <1>
000126CC FFC399BDBD99C3FF0F- <1>     db  0ffh, 0c3h, 099h, 0bdh, 0bdh, 099h, 0c3h, 0ffh, 00fh, 007h, 00fh, 07dh, 0cch, 0cch, 0cch, 078h
000126D5 070F7DCCCCCC78     <1>
000126DC 3C6666663C187E183F- <1>     db  03ch, 066h, 066h, 066h, 03ch, 018h, 07eh, 018h, 03fh, 033h, 03fh, 030h, 030h, 070h, 0f0h, 0e0h
000126E5 333F303070F0E0     <1>
000126EC 7F637F636367E6C099- <1>     db  07fh, 063h, 07fh, 063h, 063h, 067h, 0e6h, 0c0h, 099h, 05ah, 03ch, 0e7h, 0e7h, 03ch, 05ah, 099h
000126F5 5A3CE7E73C5A99     <1>
000126FC 80E0F8FEF8E0800002- <1>     db  080h, 0e0h, 0f8h, 0feh, 0f8h, 0e0h, 080h, 000h, 002h, 00eh, 03eh, 0feh, 03eh, 00eh, 002h, 000h
00012705 0E3EFE3E0E0200     <1>
0001270C 183C7E18187E3C1866- <1>     db  018h, 03ch, 07eh, 018h, 018h, 07eh, 03ch, 018h, 066h, 066h, 066h, 066h, 066h, 000h, 066h, 000h
00012715 66666666006600     <1>
0001271C 7FDBDB7B1B1B1B003E- <1>     db  07fh, 0dbh, 0dbh, 07bh, 01bh, 01bh, 01bh, 000h, 03eh, 063h, 038h, 06ch, 06ch, 038h, 0cch, 078h
00012725 63386C6C38CC78     <1>
0001272C 000000007E7E7E0018- <1>     db  000h, 000h, 000h, 000h, 07eh, 07eh, 07eh, 000h, 018h, 03ch, 07eh, 018h, 07eh, 03ch, 018h, 0ffh
00012735 3C7E187E3C18FF     <1>
0001273C 183C7E181818180018- <1>     db  018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h
00012745 1818187E3C1800     <1>
0001274C 00180CFE0C18000000- <1>     db  000h, 018h, 00ch, 0feh, 00ch, 018h, 000h, 000h, 000h, 030h, 060h, 0feh, 060h, 030h, 000h, 000h
00012755 3060FE60300000     <1>
0001275C 0000C0C0C0FE000000- <1>     db  000h, 000h, 0c0h, 0c0h, 0c0h, 0feh, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h
00012765 2466FF66240000     <1>
0001276C 00183C7EFFFF000000- <1>     db  000h, 018h, 03ch, 07eh, 0ffh, 0ffh, 000h, 000h, 000h, 0ffh, 0ffh, 07eh, 03ch, 018h, 000h, 000h
00012775 FFFF7E3C180000     <1>
0001277C 0000000000000000030- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 030h, 078h, 078h, 030h, 030h, 000h, 030h, 000h
00012785 78783030003000     <1>
0001278C 6C6C6C00000000006C- <1>     db  06ch, 06ch, 06ch, 000h, 000h, 000h, 000h, 000h, 06ch, 06ch, 0feh, 06ch, 0feh, 06ch, 06ch, 000h
00012795 6CFE6CFE6C6C00     <1>
0001279C 307CC0780CF8300000- <1>     db  030h, 07ch, 0c0h, 078h, 00ch, 0f8h, 030h, 000h, 000h, 0c6h, 0cch, 018h, 030h, 066h, 0c6h, 000h
000127A5 C6CC183066C600     <1>
000127AC 386C3876DCCC760060- <1>     db  038h, 06ch, 038h, 076h, 0dch, 0cch, 076h, 000h, 060h, 060h, 0c0h, 000h, 000h, 000h, 000h, 000h
000127B5 60C000000000000    <1>
000127BC 183060606030180060- <1>     db  018h, 030h, 060h, 060h, 060h, 030h, 018h, 000h, 060h, 030h, 018h, 018h, 018h, 030h, 060h, 000h
000127C5 30181818306000     <1>
000127CC 00663CFF3C66000000- <1>     db  000h, 066h, 03ch, 0ffh, 03ch, 066h, 000h, 000h, 000h, 030h, 030h, 0fch, 030h, 030h, 000h, 000h
000127D5 3030FC30300000     <1>
000127DC 0000000000030306000- <1>     db  000h, 000h, 000h, 000h, 000h, 030h, 030h, 060h, 000h, 000h, 000h, 0fch, 000h, 000h, 000h, 000h
000127E5 0000FC00000000     <1>
000127EC 0000000000030300006- <1>     db  000h, 000h, 000h, 000h, 000h, 030h, 030h, 000h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 080h, 000h
000127F5 0C183060C08000     <1>
000127FC 7CC6CEDEF6E67C0030- <1>     db  07ch, 0c6h, 0ceh, 0deh, 0f6h, 0e6h, 07ch, 000h, 030h, 070h, 030h, 030h, 030h, 030h, 0fch, 000h
00012805 7030303030FC00     <1>
0001280C 78CC0C3860CCFC0078- <1>     db  078h, 0cch, 00ch, 038h, 060h, 0cch, 0fch, 000h, 078h, 0cch, 00ch, 038h, 00ch, 0cch, 078h, 000h
00012815 CC0C380CCC7800     <1>
0001281C 1C3C6CCCFE0C1E00FC- <1>     db  01ch, 03ch, 06ch, 0cch, 0feh, 00ch, 01eh, 000h, 0fch, 0c0h, 0f8h, 00ch, 00ch, 0cch, 078h, 000h
00012825 C0F80C0CCC7800     <1>
0001282C 3860C0F8CCCC7800FC- <1>     db  038h, 060h, 0c0h, 0f8h, 0cch, 0cch, 078h, 000h, 0fch, 0cch, 00ch, 018h, 030h, 030h, 030h, 000h
00012835 CC0C1830303000     <1>
0001283C 78CCCC78CCCC780078- <1>     db  078h, 0cch, 0cch, 078h, 0cch, 0cch, 078h, 000h, 078h, 0cch, 0cch, 07ch, 00ch, 018h, 070h, 000h
00012845 CCCC7C0C187000     <1>
0001284C 0030300000030300000- <1>     db  000h, 030h, 030h, 000h, 000h, 030h, 030h, 000h, 000h, 030h, 030h, 000h, 000h, 030h, 030h, 060h
00012855 30300000303060     <1>
0001285C 183060C06030180000- <1>     db  018h, 030h, 060h, 0c0h, 060h, 030h, 018h, 000h, 000h, 000h, 0fch, 000h, 000h, 0fch, 000h, 000h
00012865 00FC0000FC0000     <1>
0001286C 6030180C1830600078- <1>     db  060h, 030h, 018h, 00ch, 018h, 030h, 060h, 000h, 078h, 0cch, 00ch, 018h, 030h, 000h, 030h, 000h
00012875 CC0C1830003000     <1>
0001287C 7CC6DEDEDEC0780030- <1>     db  07ch, 0c6h, 0deh, 0deh, 0deh, 0c0h, 078h, 000h, 030h, 078h, 0cch, 0cch, 0fch, 0cch, 0cch, 000h
00012885 78CCCCFCCCCC00     <1>
0001288C FC66667C6666FC003C- <1>     db  0fch, 066h, 066h, 07ch, 066h, 066h, 0fch, 000h, 03ch, 066h, 0c0h, 0c0h, 0c0h, 066h, 03ch, 000h
00012895 66C0C0C0663C00     <1>
0001289C F86C6666666CF800FE- <1>     db  0f8h, 06ch, 066h, 066h, 066h, 06ch, 0f8h, 000h, 0feh, 062h, 068h, 078h, 068h, 062h, 0feh, 000h
000128A5 6268786862FE00     <1>
000128AC FE6268786860F0003C- <1>     db  0feh, 062h, 068h, 078h, 068h, 060h, 0f0h, 000h, 03ch, 066h, 0c0h, 0c0h, 0ceh, 066h, 03eh, 000h
000128B5 66C0C0CE663E00     <1>
000128BC CCCCCCFCCCCCCC0078- <1>     db  0cch, 0cch, 0cch, 0fch, 0cch, 0cch, 0cch, 000h, 078h, 030h, 030h, 030h, 030h, 030h, 078h, 000h
000128C5 30303030307800     <1>
000128CC 1E0C0C0CCCCC7800E6- <1>     db  01eh, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 000h, 0e6h, 066h, 06ch, 078h, 06ch, 066h, 0e6h, 000h
000128D5 666C786C66E600     <1>
000128DC F060606066266FE00C6- <1>     db  0f0h, 060h, 060h, 060h, 062h, 066h, 0feh, 000h, 0c6h, 0eeh, 0feh, 0feh, 0d6h, 0c6h, 0c6h, 000h
000128E5 EEFEFED6C6C600     <1>
000128EC C6E6F6DECEC6C60038- <1>     db  0c6h, 0e6h, 0f6h, 0deh, 0ceh, 0c6h, 0c6h, 000h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h
000128F5 6CC6C6C66C3800     <1>
000128FC FC66667C6060F00078- <1>     db  0fch, 066h, 066h, 07ch, 060h, 060h, 0f0h, 000h, 078h, 0cch, 0cch, 0cch, 0dch, 078h, 01ch, 000h
00012905 CCCCCCDC781C00     <1>
0001290C FC66667C6C66E60078- <1>     db  0fch, 066h, 066h, 07ch, 06ch, 066h, 0e6h, 000h, 078h, 0cch, 0e0h, 070h, 01ch, 0cch, 078h, 000h
00012915 CCE0701CCC7800     <1>
0001291C FCB4303030307800CC- <1>     db  0fch, 0b4h, 030h, 030h, 030h, 030h, 078h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 0fch, 000h
00012925 CCCCCCCCCCFC00     <1>
0001292C CCCCCCCCCC783000C6- <1>     db  0cch, 0cch, 0cch, 0cch, 0cch, 078h, 030h, 000h, 0c6h, 0c6h, 0c6h, 0d6h, 0feh, 0eeh, 0c6h, 000h
00012935 C6C6D6FEEEC600     <1>
0001293C C6C66C38386CC600CC- <1>     db  0c6h, 0c6h, 06ch, 038h, 038h, 06ch, 0c6h, 000h, 0cch, 0cch, 0cch, 078h, 030h, 030h, 078h, 000h
00012945 CCCC7830307800     <1>
0001294C FEC68C183266FE0078- <1>     db  0feh, 0c6h, 08ch, 018h, 032h, 066h, 0feh, 000h, 078h, 060h, 060h, 060h, 060h, 060h, 078h, 000h
00012955 60606060607800     <1>
0001295C C06030180C06020078- <1>     db  0c0h, 060h, 030h, 018h, 00ch, 006h, 002h, 000h, 078h, 018h, 018h, 018h, 018h, 018h, 078h, 000h
00012965 18181818187800     <1>
```

```
169 0001296C 10386CC600000000000- <1>     db  010h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh
169 00012975 000000000000FF       <1>
170 0001297C 3030180000000000000- <1>     db  030h, 030h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 078h, 00ch, 07ch, 0cch, 076h, 000h
170 00012985 00780C7CCC7600       <1>
171 0001298C E060607C6666DC0000-  <1>     db  0e0h, 060h, 060h, 07ch, 066h, 066h, 0dch, 000h, 000h, 000h, 078h, 0cch, 0c0h, 0cch, 078h, 000h
171 00012995 0078CCC0CC7800       <1>
172 0001299C 1C0C0C7CCCCC760000-  <1>     db  01ch, 00ch, 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h
172 000129A5 0078CCFCC07800       <1>
173 000129AC 386C60F06060F00000-  <1>     db  038h, 06ch, 060h, 0f0h, 060h, 060h, 0f0h, 000h, 000h, 000h, 076h, 0cch, 0cch, 07ch, 00ch, 0f8h
173 000129B5 0076CCCC7C0CF8       <1>
174 000129BC E0606C766666E60030-  <1>     db  0e0h, 060h, 06ch, 076h, 066h, 066h, 0e6h, 000h, 030h, 000h, 070h, 030h, 030h, 030h, 078h, 000h
174 000129C5 00703030307800       <1>
175 000129CC 0C000C0C0CCCCC78E0-  <1>     db  00ch, 000h, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 0e0h, 060h, 066h, 06ch, 078h, 06ch, 0e6h, 000h
175 000129D5 60666C786CE600       <1>
176 000129DC 703030303030780000-  <1>     db  070h, 030h, 030h, 030h, 030h, 030h, 078h, 000h, 000h, 000h, 0cch, 0feh, 0feh, 0d6h, 0c6h, 000h
176 000129E5 00CCFEFED6C600       <1>
177 000129EC 0000F8CCCCCCCC0000-  <1>     db  000h, 000h, 0f8h, 0cch, 0cch, 0cch, 0cch, 000h, 000h, 000h, 078h, 0cch, 0cch, 0cch, 078h, 000h
177 000129F5 0078CCCCCC7800       <1>
178 000129FC 0000DC66667C60F000-  <1>     db  000h, 000h, 0dch, 066h, 066h, 07ch, 060h, 0f0h, 000h, 000h, 076h, 0cch, 0cch, 07ch, 00ch, 01eh
178 00012A05 0076CCCC7C0C1E       <1>
179 00012A0C 0000DC766660F00000-  <1>     db  000h, 000h, 0dch, 076h, 066h, 060h, 0f0h, 000h, 000h, 000h, 07ch, 0c0h, 078h, 00ch, 0f8h, 000h
179 00012A15 007CC0780CF800       <1>
180 00012A1C 10307C303034180000-  <1>     db  010h, 030h, 07ch, 030h, 030h, 034h, 018h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 076h, 000h
180 00012A25 00CCCCCCCC7600       <1>
181 00012A2C 0000CCCCCC78300000-  <1>     db  000h, 000h, 0cch, 0cch, 0cch, 078h, 030h, 000h, 000h, 000h, 0c6h, 0d6h, 0feh, 0feh, 06ch, 000h
181 00012A35 00C6D6FEFE6C00       <1>
182 00012A3C 0000C66C386CC60000-  <1>     db  000h, 000h, 0c6h, 06ch, 038h, 06ch, 0c6h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 07ch, 00ch, 0f8h
182 00012A45 00CCCCCC7C0CF8       <1>
183 00012A4C 0000FC983064FC001C-  <1>     db  000h, 000h, 0fch, 098h, 030h, 064h, 0fch, 000h, 01ch, 030h, 030h, 0e0h, 030h, 030h, 01ch, 000h
183 00012A55 3030E030301C00       <1>
184 00012A5C 1818180018181800E0-  <1>     db  018h, 018h, 018h, 000h, 018h, 018h, 018h, 000h, 0e0h, 030h, 030h, 01ch, 030h, 030h, 0e0h, 000h
184 00012A65 30301C3030E000       <1>
185 00012A6C 76DC000000000000000-  <1>     db  076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 000h
185 00012A75 10386CC6C6FE00       <1>
186 00012A7C 78CCC0CC78180C7800-  <1>     db  078h, 0cch, 0c0h, 0cch, 078h, 018h, 00ch, 078h, 000h, 0cch, 000h, 0cch, 0cch, 0cch, 07eh, 000h
186 00012A85 CC00CCCCCC7E00       <1>
187 00012A8C 1C0078CCFCC078007E-  <1>     db  01ch, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h, 07eh, 0c3h, 03ch, 006h, 03eh, 066h, 03fh, 000h
187 00012A95 C33C063E663F00       <1>
188 00012A9C CC00780C7CCC7E00E0-  <1>     db  0cch, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 0e0h, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h
188 00012AA5 00780C7CCC7E00       <1>
189 00012AAC 3030780C7CCC7E0000-  <1>     db  030h, 030h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 000h, 000h, 078h, 0c0h, 0c0h, 078h, 00ch, 038h
189 00012AB5 0078C0C0780C38       <1>
190 00012ABC 7EC33C667E603C00CC-  <1>     db  07eh, 0c3h, 03ch, 066h, 07eh, 060h, 03ch, 000h, 0cch, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h
190 00012AC5 0078CCFCC07800       <1>
191 00012ACC E00078CCFCC07800CC-  <1>     db  0e0h, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h, 0cch, 000h, 070h, 030h, 030h, 030h, 078h, 000h
191 00012AD5 00703030307800       <1>
192 00012ADC 7CC6381818183C00E0-  <1>     db  07ch, 0c6h, 038h, 018h, 018h, 018h, 03ch, 000h, 0e0h, 000h, 070h, 030h, 030h, 030h, 078h, 000h
192 00012AE5 00703030307800       <1>
193 00012AEC C6386CC6FEC6C60030-  <1>     db  0c6h, 038h, 06ch, 0c6h, 0feh, 0c6h, 0c6h, 000h, 030h, 030h, 000h, 078h, 0cch, 0fch, 0cch, 000h
193 00012AF5 300078CCFCCC00       <1>
194 00012AFC 1C00FC607860FC0000-  <1>     db  01ch, 000h, 0fch, 060h, 078h, 060h, 0fch, 000h, 000h, 000h, 07fh, 00ch, 07fh, 0cch, 07fh, 000h
194 00012B05 007F0C7FCC7F00       <1>
195 00012B0C 3E6CCCFECCCCCE0078-  <1>     db  03eh, 06ch, 0cch, 0feh, 0cch, 0cch, 0ceh, 000h, 078h, 0cch, 000h, 078h, 0cch, 0cch, 078h, 000h
195 00012B15 CC0078CCCC7800       <1>
196 00012B1C 00CC0078CCCC780000-  <1>     db  000h, 0cch, 000h, 078h, 0cch, 0cch, 078h, 000h, 000h, 0e0h, 000h, 078h, 0cch, 0cch, 078h, 000h
196 00012B25 E00078CCCC7800       <1>
197 00012B2C 78CC00CCCCCC7E0000-  <1>     db  078h, 0cch, 000h, 0cch, 0cch, 0cch, 07eh, 000h, 000h, 0e0h, 000h, 0cch, 0cch, 0cch, 07eh, 000h
197 00012B35 E000CCCCCC7E00       <1>
198 00012B3C 00CC00CCCC7C0CF8C3-  <1>     db  000h, 0cch, 000h, 0cch, 0cch, 07ch, 00ch, 0f8h, 0c3h, 018h, 03ch, 066h, 066h, 03ch, 018h, 000h
198 00012B45 183C66663C1800       <1>
199 00012B4C CC00CCCCCCCC780018-  <1>     db  0cch, 000h, 0cch, 0cch, 0cch, 0cch, 078h, 000h, 018h, 018h, 07eh, 0c0h, 0c0h, 07eh, 018h, 018h
199 00012B55 187EC0C07E1818       <1>
200 00012B5C 386C64F060E6FC00CC-  <1>     db  038h, 06ch, 064h, 0f0h, 060h, 0e6h, 0fch, 000h, 0cch, 0cch, 078h, 0fch, 030h, 0fch, 030h, 030h
200 00012B65 CC78FC30FC3030       <1>
201 00012B6C F8CCCCFAC6CFC6C70E-  <1>     db  0f8h, 0cch, 0cch, 0fah, 0c6h, 0cfh, 0c6h, 0c7h, 00eh, 01bh, 018h, 03ch, 018h, 018h, 0d8h, 070h
201 00012B75 1B183C1818D870       <1>
202 00012B7C 1C00780C7CCC7E0038-  <1>     db  01ch, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 038h, 000h, 070h, 030h, 030h, 030h, 078h, 000h
202 00012B85 00703030307800       <1>
203 00012B8C 001C0078CCCC780000-  <1>     db  000h, 01ch, 000h, 078h, 0cch, 0cch, 078h, 000h, 000h, 01ch, 000h, 0cch, 0cch, 0cch, 07eh, 000h
203 00012B95 1C00CCCCCC7E00       <1>
204 00012B9C 00F800F8CCCCCC00FC-  <1>     db  000h, 0f8h, 000h, 0f8h, 0cch, 0cch, 0cch, 000h, 0fch, 000h, 0cch, 0ech, 0fch, 0dch, 0cch, 000h
204 00012BA5 00CCECFCDCCC00       <1>
205 00012BAC 3C6C6C3E007E000038-  <1>     db  03ch, 06ch, 06ch, 03eh, 000h, 07eh, 000h, 000h, 038h, 06ch, 06ch, 038h, 000h, 07ch, 000h, 000h
205 00012BB5 6C6C38007C0000       <1>
206 00012BBC 30003060C0CC780000-  <1>     db  030h, 000h, 030h, 060h, 0c0h, 0cch, 078h, 000h, 000h, 000h, 000h, 0fch, 0c0h, 0c0h, 000h, 000h
206 00012BC5 0000FCC0C00000       <1>
207 00012BCC 000000FC0C0C0000C3-  <1>     db  000h, 000h, 000h, 0fch, 00ch, 00ch, 000h, 000h, 0c3h, 0c6h, 0cch, 0deh, 033h, 066h, 0cch, 00fh
207 00012BD5 C6CCDE3366CC0F       <1>
208 00012BDC C3C6CCDB376FCF0318-  <1>     db  0c3h, 0c6h, 0cch, 0dbh, 037h, 06fh, 0cfh, 003h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 000h
208 00012BE5 18001818181800       <1>
209 00012BEC 003366CC6633000000-  <1>     db  000h, 033h, 066h, 0cch, 066h, 033h, 000h, 000h, 000h, 0cch, 066h, 033h, 066h, 0cch, 000h, 000h
209 00012BF5 CC663366CC0000       <1>
210 00012BFC 228822882288228855-  <1>     db  022h, 088h, 022h, 088h, 022h, 088h, 022h, 088h, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah
210 00012C05 AA55AA55AA55AA       <1>
211 00012C0C DB77DBEEDB77DBEE18-  <1>     db  0dbh, 077h, 0dbh, 0eeh, 0dbh, 077h, 0dbh, 0eeh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
211 00012C15 18181818181818       <1>
212 00012C1C 18181818F818181818-  <1>     db  018h, 018h, 018h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h, 018h
212 00012C25 18F818F8181818       <1>
213 00012C2C 36363636F636363600-  <1>     db  036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h
213 00012C35 000000FE363636       <1>
214 00012C3C 0000F818F818181836-  <1>     db  000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h, 036h
214 00012C45 36F606F6363636       <1>
215 00012C4C 363636363636363600-  <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 000h, 000h, 0feh, 006h, 0f6h, 036h, 036h, 036h
215 00012C55 00FE06F6363636       <1>
216 00012C5C 3636F606FE00000036-  <1>     db  036h, 036h, 0f6h, 006h, 0feh, 000h, 000h, 000h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h, 000h
216 00012C65 363636FE000000       <1>
217 00012C6C 1818F818F800000000-  <1>     db  018h, 018h, 0f8h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 018h, 018h
217 00012C75 000000F8181818       <1>
218 00012C7C 181818181F00000018-  <1>     db  018h, 018h, 018h, 018h, 01fh, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 0ffh, 000h, 000h, 000h
218 00012C85 181818FF000000       <1>
219 00012C8C 00000000FF18181818-  <1>     db  000h, 000h, 000h, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 018h, 018h
219 00012C95 1818181F181818       <1>
220 00012C9C 00000000FF00000018-  <1>     db  000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h
220 00012CA5 181818FF181818       <1>
221 00012CAC 18181F181F18181836-  <1>     db  018h, 018h, 01fh, 018h, 01fh, 018h, 018h, 018h, 036h, 036h, 036h, 036h, 037h, 036h, 036h, 036h
221 00012CB5 36363637363636       <1>
222 00012CBC 363637303F00000000-  <1>     db  036h, 036h, 037h, 030h, 03fh, 000h, 000h, 000h, 000h, 000h, 03fh, 030h, 037h, 036h, 036h, 036h
222 00012CC5 003F3037363636       <1>
```

```
223 00012CCC 3636F700FF00000000- <1>     db  036h, 036h, 0f7h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0f7h, 036h, 036h, 036h
223 00012CD5 00FF00F7363636      <1>
224 00012CDC 363637303736363600- <1>     db  036h, 036h, 037h, 030h, 037h, 036h, 036h, 036h, 000h, 000h, 0ffh, 000h, 0ffh, 000h, 000h, 000h
224 00012CE5 00FF00FF000000      <1>
225 00012CEC 3636F700F736363618- <1>     db  036h, 036h, 0f7h, 000h, 0f7h, 036h, 036h, 036h, 018h, 018h, 0ffh, 000h, 0ffh, 000h, 000h, 000h
225 00012CF5 18FF00FF000000      <1>
226 00012CFC 36363636FF00000000- <1>     db  036h, 036h, 036h, 036h, 0ffh, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 018h, 018h, 018h
226 00012D05 00FF00FF181818      <1>
227 00012D0C 00000000FF36363636- <1>     db  000h, 000h, 000h, 000h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 03fh, 000h, 000h, 000h
227 00012D15 3636363F000000      <1>
228 00012D1C 18181F181F00000000- <1>     db  018h, 018h, 01fh, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h, 018h
228 00012D25 001F181F181818      <1>
229 00012D2C 000000003F36363636- <1>     db  000h, 000h, 000h, 000h, 03fh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h
229 00012D35 363636FF363636      <1>
230 00012D3C 1818FF18FF18181818- <1>     db  018h, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 000h, 000h, 000h
230 00012D45 181818F8000000      <1>
231 00012D4C 000000001F181818FF- <1>     db  000h, 000h, 000h, 000h, 01fh, 018h, 018h, 018h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
231 00012D55 FFFFFFFFFFFFFF      <1>
232 00012D5C 00000000FFFFFFFFF0- <1>     db  000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h
232 00012D65 F0F0F0F0F0F0F0      <1>
233 00012D6C 0F0F0F0F0F0F0F0FFF- <1>     db  00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h
233 00012D75 FFFFFF00000000      <1>
234 00012D7C 000076DCC8DC760000- <1>     db  000h, 000h, 076h, 0dch, 0c8h, 0dch, 076h, 000h, 000h, 078h, 0cch, 0f8h, 0cch, 0f8h, 0c0h, 0c0h
234 00012D85 78CCF8CCF8C0C0      <1>
235 00012D8C 00FCCCC0C0C0C00000- <1>     db  000h, 0fch, 0cch, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 0feh, 06ch, 06ch, 06ch, 06ch, 06ch, 000h
235 00012D95 FE6C6C6C6C6C00      <1>
236 00012D9C FCCC603060CCFC0000- <1>     db  0fch, 0cch, 060h, 030h, 060h, 0cch, 0fch, 000h, 000h, 000h, 07eh, 0d8h, 0d8h, 0d8h, 070h, 000h
236 00012DA5 007ED8D8D87000      <1>
237 00012DAC 00666666667C60C000- <1>     db  000h, 066h, 066h, 066h, 066h, 07ch, 060h, 0c0h, 000h, 076h, 0dch, 018h, 018h, 018h, 018h, 000h
237 00012DB5 76DC1818181800      <1>
238 00012DBC FC3078CCCC7830FC38- <1>     db  0fch, 030h, 078h, 0cch, 0cch, 078h, 030h, 0fch, 038h, 06ch, 0c6h, 0feh, 0c6h, 06ch, 038h, 000h
238 00012DC5 6CC6FEC66C3800      <1>
239 00012DCC 386CC6C66C6CEE001C- <1>     db  038h, 06ch, 0c6h, 0c6h, 06ch, 06ch, 0eeh, 000h, 01ch, 030h, 018h, 07ch, 0cch, 0cch, 078h, 000h
239 00012DD5 30187CCCCC7800      <1>
240 00012DDC 00007EDBDB7E000006- <1>     db  000h, 000h, 07eh, 0dbh, 0dbh, 07eh, 000h, 000h, 006h, 00ch, 07eh, 0dbh, 0dbh, 07eh, 060h, 0c0h
240 00012DE5 0C7EDBDB7E60C0      <1>
241 00012DEC 3860C0F8C060380078- <1>     db  038h, 060h, 0c0h, 0f8h, 0c0h, 060h, 038h, 000h, 078h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 000h
241 00012DF5 CCCCCCCCCCCC00      <1>
242 00012DFC 00FC00FC00FC000030- <1>     db  000h, 0fch, 000h, 0fch, 000h, 0fch, 000h, 000h, 030h, 030h, 0fch, 030h, 030h, 000h, 0fch, 000h
242 00012E05 30FC303000FC00      <1>
243 00012E0C 603018306000FC0018- <1>     db  060h, 030h, 018h, 030h, 060h, 000h, 0fch, 000h, 018h, 030h, 060h, 030h, 018h, 000h, 0fch, 000h
243 00012E15 3060301800FC00      <1>
244 00012E1C 0E1B1B181818181818- <1>     db  00eh, 01bh, 01bh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0d8h, 0d8h, 070h
244 00012E25 18181818D8D870      <1>
245 00012E2C 303000FC0030300000- <1>     db  030h, 030h, 000h, 0fch, 000h, 030h, 030h, 000h, 000h, 076h, 0dch, 000h, 076h, 0dch, 000h, 000h
245 00012E35 76DC0076DC0000      <1>
246 00012E3C 386C6C380000000000- <1>     db  038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h
246 00012E45 00001818000000      <1>
247 00012E4C 0000000180000000F- <1>     db  000h, 000h, 000h, 000h, 018h, 000h, 000h, 000h, 00fh, 00ch, 00ch, 00ch, 0ech, 06ch, 03ch, 01ch
247 00012E55 0C0C0CEC6C3C1C      <1>
248 00012E5C 786C6C6C6C00000070- <1>     db  078h, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 070h, 018h, 030h, 060h, 078h, 000h, 000h, 000h
248 00012E65 18306078000000      <1>
249 00012E6C 00003C3C3C3C000000- <1>     db  000h, 000h, 03ch, 03ch, 03ch, 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
249 00012E75 00000000000000      <1>
250                              <1> vgafont14:
251 00012E7C 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
251 00012E85 00000000000000      <1>
252 00012E8C 7E81A58181BD99817E- <1>     db  07eh, 081h, 0a5h, 081h, 081h, 0bdh, 099h, 081h, 07eh, 000h, 000h, 000h, 000h, 000h, 07eh, 0ffh
252 00012E95 00000000007EFF      <1>
253 00012E9C DBFFFFC3E7FF7E0000- <1>     db  0dbh, 0ffh, 0ffh, 0c3h, 0e7h, 0ffh, 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 06ch, 0feh, 0feh
253 00012EA5 000000006CFEFE      <1>
254 00012EAC FEFE7C381000000000- <1>     db  0feh, 0feh, 07ch, 038h, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 07ch, 0feh, 07ch
254 00012EB5 000010387CFE7C      <1>
255 00012EBC 3810000000000018- <1>     db  038h, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 03ch, 0e7h, 0e7h, 0e7h, 018h, 018h
255 00012EC5 3C3CE7E7E71818      <1>
256 00012ECC 3C0000000000183C7E- <1>     db  03ch, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 07eh, 0ffh, 0ffh, 07eh, 018h, 018h, 03ch, 000h
256 00012ED5 FFFF7E18183C00      <1>
257 00012EDC 000000000000183C- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 03ch, 018h, 000h, 000h, 000h, 000h, 000h
257 00012EE5 3C180000000000      <1>
258 00012EEC FFFFFFFFFFE7C3C3E7- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0e7h, 0c3h, 0c3h, 0e7h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h
258 00012EF5 FFFFFFFFFF0000      <1>
259 00012EFC 00003C664242663C00- <1>     db  000h, 000h, 03ch, 066h, 042h, 042h, 066h, 03ch, 000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh
259 00012F05 000000FFFFFFFF      <1>
260 00012F0C C399BDBD99C3FFFFFF- <1>     db  0c3h, 099h, 0bdh, 0bdh, 099h, 0c3h, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 01eh, 00eh, 01ah, 032h
260 00012F15 FF00001E0E1A32      <1>
261 00012F1C 78CCCCCC7800000000- <1>     db  078h, 0cch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 066h, 066h, 03ch, 018h
261 00012F25 003C6666663C18      <1>
262 00012F2C 7E1818000000000 3F- <1>     db  07eh, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 03fh, 033h, 03fh, 030h, 030h, 030h, 070h, 0f0h
262 00012F35 333F30303070F0      <1>
263 00012F3C E0000000000007F637F- <1>     db  0e0h, 000h, 000h, 000h, 000h, 000h, 07fh, 063h, 07fh, 063h, 063h, 063h, 067h, 0e7h, 0e6h, 0c0h
263 00012F45 63636367E7E6C0      <1>
264 00012F4C 000000001818DB3CE7- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 0dbh, 03ch, 0e7h, 03ch, 0dbh, 018h, 018h, 000h, 000h, 000h
264 00012F55 3CDB1818000000      <1>
265 00012F5C 000080C0E0F8FEF8E0- <1>     db  000h, 000h, 080h, 0c0h, 0e0h, 0f8h, 0feh, 0f8h, 0e0h, 0c0h, 080h, 000h, 000h, 000h, 000h, 000h
265 00012F65 C0800000000000      <1>
266 00012F6C 02060E3EFE3E0E0602- <1>     db  002h, 006h, 00eh, 03eh, 0feh, 03eh, 00eh, 006h, 002h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch
266 00012F75 0000000000183C      <1>
267 00012F7C 7E1818187E3C180000- <1>     db  07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h
267 00012F85 00000066666666      <1>
268 00012F8C 666600666600000000- <1>     db  066h, 066h, 000h, 066h, 066h, 000h, 000h, 000h, 000h, 000h, 07fh, 0dbh, 0dbh, 0dbh, 07bh, 01bh
268 00012F95 007FDBDBDB7B1B      <1>
269 00012F9C 1B1B1B000000007CC6- <1>     db  01bh, 01bh, 01bh, 000h, 000h, 000h, 000h, 07ch, 0c6h, 060h, 038h, 06ch, 0c6h, 0c6h, 06ch, 038h
269 00012FA5 60386CC6C66C38      <1>
270 00012FAC 0CC67C000000000000- <1>     db  00ch, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 0feh, 0feh, 000h
270 00012FB5 000000FEFEFE00      <1>
271 00012FBC 00000000183C7E1818- <1>     db  000h, 000h, 000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 07eh, 000h, 000h
271 00012FC5 187E3C187E0000      <1>
272 00012FCC 0000183C7E18181818- <1>     db  000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h
272 00012FD5 18180000000000      <1>
273 00012FDC 1818181818187E3C18- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
273 00012FE5 00000000000000      <1>
274 00012FEC 180CFE0C1800000000- <1>     db  018h, 00ch, 0feh, 00ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 030h, 060h
274 00012FF5 00000000003060      <1>
275 00012FFC FE60300000000000000- <1>     db  0feh, 060h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0c0h, 0c0h, 0c0h
275 00013005 000000000C0C0C0     <1>
276 0001300C FE00000000000000000- <1>     db  0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 028h, 06ch, 0feh, 06ch, 028h, 000h
276 00013015 00286CFE6C2800      <1>
277 0001301C 000000000000001038- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 038h, 07ch, 07ch, 0feh, 0feh, 000h, 000h
```

```
277 00013025 387C7CFEFE0000      <1>
278 0001302C 0000000000FEFE7C7C- <1>     db  000h, 000h, 000h, 000h, 000h, 0feh, 0feh, 07ch, 07ch, 038h, 038h, 010h, 000h, 000h, 000h, 000h
278 00013035 383810000000000     <1>
279 0001303C 0000000000000000000- <1>    db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
279 00013045 00000000000000      <1>
280 0001304C 183C3C3C1818001818- <1>     db  018h, 03ch, 03ch, 03ch, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 066h, 066h, 066h
280 00013055 00000000666666      <1>
281 0001305C 240000000000000000- <1>     db  024h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 06ch, 06ch, 0feh, 06ch
281 00013065 0000006C6CFE6C      <1>
282 0001306C 6C6CFE6C6C00000018- <1>     db  06ch, 06ch, 0feh, 06ch, 06ch, 000h, 000h, 000h, 018h, 018h, 07ch, 0c6h, 0c2h, 0c0h, 07ch, 006h
282 00013075 187CC6C2C07C06      <1>
283 0001307C 86C67C181800000000- <1>     db  086h, 0c6h, 07ch, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 0c2h, 0c6h, 00ch, 018h, 030h, 066h
283 00013085 00C2C60C183066      <1>
284 0001308C C60000000000386C6C- <1>     db  0c6h, 000h, 000h, 000h, 000h, 000h, 038h, 06ch, 06ch, 038h, 076h, 0dch, 0cch, 0cch, 076h, 000h
284 00013095 3876DCCCCC7600      <1>
285 0001309C 0000003030306000000- <1>    db  000h, 000h, 000h, 030h, 030h, 030h, 060h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
285 000130A5 00000000000000      <1>
286 000130AC 00000C183030303030- <1>     db  000h, 000h, 00ch, 018h, 030h, 030h, 030h, 030h, 030h, 018h, 00ch, 000h, 000h, 000h, 000h, 000h
286 000130B5 180C0000000000      <1>
287 000130BC 30180C0C0C0C0C1830- <1>     db  030h, 018h, 00ch, 00ch, 00ch, 00ch, 00ch, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
287 000130C5 00000000000000      <1>
288 000130CC 663CFF3C66000000000- <1>    db  066h, 03ch, 0ffh, 03ch, 066h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h
288 000130D5 00000000001818      <1>
289 000130DC 7E18180000000000000- <1>    db  07eh, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
289 000130E5 00000000000000      <1>
290 000130EC 1818183000000000000- <1>    db  018h, 018h, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 000h, 000h, 000h
290 000130F5 000000FE000000      <1>
291 000130FC 0000000000000000000- <1>    db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h
291 00013105 00000000181800      <1>
292 0001310C 0000000002060C1830- <1>     db  000h, 000h, 000h, 000h, 002h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 080h, 000h, 000h, 000h, 000h
292 00013115 60C08000000000      <1>
293 0001311C 00007CC6CEDEF6E6C6- <1>     db  000h, 000h, 07ch, 0c6h, 0ceh, 0deh, 0f6h, 0e6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
293 00013125 C67C0000000000      <1>
294 0001312C 183878181818187E- <1>       db  018h, 038h, 078h, 018h, 018h, 018h, 018h, 018h, 07eh, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h
294 00013135 000000000007CC6      <1>
295 0001313C 060C183060C6FE0000- <1>     db  006h, 00ch, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 006h, 006h
295 00013145 0000007CC60606      <1>
296 0001314C 3C0606C67C00000000- <1>     db  03ch, 006h, 006h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 00ch, 01ch, 03ch, 06ch, 0cch, 0feh
296 00013155 000C1C3C6CCCFE      <1>
297 0001315C 0C0C1E0000000000FE- <1>     db  00ch, 00ch, 01eh, 000h, 000h, 000h, 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 0fch, 006h, 006h, 0c6h
297 00013165 C0C0C0FC0606C6      <1>
298 0001316C 7C00000000003860C0- <1>     db  07ch, 000h, 000h, 000h, 000h, 000h, 038h, 060h, 0c0h, 0c0h, 0fch, 0c6h, 0c6h, 0c6h, 07ch, 000h
298 00013175 C0FCC6C6C67C00      <1>
299 0001317C 00000000FEC6060C18- <1>     db  000h, 000h, 000h, 000h, 0feh, 0c6h, 006h, 00ch, 018h, 030h, 030h, 030h, 030h, 000h, 000h, 000h
299 00013185 30303030000000      <1>
300 0001318C 00007CC6C6C67CC6C6- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 07ch, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
300 00013195 C67C0000000000      <1>
301 0001319C 7CC6C6C67E06060C78- <1>     db  07ch, 0c6h, 0c6h, 0c6h, 07eh, 006h, 006h, 00ch, 078h, 000h, 000h, 000h, 000h, 000h, 000h, 018h
301 000131A5 00000000000018      <1>
302 000131AC 1800000018180000000- <1>    db  018h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h
302 000131B5 00000000181800      <1>
303 000131BC 0000181830000000000- <1>    db  000h, 000h, 018h, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 006h, 00ch, 018h, 030h, 060h, 030h
303 000131C5 00060C18306030      <1>
304 000131CC 180C060000000000000- <1>    db  018h, 00ch, 006h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 000h, 000h, 07eh, 000h
304 000131D5 00007E00007E00      <1>
305 000131DC 0000000000000603018- <1>    db  000h, 000h, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 00ch, 006h, 00ch, 018h, 030h, 060h, 000h
305 000131E5 0C060C18306000      <1>
306 000131EC 000000007CC6C60C18- <1>     db  000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 00ch, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h
306 000131F5 18001818000000      <1>
307 000131FC 00007CC6C6DEDEDEDC- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 0deh, 0deh, 0deh, 0dch, 0c0h, 07ch, 000h, 000h, 000h, 000h, 000h
307 00013205 C07C0000000000      <1>
308 0001320C 10386CC6C6FEC6C6C6- <1>     db  010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h, 0fch, 066h
308 00013215 0000000000FC66      <1>
309 0001321C 66667C666666FC0000- <1>     db  066h, 066h, 07ch, 066h, 066h, 066h, 0fch, 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 0c2h, 0c0h
309 00013225 0000003C66C2C0      <1>
310 0001322C C0C0C2663C00000000- <1>     db  0c0h, 0c0h, 0c2h, 066h, 03ch, 000h, 000h, 000h, 000h, 000h, 0f8h, 06ch, 066h, 066h, 066h, 066h
310 00013235 00F86C66666666      <1>
311 0001323C 666CF800000000000FE- <1>    db  066h, 06ch, 0f8h, 000h, 000h, 000h, 000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 062h, 066h
311 00013245 66626878686266      <1>
312 0001324C FE0000000000FE6662- <1>     db  0feh, 000h, 000h, 000h, 000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 060h, 060h, 0f0h, 000h
312 00013255 6878686060F000      <1>
313 0001325C 000000003C66C2C0C0- <1>     db  000h, 000h, 000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0deh, 0c6h, 066h, 03ah, 000h, 000h, 000h
313 00013265 DEC6663A000000      <1>
314 0001326C 0000C6C6C6C6FEC6C6- <1>     db  000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h
314 00013275 C6C600000000000      <1>
315 0001327C 3C181818181818183C- <1>     db  03ch, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 01eh, 00ch
315 00013285 00000000001E0C      <1>
316 0001328C 0C0C0C0CCCCC780000- <1>     db  00ch, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h, 000h, 0e6h, 066h, 06ch, 06ch
316 00013295 000000E6666C6C      <1>
317 0001329C 786C6C66E600000000- <1>     db  078h, 06ch, 06ch, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 0f0h, 060h, 060h, 060h, 060h, 060h
317 000132A5 00F06060606060      <1>
318 000132AC 6266FE0000000000C6- <1>     db  062h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h, 0c6h, 0eeh, 0feh, 0feh, 0d6h, 0c6h, 0c6h, 0c6h
318 000132B5 EEFEFED6C6C6C6      <1>
319 000132BC C600000000000C6E6F6- <1>    db  0c6h, 000h, 000h, 000h, 000h, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h, 0c6h, 0c6h, 000h
319 000132C5 FEDECEC6C6C600      <1>
320 000132CC 00000000386CC6C6C6- <1>     db  000h, 000h, 000h, 000h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h
320 000132D5 C6C66C38000000      <1>
321 000132DC 0000FC6666667C6060- <1>     db  000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h, 000h
321 000132E5 60F00000000000      <1>
322 000132EC 7CC6C6C6C6D6DE7C0C- <1>     db  07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0d6h, 0deh, 07ch, 00ch, 00eh, 000h, 000h, 000h, 000h, 0fch, 066h
322 000132F5 0E00000000FC66      <1>
323 000132FC 66667C6C6666E60000- <1>     db  066h, 066h, 07ch, 06ch, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 060h
323 00013305 0000007CC6C660      <1>
324 0001330C 380CC6C67C00000000- <1>     db  038h, 00ch, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 07eh, 07eh, 05ah, 018h, 018h, 018h
324 00013315 007E7E5A181818      <1>
325 0001331C 18183C0000000000C6- <1>     db  018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h
325 00013325 C6C6C6C6C6C6C6      <1>
326 0001332C 7C0000000000C6C6C6- <1>     db  07ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 010h, 000h
326 00013335 C6C6C66C381000      <1>
327 0001333C 00000000C6C6C6C6D6- <1>     db  000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0d6h, 0d6h, 0feh, 07ch, 06ch, 000h, 000h, 000h
327 00013345 D6FE7C6C000000      <1>
328 0001334C 0000C6C66C3838386C- <1>     db  000h, 000h, 0c6h, 0c6h, 06ch, 038h, 038h, 038h, 06ch, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h
328 00013355 C6C600000000000      <1>
329 0001335C 666666663C1818183C- <1>     db  066h, 066h, 066h, 066h, 03ch, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 0feh, 0c6h
329 00013365 0000000000FEC6      <1>
330 0001336C 8C183060C2C6FE0000- <1>     db  08ch, 018h, 030h, 060h, 0c2h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h, 03ch, 030h, 030h, 030h
330 00013375 0000003C303030      <1>
331 0001337C 303030303C00000000- <1>     db  030h, 030h, 030h, 030h, 03ch, 000h, 000h, 000h, 000h, 000h, 080h, 0c0h, 0e0h, 070h, 038h, 01ch
```

```
331 00013385 0080C0E070381C      <1>
332 0001338C 0E0602000000000 3C- <1>     db  00eh, 006h, 002h, 000h, 000h, 000h, 000h, 000h, 03ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch
332 00013395 0C0C0C0C0C0C0C      <1>
333 0001339C 3C000000010386CC600- <1>     db  03ch, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
333 000133A5 00000000000000      <1>
334 000133AC 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h
334 000133B5 0000000000FF00      <1>
335 000133BC 3030180000000000000- <1>     db  030h, 030h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
335 000133C5 00000000000000      <1>
336 000133CC 000000780C7CCCCC76- <1>     db  000h, 000h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 0e0h, 060h
336 000133D5 0000000000E060      <1>
337 000133DC 60786C6666667C0000- <1>     db  060h, 078h, 06ch, 066h, 066h, 066h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch
337 000133E5 0000000000007C      <1>
338 000133EC C6C0C0C67C00000000- <1>     db  0c6h, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 01ch, 00ch, 00ch, 03ch, 06ch, 0cch
338 000133F5 001C0C0C3C6CCC      <1>
339 000133FC CCCC760000000000000- <1>     db  0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h
339 00013405 00007CC6FEC0C6      <1>
340 0001340C 7C0000000000386C64- <1>     db  07ch, 000h, 000h, 000h, 000h, 000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h, 060h, 0f0h, 000h
340 00013415 60F0606060F000      <1>
341 0001341C 0000000000000076CC- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 07ch, 00ch, 0cch, 078h, 000h
341 00013425 CCCC7C0CCC7800      <1>
342 0001342C 0000E060606C766666- <1>     db  000h, 000h, 0e0h, 060h, 060h, 06ch, 076h, 066h, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h
342 00013435 66E60000000000      <1>
343 0001343C 18180038181818183C- <1>     db  018h, 018h, 000h, 038h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 006h, 006h
343 00013445 00000000000606      <1>
344 0001344C 000E0606060666663C- <1>     db  000h, 00eh, 006h, 006h, 006h, 006h, 066h, 066h, 03ch, 000h, 000h, 000h, 0e0h, 060h, 060h, 066h
344 00013455 000000E0606066      <1>
345 0001345C 6C786C66E600000000- <1>     db  06ch, 078h, 06ch, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 038h, 018h, 018h, 018h, 018h, 018h
345 00013465 00381818181818      <1>
346 0001346C 18183C000000000000- <1>     db  018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ech, 0feh, 0d6h, 0d6h, 0d6h
346 00013475 0000ECFED6D6D6      <1>
347 0001347C C600000000000000000- <1>     db  0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 000h
347 00013485 DC666666666600      <1>
348 0001348C 000000000000007CC6- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h
348 00013495 C6C6C67C000000      <1>
349 0001349C 0000000000DC666666- <1>     db  000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 07ch, 060h, 060h, 0f0h, 000h, 000h, 000h
349 000134A5 7C6060F0000000      <1>
350 000134AC 00000076CCCCCC7C0C- <1>     db  000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 07ch, 00ch, 00ch, 01eh, 000h, 000h, 000h, 000h, 000h
350 000134B5 0C1E0000000000      <1>
351 000134BC 00DC76666060F00000- <1>     db  000h, 0dch, 076h, 066h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch
351 000134C5 0000000000007C      <1>
352 000134CC C6701CC67C00000000- <1>     db  0c6h, 070h, 01ch, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 010h, 030h, 030h, 0fch, 030h, 030h
352 000134D5 00103030FC3030      <1>
353 000134DC 30361C000000000000- <1>     db  030h, 036h, 01ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch
353 000134E5 0000CCCCCCCCCC      <1>
354 000134EC 760000000000000000- <1>     db  076h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h, 03ch, 018h, 000h
354 000134F5 666666663C1800      <1>
355 000134FC 00000000000000C6C6- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0d6h, 0d6h, 0feh, 06ch, 000h, 000h, 000h
355 00013505 D6D6FE6C000000      <1>
356 0001350C 00000000000C66C3838- <1>     db  000h, 000h, 000h, 000h, 000h, 0c6h, 06ch, 038h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h
356 00013515 6CC600000000000      <1>
357 0001351C 000000C6C6C6C67E06- <1>     db  000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 0f8h, 000h, 000h, 000h, 000h, 000h
357 00013525 0CF800000000000      <1>
358 0001352C 00FECC183066FE0000- <1>     db  000h, 0feh, 0cch, 018h, 030h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h, 00eh, 018h, 018h, 018h
358 00013535 0000000E181818      <1>
359 0001353C 701818180E00000000- <1>     db  070h, 018h, 018h, 018h, 00eh, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 000h, 018h
359 00013545 00181818180018      <1>
360 0001354C 1818180000000000070- <1>     db  018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 070h, 018h, 018h, 018h, 00eh, 018h, 018h, 018h
360 00013555 1818180E181818      <1>
361 0001355C 70000000000076DC00- <1>     db  070h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
361 00013565 00000000000000      <1>
362 0001356C 000000000000010386C- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 000h, 000h, 000h, 000h
362 00013575 C6C6FE00000000      <1>
363 0001357C 00003C66C2C0C0C266- <1>     db  000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0c2h, 066h, 03ch, 00ch, 006h, 07ch, 000h, 000h, 000h
363 00013585 3C0C067C000000      <1>
364 0001358C CCCC00CCCCCCCCCC76- <1>     db  0cch, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 00ch, 018h, 030h
364 00013595 000000000C1830      <1>
365 0001359C 007CC6FEC0C67C0000- <1>     db  000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 000h, 078h
365 000135A5 000010386C0078      <1>
366 000135AC 0C7CCCCC7600000000- <1>     db  00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 0cch, 0cch, 000h, 078h, 00ch, 07ch
366 000135B5 00CCCC00780C7C      <1>
367 000135BC CCCC760000000006030- <1>     db  0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 078h, 00ch, 07ch, 0cch, 0cch
367 000135C5 1800780C7CCCCC      <1>
368 000135CC 76000000000386C3800- <1>     db  076h, 000h, 000h, 000h, 000h, 038h, 06ch, 038h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h
368 000135D5 780C7CCCCC7600      <1>
369 000135DC 00000000000003C6660- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 060h, 066h, 03ch, 00ch, 006h, 03ch, 000h, 000h
369 000135E5 663C0C063C0000      <1>
370 000135EC 0010386C007CC6FEC0- <1>     db  000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
370 000135F5 C67C0000000000      <1>
371 000135FC CCCC007CC6FEC0C67C- <1>     db  0cch, 0cch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h
371 00013605 00000000603018      <1>
372 0001360C 007CC6FEC0C67C0000- <1>     db  000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 000h, 038h
372 00013615 00000066660038      <1>
373 0001361C 181818183C00000000- <1>     db  018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 018h, 03ch, 066h, 000h, 038h, 018h, 018h
373 00013625 183C6600381818      <1>
374 0001362C 18183C000000006030- <1>     db  018h, 018h, 03ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 038h, 018h, 018h, 018h, 018h
374 00013635 18003818181818      <1>
375 0001363C 3C00000000C6C61038- <1>     db  03ch, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 000h
375 00013645 6CC6C6FEC6C600      <1>
376 0001364C 0000386C3800386CC6- <1>     db  000h, 000h, 038h, 06ch, 038h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 000h, 000h, 000h
376 00013655 C6FEC6C6000000      <1>
377 0001365C 18306000FE66607C60- <1>     db  018h, 030h, 060h, 000h, 0feh, 066h, 060h, 07ch, 060h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h
377 00013665 66FE0000000000      <1>
378 0001366C 0000CC76367ED8D86E- <1>     db  000h, 000h, 0cch, 076h, 036h, 07eh, 0d8h, 0d8h, 06eh, 000h, 000h, 000h, 000h, 000h, 03eh, 06ch
378 00013675 00000000003E6C      <1>
379 0001367C CCCCFECCCCCCCE0000- <1>     db  0cch, 0cch, 0feh, 0cch, 0cch, 0cch, 0ceh, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 000h, 07ch
379 00013685 000010386C007C      <1>
380 0001368C C6C6C6C67C00000000- <1>     db  0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 000h, 07ch, 0c6h, 0c6h
380 00013695 00C6C6007CC6C6      <1>
381 0001369C C6C67C000000006030- <1>     db  0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h
381 000136A5 18007CC6C6C6C6      <1>
382 000136AC 7C000000003078CC00- <1>     db  07ch, 000h, 000h, 000h, 000h, 030h, 078h, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h
382 000136B5 CCCCCCCCCC7600      <1>
383 000136BC 00000060301800CCCC- <1>     db  000h, 000h, 000h, 060h, 030h, 018h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h
383 000136C5 CCCCCC76000000      <1>
384 000136CC 0000C6C600C6C6C6C6- <1>     db  000h, 000h, 0c6h, 0c6h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 078h, 000h, 000h, 0c6h
384 000136D5 7E060C780000C6      <1>
385 000136DC C6386CC6C6C6C66C38- <1>     db  0c6h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 000h
```

```
385 000136E5 00000000C6C600      <1>
386 000136EC C6C6C6C6C6C67C0000- <1>     db  0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 018h, 018h, 03ch, 066h, 060h
386 000136F5 000018183C6660      <1>
387 000136FC 60663C1818000000000- <1>     db  060h, 066h, 03ch, 018h, 018h, 000h, 000h, 000h, 000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h
387 00013705 386C6460F06060      <1>
388 0001370C 60E6FC000000000066- <1>     db  060h, 0e6h, 0fch, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 03ch, 018h, 07eh, 018h, 07eh, 018h
388 00013715 663C187E187E18      <1>
389 0001371C 1800000000F8CCCCF8- <1>     db  018h, 000h, 000h, 000h, 000h, 0f8h, 0cch, 0cch, 0f8h, 0c4h, 0cch, 0deh, 0cch, 0cch, 0c6h, 000h
389 00013725 C4CCDECCCCC600      <1>
390 0001372C 0000000E1B1818187E- <1>     db  000h, 000h, 000h, 00eh, 01bh, 018h, 018h, 018h, 07eh, 018h, 018h, 018h, 018h, 0d8h, 070h, 000h
390 00013735 18181818D87000      <1>
391 0001373C 0018306000780C7CCC- <1>     db  000h, 018h, 030h, 060h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 00ch
391 00013745 CC76000000000C      <1>
392 0001374C 183000381818183C- <1>     db  018h, 030h, 000h, 038h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 018h, 030h, 060h
392 00013755 00000000183060      <1>
393 0001375C 007CC6C6C67C0000- <1>     db  000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 018h, 030h, 060h, 000h, 0cch
393 00013765 000018306000CC      <1>
394 0001376C CCCCCCCC7600000000- <1>     db  0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 0dch, 066h, 066h
394 00013775 0076DC00DC6666      <1>
395 0001377C 66666600000076DC00- <1>     db  066h, 066h, 066h, 000h, 000h, 000h, 076h, 0dch, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h
395 00013785 C6E6F6FEDECEC6      <1>
396 0001378C C60000000003C6C6C3E- <1>     db  0c6h, 000h, 000h, 000h, 000h, 03ch, 06ch, 06ch, 03eh, 000h, 07eh, 000h, 000h, 000h, 000h, 000h
396 00013795 007E0000000000      <1>
397 0001379C 000000386C6C38007C- <1>     db  000h, 000h, 000h, 038h, 06ch, 06ch, 038h, 000h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h
397 000137A5 00000000000000      <1>
398 000137AC 00003030003030 60C6- <1>     db  000h, 000h, 030h, 030h, 000h, 030h, 030h, 060h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
398 000137B5 C67C0000000000      <1>
399 000137BC 00000000FEC0C0C000- <1>     db  000h, 000h, 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
399 000137C5 00000000000000      <1>
400 000137CC 0000FE060606000000- <1>     db  000h, 000h, 0feh, 006h, 006h, 006h, 000h, 000h, 000h, 000h, 000h, 0c0h, 0c0h, 0c6h, 0cch, 0d8h
400 000137D5 0000C0C0C6CCD8      <1>
401 000137DC 3060DC860C183E0000- <1>     db  030h, 060h, 0dch, 086h, 00ch, 018h, 03eh, 000h, 000h, 0c0h, 0c0h, 0c6h, 0cch, 0d8h, 030h, 066h
401 000137E5 C0C0C6CCD83066      <1>
402 000137EC CE9E3E0606000000018- <1>     db  0ceh, 09eh, 03eh, 006h, 006h, 000h, 000h, 000h, 018h, 018h, 000h, 018h, 018h, 03ch, 03ch, 03ch
402 000137F5 180018183C3C3C      <1>
403 000137FC 1800000000000000036- <1>     db  018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 036h, 06ch, 0d8h, 06ch, 036h, 000h, 000h, 000h
403 00013805 6CD86C36000000      <1>
404 0001380C 0000000000000D86C36- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 0d8h, 06ch, 036h, 06ch, 0d8h, 000h, 000h, 000h, 000h, 000h
404 00013815 6CD80000000000      <1>
405 0001381C 114411441144114411- <1>     db  011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 055h, 0aah
405 00013825 441144114455AA      <1>
406 0001382C 55AA55AA55AA55AA55- <1>     db  055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 0ddh, 077h, 0ddh, 077h
406 00013835 AA55AADD77DD77      <1>
407 0001383C DD77DD77DD77DD77DD- <1>     db  0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 018h, 018h, 018h, 018h, 018h, 018h
407 00013845 77181818181818      <1>
408 0001384C 181818181818181818- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h
408 00013855 181818181818F8      <1>
409 0001385C 181818181818181818- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h
409 00013865 1818F818F81818      <1>
410 0001386C 181818183636363636- <1>     db  018h, 018h, 018h, 018h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 036h
410 00013875 3636F636363636      <1>
411 0001387C 3636000000000000000- <1>     db  036h, 036h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h, 036h, 036h, 036h
411 00013885 FE363636363636      <1>
412 0001388C 0000000000F818F818- <1>     db  000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 036h, 036h
412 00013895 18181818183636      <1>
413 0001389C 363636F606F6363636- <1>     db  036h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
413 000138A5 36363636363636      <1>
414 000138AC 3636363636363636- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 000h, 0feh
414 000138B5 3600000000000FE      <1>
415 000138BC 06F636363636363636- <1>     db  006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0f6h, 006h, 0feh
415 000138C5 36363636F606FE      <1>
416 000138CC 0000000000000363636- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h
416 000138D5 36363636FE0000      <1>
417 000138DC 0000000001818181818- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 000h, 000h, 000h, 000h
417 000138E5 F818F800000000      <1>
418 000138EC 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h
418 000138F5 F8181818181818      <1>
419 000138FC 181818181818181F00- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h
419 00013905 00000000001818      <1>
420 0001390C 1818181818FF000000- <1>     db  018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
420 00013915 00000000000000      <1>
421 0001391C 000000FF1818181818- <1>     db  000h, 000h, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
421 00013925 18181818181818      <1>
422 0001392C 181F181818181800- <1>     db  018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh
422 00013935 0000000000FF      <1>
423 0001393C 0000000000181818- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h
423 00013945 18181818FF1818      <1>
424 0001394C 181818181818181818- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh, 018h, 018h, 018h, 018h
424 00013955 1F181F18181818      <1>
425 0001395C 1818363636363636- <1>     db  018h, 018h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 037h, 036h, 036h, 036h, 036h, 036h, 036h
425 00013965 37363636363636      <1>
426 0001396C 3636363637303F00- <1>     db  036h, 036h, 036h, 036h, 036h, 037h, 030h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
426 00013975 00000000000000      <1>
427 0001397C 0000003F3037363636- <1>     db  000h, 000h, 000h, 03fh, 030h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
427 00013985 36363636363636      <1>
428 0001398C 36F700FF00000000000- <1>     db  036h, 0f7h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh
428 00013995 0000000000FF      <1>
429 0001399C 00F736363636363636- <1>     db  000h, 0f7h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 037h, 030h, 037h
429 000139A5 36363636373037      <1>
430 000139AC 3636363636000000- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 000h, 000h
430 000139B5 0000FF00FF0000      <1>
431 000139BC 0000000036363636- <1>     db  000h, 000h, 000h, 000h, 036h, 036h, 036h, 036h, 036h, 0f7h, 000h, 0f7h, 036h, 036h, 036h, 036h
431 000139C5 F700F736363636      <1>
432 000139CC 36361818181818FF00- <1>     db  036h, 036h, 018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h
432 000139D5 FF000000000000      <1>
433 000139DC 36363636363636FF00- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
433 000139E5 00000000000000      <1>
434 000139EC 000000FF00FF181818- <1>     db  000h, 000h, 000h, 0ffh, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h
434 000139F5 18181800000000      <1>
435 000139FC 000000FF3636363636- <1>     db  000h, 000h, 000h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
435 00013A05 36363636363636      <1>
436 00013A0C 363F00000000000018- <1>     db  036h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh
436 00013A15 181818181F181F      <1>
437 00013A1C 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h
437 00013A25 00001F181F1818      <1>
438 00013A2C 1818181800000000000- <1>     db  018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 03fh, 036h, 036h, 036h, 036h
438 00013A35 00003F36363636      <1>
439 00013A3C 3636363636363636- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h
```

```
439 00013A45 FF363636363636      <1>
440 00013A4C 1818181818FF18FF18- <1>     db  018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
440 00013A55 18181818181818      <1>
441 00013A5C 1818181818F8000000- <1>     db  018h, 018h, 018h, 018h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
441 00013A65 00000000000000      <1>
442 00013A6C 0000001F1818181818- <1>     db  000h, 000h, 000h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
442 00013A75 18FFFFFFFFFFFF      <1>
443 00013A7C FFFFFFFFFFFFFFFF00- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh
443 00013A85 0000000000000FF     <1>
444 00013A8C FFFFFFFFFFFF0F0F0- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h
444 00013A95 F0F0F0F0F0F0F0      <1>
445 00013A9C F0F0F0F00F0F0F0F0F- <1>     db  0f0h, 0f0h, 0f0h, 0f0h, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh
445 00013AA5 0F0F0F0F0F0F0F      <1>
446 00013AAC 0F0FFFFFFFFFFFFFFF- <1>     db  00fh, 00fh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h
446 00013AB5 00000000000000      <1>
447 00013ABC 000000000076DCD8D8- <1>     db  000h, 000h, 000h, 000h, 000h, 076h, 0dch, 0d8h, 0d8h, 0dch, 076h, 000h, 000h, 000h, 000h, 000h
447 00013AC5 DC760000000000      <1>
448 00013ACC 00007CC6FCC6C6FCC0- <1>     db  000h, 000h, 07ch, 0c6h, 0fch, 0c6h, 0c6h, 0fch, 0c0h, 0c0h, 040h, 000h, 000h, 000h, 0feh, 0c6h
448 00013AD5 C040000000FEC6      <1>
449 00013ADC C6C0C0C0C0C0C00000- <1>     db  0c6h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 06ch
449 00013AE5 0000000000FE6C      <1>
450 00013AEC 6C6C6C6C6C00000000- <1>     db  06ch, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h, 000h, 0feh, 0c6h, 060h, 030h, 018h, 030h
450 00013AF5 00FEC660301830      <1>
451 00013AFC 60C6FE000000000000- <1>     db  060h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 0d8h, 0d8h, 0d8h, 0d8h
451 00013B05 00007ED8D8D8D8      <1>
452 00013B0C 7000000000000000066- <1>     db  070h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h, 07ch, 060h, 060h, 0c0h
452 00013B15 6666667C6060C0      <1>
453 00013B1C 00000000000076DC18- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h
453 00013B25 18181818000000      <1>
454 00013B2C 00007E183C6666663C- <1>     db  000h, 000h, 07eh, 018h, 03ch, 066h, 066h, 066h, 03ch, 018h, 07eh, 000h, 000h, 000h, 000h, 000h
454 00013B35 187E0000000000      <1>
455 00013B3C 386CC6C6FEC6C66C38- <1>     db  038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 038h, 06ch
455 00013B45 0000000000386C      <1>
456 00013B4C C6C6C66C6C6CEE0000- <1>     db  0c6h, 0c6h, 0c6h, 06ch, 06ch, 06ch, 0eeh, 000h, 000h, 000h, 000h, 000h, 01eh, 030h, 018h, 00ch
456 00013B55 0000001E30180C      <1>
457 00013B5C 3E6666663C00000000- <1>     db  03eh, 066h, 066h, 066h, 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 0dbh, 0dbh
457 00013B65 000000007EDBDB      <1>
458 00013B6C 7E00000000000000003- <1>     db  07eh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 003h, 006h, 07eh, 0dbh, 0dbh, 0f3h, 07eh, 060h
458 00013B75 067EDBDBF37E60      <1>
459 00013B7C C000000000001C3060- <1>     db  0c0h, 000h, 000h, 000h, 000h, 000h, 01ch, 030h, 060h, 060h, 07ch, 060h, 060h, 030h, 01ch, 000h
459 00013B85 607C6060301C00      <1>
460 00013B8C 00000000007CC6C6C6- <1>     db  000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h
460 00013B95 C6C6C6C6000000      <1>
461 00013B9C 000000FE0000FE0000- <1>     db  000h, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h
461 00013BA5 FE000000000000      <1>
462 00013BAC 0018187E18180000FF- <1>     db  000h, 018h, 018h, 07eh, 018h, 018h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 030h, 018h
462 00013BB5 00000000003018      <1>
463 00013BBC 0C060C1830007E0000- <1>     db  00ch, 006h, 00ch, 018h, 030h, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 00ch, 018h, 030h, 060h
463 00013BC5 0000000C183060      <1>
464 00013BCC 30180C007E00000000- <1>     db  030h, 018h, 00ch, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 00eh, 01bh, 01bh, 018h, 018h, 018h
464 00013BD5 000E1B1B181818      <1>
465 00013BDC 181818181818181818- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0d8h, 0d8h
465 00013BE5 1818181818D8D8      <1>
466 00013BEC 700000000000001818- <1>     db  070h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 07eh, 000h, 018h, 018h, 000h, 000h
466 00013BF5 007E0018180000      <1>
467 00013BFC 00000000000076DC00- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h
467 00013C05 76DC0000000000      <1>
468 00013C0C 00386C6C3800000000- <1>     db  000h, 038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
468 00013C15 00000000000000      <1>
469 00013C1C 000000001818000000- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
469 00013C25 00000000000000      <1>
470 00013C2C 0000001800000000000- <1>     db  000h, 000h, 000h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 00fh, 00ch, 00ch, 00ch, 00ch
470 00013C35 00000F0C0C0C0C      <1>
471 00013C3C 0CEC6C3C1C00000000- <1>     db  00ch, 0ech, 06ch, 03ch, 01ch, 000h, 000h, 000h, 000h, 0d8h, 06ch, 06ch, 06ch, 06ch, 06ch, 000h
471 00013C45 D86C6C6C6C6C00      <1>
472 00013C4C 000000000000070D8- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 070h, 0d8h, 030h, 060h, 0c8h, 0f8h, 000h, 000h, 000h
472 00013C55 3060C8F8000000      <1>
473 00013C5C 00000000000000007C- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 000h, 000h
473 00013C65 7C7C7C7C7C0000      <1>
474 00013C6C 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
474 00013C75 00000000000000      <1>
475                              <1> vgafont16:
476 00013C7C 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
476 00013C85 00000000000000      <1>
477 00013C8C 00007E81A58181BD99- <1>     db  000h, 000h, 07eh, 081h, 0a5h, 081h, 081h, 0bdh, 099h, 081h, 081h, 07eh, 000h, 000h, 000h, 000h
477 00013C95 81817E00000000      <1>
478 00013C9C 00007EFFDBFFFFC3E7- <1>     db  000h, 000h, 07eh, 0ffh, 0dbh, 0ffh, 0ffh, 0c3h, 0e7h, 0ffh, 0ffh, 07eh, 000h, 000h, 000h, 000h
478 00013CA5 FFFF7E00000000      <1>
479 00013CAC 000000006CFEFEFEFE- <1>     db  000h, 000h, 000h, 000h, 06ch, 0feh, 0feh, 0feh, 0feh, 07ch, 038h, 010h, 000h, 000h, 000h, 000h
479 00013CB5 7C381000000000      <1>
480 00013CBC 0000000010387CFE7C- <1>     db  000h, 000h, 000h, 000h, 010h, 038h, 07ch, 0feh, 07ch, 038h, 010h, 000h, 000h, 000h, 000h, 000h
480 00013CC5 38100000000000      <1>
481 00013CCC 000000183C3CE7E7E7- <1>     db  000h, 000h, 000h, 018h, 03ch, 03ch, 0e7h, 0e7h, 0e7h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
481 00013CD5 18183C00000000      <1>
482 00013CDC 000000183C7EFFFF7E- <1>     db  000h, 000h, 000h, 018h, 03ch, 07eh, 0ffh, 0ffh, 07eh, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
482 00013CE5 18183C00000000      <1>
483 00013CEC 000000000000183C3C- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 03ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h
483 00013CF5 18000000000000      <1>
484 00013CFC FFFFFFFFFFFE7C3C3- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0e7h, 0c3h, 0c3h, 0e7h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
484 00013D05 E7FFFFFFFFFFFF      <1>
485 00013D0C 000000000003C664242- <1>     db  000h, 000h, 000h, 000h, 000h, 03ch, 066h, 042h, 042h, 066h, 03ch, 000h, 000h, 000h, 000h, 000h
485 00013D15 663C0000000000      <1>
486 00013D1C FFFFFFFFFFC399BDBD- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0c3h, 099h, 0bdh, 0bdh, 099h, 0c3h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
486 00013D25 99C3FFFFFFFFFF      <1>
487 00013D2C 00001E0E1A3278CCCC- <1>     db  000h, 000h, 01eh, 00eh, 01ah, 032h, 078h, 0cch, 0cch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h
487 00013D35 CCCC7800000000      <1>
488 00013D3C 00003C666666663C18- <1>     db  000h, 000h, 03ch, 066h, 066h, 066h, 066h, 03ch, 018h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h
488 00013D45 7E181800000000      <1>
489 00013D4C 00003F333F30303030- <1>     db  000h, 000h, 03fh, 033h, 03fh, 030h, 030h, 030h, 030h, 070h, 0f0h, 0e0h, 000h, 000h, 000h, 000h
489 00013D55 70F0E000000000      <1>
490 00013D5C 00007F637F63636363- <1>     db  000h, 000h, 07fh, 063h, 07fh, 063h, 063h, 063h, 063h, 067h, 0e7h, 0e6h, 0c0h, 000h, 000h, 000h
490 00013D65 67E7E6C0000000      <1>
491 00013D6C 0000001818DB3CE73C- <1>     db  000h, 000h, 000h, 018h, 018h, 0dbh, 03ch, 0e7h, 03ch, 0dbh, 018h, 018h, 000h, 000h, 000h, 000h
491 00013D75 DB181800000000      <1>
492 00013D7C 0080C0E0F0F8FEF8F0- <1>     db  000h, 080h, 0c0h, 0e0h, 0f0h, 0f8h, 0feh, 0f8h, 0f0h, 0e0h, 0c0h, 080h, 000h, 000h, 000h, 000h
492 00013D85 E0C080000000000     <1>
493 00013D8C 0002060E1E3EFE3E1E- <1>     db  000h, 002h, 006h, 00eh, 01eh, 03eh, 0feh, 03eh, 01eh, 00eh, 006h, 002h, 000h, 000h, 000h, 000h
493 00013D95 0E060200000000      <1>
```

```
494 00013D9C 0000183C7E1818187E- <1>     db  000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h
494 00013DA5 3C180000000000      <1>
495 00013DAC 0006666666666666- <1>     db  000h, 000h, 066h, 066h, 066h, 066h, 066h, 066h, 066h, 000h, 066h, 066h, 000h, 000h, 000h, 000h
495 00013DB5 0066660000000000      <1>
496 00013DBC 00007FDBDBDB7B1B1B- <1>     db  000h, 000h, 07fh, 0dbh, 0dbh, 0dbh, 07bh, 01bh, 01bh, 01bh, 01bh, 01bh, 000h, 000h, 000h, 000h
496 00013DC5 1B1B1B00000000      <1>
497 00013DCC 007CC660386CC6C66C- <1>     db  000h, 07ch, 0c6h, 060h, 038h, 06ch, 0c6h, 0c6h, 06ch, 038h, 00ch, 0c6h, 07ch, 000h, 000h, 000h
497 00013DD5 380CC67C000000      <1>
498 00013DDC 00000000000000FE- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 0feh, 0feh, 0feh, 000h, 000h, 000h, 000h
498 00013DE5 FEFEFE00000000      <1>
499 00013DEC 0000183C7E1818187E- <1>     db  000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 07eh, 000h, 000h, 000h, 000h
499 00013DF5 3C187E00000000      <1>
500 00013DFC 0000183C7E18181818- <1>     db  000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h
500 00013E05 18181800000000      <1>
501 00013E0C 000018181818181818- <1>     db  000h, 000h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h
501 00013E15 7E3C1800000000      <1>
502 00013E1C 0000000000180CFE0C- <1>     db  000h, 000h, 000h, 000h, 000h, 018h, 00ch, 0feh, 00ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h
502 00013E25 18000000000000      <1>
503 00013E2C 00000000003060FE60- <1>     db  000h, 000h, 000h, 000h, 000h, 030h, 060h, 0feh, 060h, 030h, 000h, 000h, 000h, 000h, 000h, 000h
503 00013E35 30000000000000      <1>
504 00013E3C 0000000000000C0C0C0- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 0c0h, 0c0h, 0c0h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h
504 00013E45 FE0000000000000      <1>
505 00013E4C 00000000002466FF66- <1>     db  000h, 000h, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h, 000h, 000h, 000h, 000h
505 00013E55 24000000000000      <1>
506 00013E5C 000000001038387C7C- <1>     db  000h, 000h, 000h, 000h, 010h, 038h, 038h, 07ch, 07ch, 0feh, 0feh, 000h, 000h, 000h, 000h, 000h
506 00013E65 FEFE0000000000      <1>
507 00013E6C 00000000FEFE7C7C38- <1>     db  000h, 000h, 000h, 000h, 0feh, 0feh, 07ch, 07ch, 038h, 038h, 010h, 000h, 000h, 000h, 000h, 000h
507 00013E75 381000000000000      <1>
508 00013E7C 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
508 00013E85 00000000000000      <1>
509 00013E8C 0000183C3C3C181818- <1>     db  000h, 000h, 018h, 03ch, 03ch, 03ch, 018h, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h, 000h
509 00013E95 00181800000000      <1>
510 00013E9C 0066666624000000000- <1>     db  000h, 066h, 066h, 066h, 024h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
510 00013EA5 000000000000000      <1>
511 00013EAC 0000006C6CFE6C6C6C- <1>     db  000h, 000h, 000h, 06ch, 06ch, 0feh, 06ch, 06ch, 06ch, 0feh, 06ch, 06ch, 000h, 000h, 000h, 000h
511 00013EB5 FE6C6C00000000      <1>
512 00013EBC 18187CC6C2C07C0606- <1>     db  018h, 018h, 07ch, 0c6h, 0c2h, 0c0h, 07ch, 006h, 006h, 086h, 0c6h, 07ch, 018h, 018h, 000h, 000h
512 00013EC5 86C67C18180000      <1>
513 00013ECC 00000000C2C60C1830- <1>     db  000h, 000h, 000h, 000h, 0c2h, 0c6h, 00ch, 018h, 030h, 060h, 0c6h, 086h, 000h, 000h, 000h, 000h
513 00013ED5 60C686000000000      <1>
514 00013EDC 0000386C6C3876DCCC- <1>     db  000h, 000h, 038h, 06ch, 06ch, 038h, 076h, 0dch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
514 00013EE5 CCCC7600000000      <1>
515 00013EEC 0030303060000000000- <1>     db  000h, 030h, 030h, 030h, 060h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
515 00013EF5 000000000000000      <1>
516 00013EFC 00000C183030303030- <1>     db  000h, 000h, 00ch, 018h, 030h, 030h, 030h, 030h, 030h, 030h, 018h, 00ch, 000h, 000h, 000h, 000h
516 00013F05 30180C00000000      <1>
517 00013F0C 000030180C0C0C0C0C- <1>     db  000h, 000h, 030h, 018h, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 018h, 030h, 000h, 000h, 000h, 000h
517 00013F15 0C183000000000      <1>
518 00013F1C 0000000000663CFF3C- <1>     db  000h, 000h, 000h, 000h, 000h, 066h, 03ch, 0ffh, 03ch, 066h, 000h, 000h, 000h, 000h, 000h, 000h
518 00013F25 660000000000000      <1>
519 00013F2C 0000000000018187E18- <1>     db  000h, 000h, 000h, 000h, 000h, 018h, 018h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h
519 00013F35 18000000000000      <1>
520 00013F3C 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 030h, 000h, 000h, 000h
520 00013F45 18181830000000      <1>
521 00013F4C 00000000000000FE00- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
521 00013F55 000000000000000      <1>
522 00013F5C 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h
522 00013F65 00181800000000      <1>
523 00013F6C 0000000002060C1830- <1>     db  000h, 000h, 000h, 000h, 002h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 080h, 000h, 000h, 000h, 000h
523 00013F75 60C08000000000      <1>
524 00013F7C 00003C66C3C3DBDBC3- <1>     db  000h, 000h, 03ch, 066h, 0c3h, 0c3h, 0dbh, 0dbh, 0c3h, 0c3h, 066h, 03ch, 000h, 000h, 000h, 000h
524 00013F85 C3663C00000000      <1>
525 00013F8C 000018387818181818- <1>     db  000h, 000h, 018h, 038h, 078h, 018h, 018h, 018h, 018h, 018h, 018h, 07eh, 000h, 000h, 000h, 000h
525 00013F95 18187E00000000      <1>
526 00013F9C 00007CC6060C183060- <1>     db  000h, 000h, 07ch, 0c6h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 0c6h, 0feh, 000h, 000h, 000h, 000h
526 00013FA5 C0C6FE00000000      <1>
527 00013FAC 00007CC606063C0606- <1>     db  000h, 000h, 07ch, 0c6h, 006h, 006h, 03ch, 006h, 006h, 006h, 0c6h, 07ch, 000h, 000h, 000h, 000h
527 00013FB5 06C67C00000000      <1>
528 00013FBC 00000C1C3C6CCCFE0C- <1>     db  000h, 000h, 00ch, 01ch, 03ch, 06ch, 0cch, 0feh, 00ch, 00ch, 00ch, 01eh, 000h, 000h, 000h, 000h
528 00013FC5 0C0C1E00000000      <1>
529 00013FCC 0000FEC0C0C0FC0606- <1>     db  000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 0fch, 006h, 006h, 006h, 0c6h, 07ch, 000h, 000h, 000h, 000h
529 00013FD5 06C67C00000000      <1>
530 00013FDC 00003860C0C0FCC6C6- <1>     db  000h, 000h, 038h, 060h, 0c0h, 0c0h, 0fch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
530 00013FE5 C6C67C00000000      <1>
531 00013FEC 0000FEC606060C1830- <1>     db  000h, 000h, 0feh, 0c6h, 006h, 006h, 00ch, 018h, 030h, 030h, 030h, 030h, 000h, 000h, 000h, 000h
531 00013FF5 303030000000000      <1>
532 00013FFC 00007CC6C6C67CC6C6- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
532 00014005 C6C67C00000000      <1>
533 0001400C 00007CC6C6C67E0606- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 07eh, 006h, 006h, 006h, 00ch, 078h, 000h, 000h, 000h, 000h
533 00014015 060C7800000000      <1>
534 0001401C 000000001818000000- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h
534 00014025 18180000000000      <1>
535 0001402C 000000001818000000- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 018h, 018h, 030h, 000h, 000h, 000h, 000h
535 00014035 18183000000000      <1>
536 0001403C 0000000060C18306030- <1>     db  000h, 000h, 000h, 006h, 00ch, 018h, 030h, 060h, 030h, 018h, 00ch, 006h, 000h, 000h, 000h, 000h
536 00014045 180C0600000000      <1>
537 0001404C 00000000007E00007E- <1>     db  000h, 000h, 000h, 000h, 000h, 07eh, 000h, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 000h
537 00014055 000000000000000      <1>
538 0001405C 0000006030180C060C- <1>     db  000h, 000h, 000h, 060h, 030h, 018h, 00ch, 006h, 00ch, 018h, 030h, 060h, 000h, 000h, 000h, 000h
538 00014065 183060000000000      <1>
539 0001406C 00007CC6C60C181818- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 00ch, 018h, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h, 000h
539 00014075 00181800000000      <1>
540 0001407C 0000007CC6C6DEDEDE- <1>     db  000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0deh, 0deh, 0deh, 0dch, 0c0h, 07ch, 000h, 000h, 000h, 000h
540 00014085 DCC07C00000000      <1>
541 0001408C 000010386CC6C6FEC6- <1>     db  000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
541 00014095 C6C6C600000000      <1>
542 0001409C 0000FC6666667C6666- <1>     db  000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 066h, 066h, 066h, 066h, 0fch, 000h, 000h, 000h, 000h
542 000140A5 6666FC00000000      <1>
543 000140AC 00003C66C2C0C0C0C0- <1>     db  000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0c0h, 0c0h, 0c2h, 066h, 03ch, 000h, 000h, 000h, 000h
543 000140B5 C2663C00000000      <1>
544 000140BC 0000F86C6666666666- <1>     db  000h, 000h, 0f8h, 06ch, 066h, 066h, 066h, 066h, 066h, 066h, 06ch, 0f8h, 000h, 000h, 000h, 000h
544 000140C5 666CF800000000      <1>
545 000140CC 0000FE666268786860- <1>     db  000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 060h, 062h, 066h, 0feh, 000h, 000h, 000h, 000h
545 000140D5 6266FE00000000      <1>
546 000140DC 0000FE666268786860- <1>     db  000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h
546 000140E5 6060F000000000      <1>
547 000140EC 00003C66C2C0C0DEC6- <1>     db  000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0deh, 0c6h, 0c6h, 066h, 03ah, 000h, 000h, 000h, 000h
547 000140F5 C6663A00000000      <1>
```

```
548 000140FC 0000C6C6C6C6FEC6C6- <1>     db  000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
548 00014105 C6C6C600000000      <1>
549 0001410C 00003C181818181818- <1>     db  000h, 000h, 03ch, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
549 00014115 18183C00000000      <1>
550 0001411C 00001E0C0C0C0C0CCC- <1>     db  000h, 000h, 01eh, 00ch, 00ch, 00ch, 00ch, 00ch, 0cch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h
550 00014125 CCCC7800000000      <1>
551 0001412C 0000E666666C78786C- <1>     db  000h, 000h, 0e6h, 066h, 066h, 06ch, 078h, 078h, 06ch, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h
551 00014135 6666E600000000      <1>
552 0001413C 0000F06060606060060- <1>     db  000h, 000h, 0f0h, 060h, 060h, 060h, 060h, 060h, 060h, 062h, 066h, 0feh, 000h, 000h, 000h, 000h
552 00014145 6266FE00000000      <1>
553 0001414C 0000C3E7FFFFDBC3C3- <1>     db  000h, 000h, 0c3h, 0e7h, 0ffh, 0ffh, 0dbh, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 000h, 000h, 000h, 000h
553 00014155 C3C3C300000000      <1>
554 0001415C 0000C6E6F6FEDECEC6- <1>     db  000h, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
554 00014165 C6C6C600000000      <1>
555 0001416C 00007CC6C6C6C6C6C6- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
555 00014175 C6C67C00000000      <1>
556 0001417C 0000FC6666667C6060- <1>     db  000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 060h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h
556 00014185 6060F000000000      <1>
557 0001418C 00007CC6C6C6C6C6C6- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0d6h, 0deh, 07ch, 00ch, 00eh, 000h, 000h
557 00014195 D6DE7C0C0E0000      <1>
558 0001419C 0000FC6666667C6C66- <1>     db  000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 06ch, 066h, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h
558 000141A5 6666E600000000      <1>
559 000141AC 00007CC6C660380C06- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 060h, 038h, 00ch, 006h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
559 000141B5 C6C67C00000000      <1>
560 000141BC 0000FFDB9918181818- <1>     db  000h, 000h, 0ffh, 0dbh, 099h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
560 000141C5 18183C00000000      <1>
561 000141CC 0000C6C6C6C6C6C6C6- <1>     db  000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
561 000141D5 C6C67C00000000      <1>
562 000141DC 0000C3C3C3C3C3C3C3- <1>     db  000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 000h
562 000141E5 663C1800000000      <1>
563 000141EC 0000C3C3C3C3C3DBDB- <1>     db  000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 066h, 000h, 000h, 000h, 000h
563 000141F5 FF666600000000      <1>
564 000141FC 0000C3C3663C18183C- <1>     db  000h, 000h, 0c3h, 0c3h, 066h, 03ch, 018h, 018h, 03ch, 066h, 0c3h, 0c3h, 000h, 000h, 000h, 000h
564 00014205 66C3C300000000      <1>
565 0001420C 0000C3C3C3663C1818- <1>     db  000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
565 00014215 18183C00000000      <1>
566 0001421C 0000FFC3860C183060- <1>     db  000h, 000h, 0ffh, 0c3h, 086h, 00ch, 018h, 030h, 060h, 0c1h, 0c3h, 0ffh, 000h, 000h, 000h, 000h
566 00014225 C1C3FF00000000      <1>
567 0001422C 00003C303030303030- <1>     db  000h, 000h, 03ch, 030h, 030h, 030h, 030h, 030h, 030h, 030h, 030h, 03ch, 000h, 000h, 000h, 000h
567 00014235 30303C00000000      <1>
568 0001423C 00000080C0E070381C- <1>     db  000h, 000h, 000h, 080h, 0c0h, 0e0h, 070h, 038h, 01ch, 00eh, 006h, 002h, 000h, 000h, 000h, 000h
568 00014245 0E060200000000      <1>
569 0001424C 00003C0C0C0C0C0C0C- <1>     db  000h, 000h, 03ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 03ch, 000h, 000h, 000h, 000h
569 00014255 0C0C3C00000000      <1>
570 0001425C 10386CC600000000000- <1>     db  010h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
570 00014265 000000000000000     <1>
571 0001426C 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 000h
571 00014275 00000000FF0000      <1>
572 0001427C 3030180000000000000- <1>     db  030h, 030h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
572 00014285 000000000000000     <1>
573 0001428C 0000000000780C7CCC- <1>     db  000h, 000h, 000h, 000h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
573 00014295 CCCC7600000000      <1>
574 0001429C 0000E06060786C6666- <1>     db  000h, 000h, 0e0h, 060h, 060h, 078h, 06ch, 066h, 066h, 066h, 066h, 07ch, 000h, 000h, 000h, 000h
574 000142A5 66667C00000000      <1>
575 000142AC 00000000007CC6C0C0- <1>     db  000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c0h, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
575 000142B5 C0C67C00000000      <1>
576 000142BC 00001C0C0C3C6CCCCC- <1>     db  000h, 000h, 01ch, 00ch, 00ch, 03ch, 06ch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
576 000142C5 CCCC7600000000      <1>
577 000142CC 00000000007CC6FEC0- <1>     db  000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
577 000142D5 C0C67C00000000      <1>
578 000142DC 0000386C6460F06060- <1>     db  000h, 000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h
578 000142E5 6060F000000000      <1>
579 000142EC 000000000076CCCCCC- <1>     db  000h, 000h, 000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 0cch, 0cch, 07ch, 00ch, 0cch, 078h, 000h
579 000142F5 CCCC7C0CCC7800      <1>
580 000142FC 0000E060606C766666- <1>     db  000h, 000h, 0e0h, 060h, 060h, 06ch, 076h, 066h, 066h, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h
580 00014305 6666E600000000      <1>
581 0001430C 0000181800381818 18- <1>     db  000h, 000h, 018h, 018h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
581 00014315 18183C00000000      <1>
582 0001431C 00000606000E060606- <1>     db  000h, 000h, 006h, 006h, 000h, 00eh, 006h, 006h, 006h, 006h, 006h, 006h, 066h, 066h, 03ch, 000h
582 00014325 06060666663C00      <1>
583 0001432C 0000E06060666C7878- <1>     db  000h, 000h, 0e0h, 060h, 060h, 066h, 06ch, 078h, 078h, 06ch, 066h, 0e6h, 000h, 000h, 000h, 000h
583 00014335 6C66E600000000      <1>
584 0001433C 0000381818181818 18- <1>     db  000h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
584 00014345 18183C00000000      <1>
585 0001434C 0000000000E6FFDBDB- <1>     db  000h, 000h, 000h, 000h, 000h, 0e6h, 0ffh, 0dbh, 0dbh, 0dbh, 0dbh, 0dbh, 000h, 000h, 000h, 000h
585 00014355 DBDBDB00000000      <1>
586 0001435C 0000000000DC666666- <1>     db  000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 066h, 000h, 000h, 000h, 000h
586 00014365 66666600000000      <1>
587 0001436C 00000000007CC6C6C6- <1>     db  000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
587 00014375 C6C67C00000000      <1>
588 0001437C 0000000000DC666666- <1>     db  000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 07ch, 060h, 060h, 0f0h, 000h
588 00014385 66667C6060F000      <1>
589 0001438C 000000000076CCCCCC- <1>     db  000h, 000h, 000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 0cch, 0cch, 07ch, 00ch, 00ch, 01eh, 000h
589 00014395 CCCC7C0C0C1E00      <1>
590 0001439C 0000000000DC766660- <1>     db  000h, 000h, 000h, 000h, 000h, 0dch, 076h, 066h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h
590 000143A5 6060F000000000      <1>
591 000143AC 00000000007CC66038- <1>     db  000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 060h, 038h, 00ch, 0c6h, 07ch, 000h, 000h, 000h, 000h
591 000143B5 0CC67C00000000      <1>
592 000143BC 0000103030FC303030- <1>     db  000h, 000h, 010h, 030h, 030h, 0fch, 030h, 030h, 030h, 030h, 036h, 01ch, 000h, 000h, 000h, 000h
592 000143C5 30361C00000000      <1>
593 000143CC 0000000000CCCCCCCC- <1>     db  000h, 000h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
593 000143D5 CCCC7600000000      <1>
594 000143DC 0000000000C3C3C3C3- <1>     db  000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 000h
594 000143E5 663C1800000000      <1>
595 000143EC 0000000000C3C3C3DB- <1>     db  000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h, 000h, 000h, 000h
595 000143F5 DBFF6600000000      <1>
596 000143FC 0000000000C3663C18- <1>     db  000h, 000h, 000h, 000h, 000h, 0c3h, 066h, 03ch, 018h, 03ch, 066h, 0c3h, 000h, 000h, 000h, 000h
596 00014405 3C66C300000000      <1>
597 0001440C 0000000000C6C6C6C6- <1>     db  000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 0f8h, 000h
597 00014415 C6C67E060CF800      <1>
598 0001441C 0000000000FECC1830- <1>     db  000h, 000h, 000h, 000h, 000h, 0feh, 0cch, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h
598 00014425 60C6FE00000000      <1>
599 0001442C 00000E181818701818- <1>     db  000h, 000h, 00eh, 018h, 018h, 018h, 070h, 018h, 018h, 018h, 018h, 00eh, 000h, 000h, 000h, 000h
599 00014435 18180E00000000      <1>
600 0001443C 0000181818180018 18- <1>     db  000h, 000h, 018h, 018h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h
600 00014445 18181800000000      <1>
601 0001444C 0000701818180E1818- <1>     db  000h, 000h, 070h, 018h, 018h, 018h, 00eh, 018h, 018h, 018h, 018h, 070h, 000h, 000h, 000h, 000h
601 00014455 18187000000000      <1>
```

```
602 0001445C 000076DC0000000000- <1>    db  000h, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
602 00014465 00000000000000      <1>
603 0001446C 0000000010386CC6C6- <1>    db  000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h
603 00014475 C6FE0000000000      <1>
604 0001447C 00003C66C2C0C0C0C2- <1>    db  000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0c0h, 0c2h, 066h, 03ch, 00ch, 006h, 07ch, 000h, 000h
604 00014485 663C0C067C0000      <1>
605 0001448C 0000CC0000CCCCCCCC- <1>    db  000h, 000h, 0cch, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
605 00014495 CCCC7600000000      <1>
606 0001449C 000C1830007CC6FEC0- <1>    db  000h, 00ch, 018h, 030h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
606 000144A5 C0C67C00000000      <1>
607 000144AC 0010386C00780C7CCC- <1>    db  000h, 010h, 038h, 06ch, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
607 000144B5 CCCC7600000000      <1>
608 000144BC 0000CC0000780C7CCC- <1>    db  000h, 000h, 0cch, 000h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
608 000144C5 CCCC7600000000      <1>
609 000144CC 0060301800780C7CCC- <1>    db  000h, 060h, 030h, 018h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
609 000144D5 CCCC7600000000      <1>
610 000144DC 00386C3800780C7CCC- <1>    db  000h, 038h, 06ch, 038h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
610 000144E5 CCCC7600000000      <1>
611 000144EC 000000003C66606066- <1>    db  000h, 000h, 000h, 000h, 03ch, 066h, 060h, 060h, 066h, 03ch, 00ch, 006h, 03ch, 000h, 000h, 000h
611 000144F5 3C0C063C000000      <1>
612 000144FC 0010386C007CC6FEC0- <1>    db  000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
612 00014505 C0C67C00000000      <1>
613 0001450C 0000C600007CC6FEC0- <1>    db  000h, 000h, 0c6h, 000h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
613 00014515 C0C67C00000000      <1>
614 0001451C 00603018007CC6FEC0- <1>    db  000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
614 00014525 C0C67C00000000      <1>
615 0001452C 000066000038181818- <1>    db  000h, 000h, 066h, 000h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
615 00014535 18183C00000000      <1>
616 0001453C 00183C660038181818- <1>    db  000h, 018h, 03ch, 066h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
616 00014545 18183C00000000      <1>
617 0001454C 006030180038181818- <1>    db  000h, 060h, 030h, 018h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
617 00014555 18183C00000000      <1>
618 0001455C 00C60010386CC6C6FE- <1>    db  000h, 0c6h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
618 00014565 C6C6C600000000      <1>
619 0001456C 386C3800386CC6C6FE- <1>    db  038h, 06ch, 038h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
619 00014575 C6C6C600000000      <1>
620 0001457C 18306000FE66607C60- <1>    db  018h, 030h, 060h, 000h, 0feh, 066h, 060h, 07ch, 060h, 060h, 066h, 0feh, 000h, 000h, 000h, 000h
620 00014585 6066FE00000000      <1>
621 0001458C 00000000006E3B1B7E- <1>    db  000h, 000h, 000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h, 000h, 000h
621 00014595 D8DC7700000000      <1>
622 0001459C 00003E6CCCCCFECCCC- <1>    db  000h, 000h, 03eh, 06ch, 0cch, 0cch, 0feh, 0cch, 0cch, 0cch, 0cch, 0ceh, 000h, 000h, 000h, 000h
622 000145A5 CCCCCE00000000      <1>
623 000145AC 0010386C007CC6C6C6- <1>    db  000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
623 000145B5 C6C67C00000000      <1>
624 000145BC 0000C600007CC6C6C6- <1>    db  000h, 000h, 0c6h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
624 000145C5 C6C67C00000000      <1>
625 000145CC 00603018007CC6C6C6- <1>    db  000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
625 000145D5 C6C67C00000000      <1>
626 000145DC 003078CC00CCCCCCCC- <1>    db  000h, 030h, 078h, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
626 000145E5 CCCC7600000000      <1>
627 000145EC 0060301800CCCCCCCC- <1>    db  000h, 060h, 030h, 018h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
627 000145F5 CCCC7600000000      <1>
628 000145FC 0000C60000C6C6C6C6- <1>    db  000h, 000h, 0c6h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 078h, 000h
628 00014605 C6C67E060C7800      <1>
629 0001460C 00C6007CC6C6C6C6C6- <1>    db  000h, 0c6h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
629 00014615 C6C67C00000000      <1>
630 0001461C 00C600C6C6C6C6C6C6- <1>    db  000h, 0c6h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
630 00014625 C6C67C00000000      <1>
631 0001462C 0018187EC3C0C0C0C3- <1>    db  000h, 018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h
631 00014635 7E181800000000      <1>
632 0001463C 00386C6460F0606060- <1>    db  000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h, 060h, 060h, 0e6h, 0fch, 000h, 000h, 000h, 000h
632 00014645 60E6FC00000000      <1>
633 0001464C 0000C3663C18FF18FF- <1>    db  000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 000h, 000h, 000h, 000h
633 00014655 18181800000000      <1>
634 0001465C 00FC66667C62666F66- <1>    db  000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 066h, 0f3h, 000h, 000h, 000h, 000h
634 00014665 6666F300000000      <1>
635 0001466C 000E1B1818187E1818- <1>    db  000h, 00eh, 01bh, 018h, 018h, 018h, 07eh, 018h, 018h, 018h, 018h, 018h, 0d8h, 070h, 000h, 000h
635 00014675 181818D8700000      <1>
636 0001467C 0018306000780C7CCC- <1>    db  000h, 018h, 030h, 060h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
636 00014685 CCCC7600000000      <1>
637 0001468C 000C18300038181818- <1>    db  000h, 00ch, 018h, 030h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
637 00014695 18183C00000000      <1>
638 0001469C 00183060007CC6C6C6- <1>    db  000h, 018h, 030h, 060h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
638 000146A5 C6C67C00000000      <1>
639 000146AC 0018306000CCCCCCCC- <1>    db  000h, 018h, 030h, 060h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
639 000146B5 CCCC7600000000      <1>
640 000146BC 000076DC00DC666666- <1>    db  000h, 000h, 076h, 0dch, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 066h, 000h, 000h, 000h, 000h
640 000146C5 66666600000000      <1>
641 000146CC 76DC00C6E6F6FEDECE- <1>    db  076h, 0dch, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
641 000146D5 C6C6C600000000      <1>
642 000146DC 003C6C6C3E007E0000- <1>    db  000h, 03ch, 06ch, 06ch, 03eh, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
642 000146E5 00000000000000      <1>
643 000146EC 00386C6C38007C0000- <1>    db  000h, 038h, 06ch, 06ch, 038h, 000h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
643 000146F5 00000000000000      <1>
644 000146FC 0000303000303060C0- <1>    db  000h, 000h, 030h, 030h, 000h, 030h, 030h, 060h, 0c0h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
644 00014705 C6C67C00000000      <1>
645 0001470C 000000000000FEC0C0- <1>    db  000h, 000h, 000h, 000h, 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h, 000h
645 00014715 C0C00000000000      <1>
646 0001471C 000000000000FE0606- <1>    db  000h, 000h, 000h, 000h, 000h, 000h, 0feh, 006h, 006h, 006h, 006h, 000h, 000h, 000h, 000h, 000h
646 00014725 06060000000000      <1>
647 0001472C 00C0C0C2C6CC183060- <1>    db  000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 060h, 0ceh, 09bh, 006h, 00ch, 01fh, 000h, 000h
647 00014735 CE9B060C1F0000      <1>
648 0001473C 00C0C0C2C6CC183066- <1>    db  000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 066h, 0ceh, 096h, 03eh, 006h, 006h, 000h, 000h
648 00014745 CE963E06060000      <1>
649 0001474C 000018180018183C- <1>    db  000h, 000h, 018h, 018h, 000h, 018h, 018h, 018h, 03ch, 03ch, 03ch, 018h, 000h, 000h, 000h, 000h
649 00014755 3C3C1800000000      <1>
650 0001475C 0000000000366CD86C- <1>    db  000h, 000h, 000h, 000h, 000h, 036h, 06ch, 0d8h, 06ch, 036h, 000h, 000h, 000h, 000h, 000h, 000h
650 00014765 36000000000000      <1>
651 0001476C 0000000000D86C366C- <1>    db  000h, 000h, 000h, 000h, 000h, 0d8h, 06ch, 036h, 06ch, 0d8h, 000h, 000h, 000h, 000h, 000h, 000h
651 00014775 D8000000000000      <1>
652 0001477C 114411441144114411- <1>    db  011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h
652 00014785 44114411441144      <1>
653 0001478C 55AA55AA55AA55AA55- <1>    db  055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah
653 00014795 AA55AA55AA55AA      <1>
654 0001479C DD77DD77DD77DD77DD- <1>    db  0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h
654 000147A5 77DD77DD77DD77      <1>
655 000147AC 181818181818181818- <1>    db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
655 000147B5 18181818181818      <1>
```

```
656 000147BC 18181818181818F818- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
656 000147C5 18181818181818      <1>
657 000147CC 1818181818F818F818- <1>     db  018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
657 000147D5 18181818181818      <1>
658 000147DC 36363636363636F636- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
658 000147E5 36363636363636      <1>
659 000147EC 00000000000000FE36- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
659 000147F5 36363636363636      <1>
660 000147FC 0000000000F818F818- <1>     db  000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
660 00014805 18181818181818      <1>
661 0001480C 3636363636F606F636- <1>     db  036h, 036h, 036h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
661 00014815 36363636363636      <1>
662 0001481C 363636363636363636- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
662 00014825 36363636363636      <1>
663 0001482C 0000000000FE06F636- <1>     db  000h, 000h, 000h, 000h, 000h, 0feh, 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
663 00014835 36363636363636      <1>
664 0001483C 3636363636F606FE00- <1>     db  036h, 036h, 036h, 036h, 036h, 0f6h, 006h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
664 00014845 00000000000000      <1>
665 0001484C 36363636363636FE00- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
665 00014855 00000000000000      <1>
666 0001485C 1818181818F818F800- <1>     db  018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
666 00014865 00000000000000      <1>
667 0001486C 00000000000000F818- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
667 00014875 18181818181818      <1>
668 0001487C 181818181818181F00- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
668 00014885 00000000000000      <1>
669 0001488C 18181818181818FF00- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
669 00014895 00000000000000      <1>
670 0001489C 00000000000000FF18- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
670 000148A5 18181818181818      <1>
671 000148AC 181818181818181F18- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
671 000148B5 18181818181818      <1>
672 000148BC 00000000000000FF00- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
672 000148C5 00000000000000      <1>
673 000148CC 18181818181818FF18- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
673 000148D5 18181818181818      <1>
674 000148DC 18181818181F181F18- <1>     db  018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
674 000148E5 18181818181818      <1>
675 000148EC 363636363636363736- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
675 000148F5 36363636363636      <1>
676 000148FC 363636363637303F00- <1>     db  036h, 036h, 036h, 036h, 036h, 037h, 030h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
676 00014905 00000000000000      <1>
677 0001490C 00000000003F303736- <1>     db  000h, 000h, 000h, 000h, 000h, 03fh, 030h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
677 00014915 36363636363636      <1>
678 0001491C 3636363636F700FF00- <1>     db  036h, 036h, 036h, 036h, 036h, 0f7h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
678 00014925 00000000000000      <1>
679 0001492C 0000000000FF00F736- <1>     db  000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0f7h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
679 00014935 36363636363636      <1>
680 0001493C 363636363637303736- <1>     db  036h, 036h, 036h, 036h, 036h, 037h, 030h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
680 00014945 36363636363636      <1>
681 0001494C 0000000000FF00FF00- <1>     db  000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
681 00014955 00000000000000      <1>
682 0001495C 3636363636F700F736- <1>     db  036h, 036h, 036h, 036h, 036h, 0f7h, 000h, 0f7h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
682 00014965 36363636363636      <1>
683 0001496C 1818181818FF00FF00- <1>     db  018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
683 00014975 00000000000000      <1>
684 0001497C 36363636363636FF00- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
684 00014985 00000000000000      <1>
685 0001498C 0000000000FF00FF18- <1>     db  000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
685 00014995 18181818181818      <1>
686 0001499C 00000000000000FF36- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
686 000149A5 36363636363636      <1>
687 000149AC 363636363636363F00- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
687 000149B5 00000000000000      <1>
688 000149BC 18181818181F181F00- <1>     db  018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
688 000149C5 00000000000000      <1>
689 000149CC 00000000001F181F18- <1>     db  000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
689 000149D5 18181818181818      <1>
690 000149DC 000000000000003F36- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 03fh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
690 000149E5 36363636363636      <1>
691 000149EC 36363636363636FF36- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
691 000149F5 36363636363636      <1>
692 000149FC 1818181818FF18FF18- <1>     db  018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
692 00014A05 18181818181818      <1>
693 00014A0C 18181818181818F800- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
693 00014A15 00000000000000      <1>
694 00014A1C 000000000000001F18- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
694 00014A25 18181818181818      <1>
695 00014A2C FFFFFFFFFFFFFFFFFF- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
695 00014A35 FFFFFFFFFFFFFF      <1>
696 00014A3C 00000000000000FFFF- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
696 00014A45 FFFFFFFFFFFFFF      <1>
697 00014A4C F0F0F0F0F0F0F0F0F0- <1>     db  0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h
697 00014A55 F0F0F0F0F0F0F0      <1>
698 00014A5C 0F0F0F0F0F0F0F0F0F- <1>     db  00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh
698 00014A65 0F0F0F0F0F0F0F      <1>
699 00014A6C FFFFFFFFFFFFFF0000- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
699 00014A75 00000000000000      <1>
700 00014A7C 000000000076DCD8D8- <1>     db  000h, 000h, 000h, 000h, 000h, 076h, 0dch, 0d8h, 0d8h, 0d8h, 0dch, 076h, 000h, 000h, 000h, 000h
700 00014A85 D8DC7600000000      <1>
701 00014A8C 000078CCCCCCD8CCC6- <1>     db  000h, 000h, 078h, 0cch, 0cch, 0cch, 0d8h, 0cch, 0c6h, 0c6h, 0c6h, 0cch, 000h, 000h, 000h, 000h
701 00014A95 C6C6CC00000000      <1>
702 00014A9C 0000FEC6C6C0C0C0C0- <1>     db  000h, 000h, 0feh, 0c6h, 0c6h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h
702 00014AA5 C0C0C000000000      <1>
703 00014AAC 00000000FE6C6C6C6C- <1>     db  000h, 000h, 000h, 000h, 0feh, 06ch, 06ch, 06ch, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h
703 00014AB5 6C6C6C00000000      <1>
704 00014ABC 000000FEC660301830- <1>     db  000h, 000h, 000h, 0feh, 0c6h, 060h, 030h, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h
704 00014AC5 60C6FE00000000      <1>
705 00014ACC 00000000007ED8D8D8- <1>     db  000h, 000h, 000h, 000h, 000h, 07eh, 0d8h, 0d8h, 0d8h, 0d8h, 0d8h, 070h, 000h, 000h, 000h, 000h
705 00014AD5 D8D87000000000      <1>
706 00014ADC 0000000066666666- <1>     db  000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h, 066h, 07ch, 060h, 060h, 0c0h, 000h, 000h, 000h
706 00014AE5 7C6060C0000000      <1>
707 00014AEC 0000000076DC181818- <1>     db  000h, 000h, 000h, 000h, 076h, 0dch, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h
707 00014AF5 18181800000000      <1>
708 00014AFC 0000007E183C666666- <1>     db  000h, 000h, 000h, 07eh, 018h, 03ch, 066h, 066h, 066h, 03ch, 018h, 07eh, 000h, 000h, 000h, 000h
708 00014B05 3C187E00000000      <1>
709 00014B0C 000000386CC6C6FEC6- <1>     db  000h, 000h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h
709 00014B15 C66C3800000000      <1>
```

```
710 00014B1C 0000386CC6C6C66C6C- <1>     db  000h, 000h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 06ch, 06ch, 06ch, 06ch, 0eeh, 000h, 000h, 000h, 000h
710 00014B25 6C6CEE00000000      <1>
711 00014B2C 00001E30180C3E6666- <1>     db  000h, 000h, 01eh, 030h, 018h, 00ch, 03eh, 066h, 066h, 066h, 066h, 03ch, 000h, 000h, 000h, 000h
711 00014B35 66663C00000000      <1>
712 00014B3C 00000000007EDBDBDB- <1>     db  000h, 000h, 000h, 000h, 000h, 07eh, 0dbh, 0dbh, 0dbh, 07eh, 000h, 000h, 000h, 000h, 000h, 000h
712 00014B45 7E000000000000      <1>
713 00014B4C 00000003067EDBDBF3- <1>     db  000h, 000h, 000h, 003h, 006h, 07eh, 0dbh, 0dbh, 0f3h, 07eh, 060h, 0c0h, 000h, 000h, 000h, 000h
713 00014B55 7E60C000000000      <1>
714 00014B5C 00001C3060607C6060- <1>     db  000h, 000h, 01ch, 030h, 060h, 060h, 07ch, 060h, 060h, 060h, 030h, 01ch, 000h, 000h, 000h, 000h
714 00014B65 60301C00000000      <1>
715 00014B6C 0000007CC6C6C6C6C6- <1>     db  000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
715 00014B75 C6C6C600000000      <1>
716 00014B7C 00000000FE0000FE00- <1>     db  000h, 000h, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 000h, 000h, 000h
716 00014B85 00FE0000000000      <1>
717 00014B8C 0000000018187E1818- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 07eh, 018h, 018h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h
717 00014B95 0000FF00000000      <1>
718 00014B9C 00000030180C060C18- <1>     db  000h, 000h, 000h, 030h, 018h, 00ch, 006h, 00ch, 018h, 030h, 000h, 07eh, 000h, 000h, 000h, 000h
718 00014BA5 30007E00000000      <1>
719 00014BAC 0000000C1830603018- <1>     db  000h, 000h, 000h, 00ch, 018h, 030h, 060h, 030h, 018h, 00ch, 000h, 07eh, 000h, 000h, 000h, 000h
719 00014BB5 0C007E00000000      <1>
720 00014BBC 00000E1B1B18181818- <1>     db  000h, 000h, 00eh, 01bh, 01bh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
720 00014BC5 18181818181818      <1>
721 00014BCC 1818181818181818D8- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0d8h, 0d8h, 0d8h, 070h, 000h, 000h, 000h, 000h
721 00014BD5 D8D87000000000      <1>
722 00014BDC 0000000018180 07E00- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 000h, 07eh, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h
722 00014BE5 18180000000000      <1>
723 00014BEC 0000000000076DC0076- <1>     db  000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h
723 00014BF5 DC0000000000000     <1>
724 00014BFC 00386C6C3800000000- <1>     db  000h, 038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
724 00014C05 00000000000000      <1>
725 00014C0C 0000000000000001818- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
725 00014C15 00000000000000      <1>
726 00014C1C 0000000000000000018- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
726 00014C25 000000000000000     <1>
727 00014C2C 000F0C0C0C0C0CEC6C- <1>     db  000h, 00fh, 00ch, 00ch, 00ch, 00ch, 00ch, 0ech, 06ch, 06ch, 03ch, 01ch, 000h, 000h, 000h, 000h
727 00014C35 6C3C1C00000000      <1>
728 00014C3C 00D86C6C6C6C6C0000- <1>     db  000h, 0d8h, 06ch, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
728 00014C45 00000000000000      <1>
729 00014C4C 0070D83060C8F80000- <1>     db  000h, 070h, 0d8h, 030h, 060h, 0c8h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
729 00014C55 00000000000000      <1>
730 00014C5C 000000007C7C7C7C7C- <1>     db  000h, 000h, 000h, 000h, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 000h, 000h, 000h, 000h, 000h
730 00014C65 7C7C0000000000      <1>
731 00014C6C 0000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
731 00014C75 00000000000000      <1>
732                              <1> vgafont14alt:
733 00014C7C 1D000000002466FF66- <1>     db  01dh, 000h, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h, 000h, 000h, 000h, 022h
733 00014C85 24000000000022      <1>
734 00014C8C 006363632200000000- <1>     db  000h, 063h, 063h, 063h, 022h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 02bh, 000h
734 00014C95 00000000002B00      <1>
735 00014C9C 0000181818FF181818- <1>     db  000h, 000h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 02dh, 000h, 000h
735 00014CA5 000000002D0000      <1>
736 00014CAC 00000000FF00000000- <1>     db  000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 04dh, 000h, 000h, 0c3h
736 00014CB5 0000004D0000C3      <1>
737 00014CBC E7FFDBC3C3C3C3C300- <1>     db  0e7h, 0ffh, 0dbh, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 000h, 000h, 000h, 054h, 000h, 000h, 0ffh, 0dbh
737 00014CC5 0000540000FFDB      <1>
738 00014CCC 9918181818183C0000- <1>     db  099h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 056h, 000h, 000h, 0c3h, 0c3h, 0c3h
738 00014CD5 00560000C3C3C3      <1>
739 00014CDC C3C3C3663C18000000- <1>     db  0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 057h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h
739 00014CE5 570000C3C3C3C3      <1>
740 00014CEC DBDBFF666600000058- <1>     db  0dbh, 0dbh, 0ffh, 066h, 066h, 000h, 000h, 000h, 058h, 000h, 000h, 0c3h, 0c3h, 066h, 03ch, 018h
740 00014CF5 0000C3C3663C18      <1>
741 00014CFC 3C66C3C30000005900- <1>     db  03ch, 066h, 0c3h, 0c3h, 000h, 000h, 000h, 059h, 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h
741 00014D05 00C3C3C3663C18      <1>
742 00014D0C 18183C0000005A0000- <1>     db  018h, 018h, 03ch, 000h, 000h, 000h, 05ah, 000h, 000h, 0ffh, 0c3h, 086h, 00ch, 018h, 030h, 061h
742 00014D15 FFC3860C183061      <1>
743 00014D1C C3FF0000006D000000- <1>     db  0c3h, 0ffh, 000h, 000h, 000h, 06dh, 000h, 000h, 000h, 000h, 000h, 0e6h, 0ffh, 0dbh, 0dbh, 0dbh
743 00014D25 0000E6FFDBDBDB      <1>
744 00014D2C DB0000007600000000- <1>     db  0dbh, 000h, 000h, 000h, 076h, 000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h
744 00014D35 00C3C3C3663C18      <1>
745 00014D3C 000000770000000000- <1>     db  000h, 000h, 000h, 077h, 000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h
745 00014D45 C3C3DBDBFF6600      <1>
746 00014D4C 000091000000006E3B- <1>     db  000h, 000h, 091h, 000h, 000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h
746 00014D55 1B7ED8DC770000      <1>
747 00014D5C 009B0018187EC3C0C0- <1>     db  000h, 09bh, 000h, 018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h
747 00014D65 C37E1818000000      <1>
748 00014D6C 9D0000C3663C18FF18- <1>     db  09dh, 000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 000h, 000h, 000h, 09eh
748 00014D75 FF18180000009E      <1>
749 00014D7C 00FC66667C62666F66- <1>     db  000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 0f3h, 000h, 000h, 000h, 0f1h, 000h
749 00014D85 66F3000000F100      <1>
750 00014D8C 00181818FF18181800- <1>     db  000h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 000h, 0ffh, 000h, 000h, 000h, 0f6h, 000h, 000h
750 00014D95 FF000000F60000      <1>
751 00014D9C 18180000FF00001818- <1>     db  018h, 018h, 000h, 000h, 0ffh, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h
751 00014DA5 00000000            <1>
752                              <1> vgafont16alt:
753 00014DA9 1D00000000002466FF- <1>     db  01dh, 000h, 000h, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h, 000h, 000h, 000h
753 00014DB2 66240000000000      <1>
754 00014DB9 003000003C66C3C3DB- <1>     db  000h, 030h, 000h, 000h, 03ch, 066h, 0c3h, 0c3h, 0dbh, 0dbh, 0c3h, 0c3h, 066h, 03ch, 000h, 000h
754 00014DC2 DBC3C3663C0000      <1>
755 00014DC9 00004D0000C3E7FFFF- <1>     db  000h, 000h, 04dh, 000h, 000h, 0c3h, 0e7h, 0ffh, 0ffh, 0dbh, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 000h
755 00014DD2 DBC3C3C3C3C300      <1>
756 00014DD9 0000000540000FFDB99- <1>     db  000h, 000h, 000h, 054h, 000h, 000h, 0ffh, 0dbh, 099h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch
756 00014DE2 1818181818183C      <1>
757 00014DE9 000000005600000C3C3- <1>     db  000h, 000h, 000h, 000h, 056h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch
757 00014DF2 C3C3C3C3C3663C      <1>
758 00014DF9 1800000000570000C3- <1>     db  018h, 000h, 000h, 000h, 000h, 057h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh
758 00014E02 C3C3C3C3DBDBFF      <1>
759 00014E09 666600000000580000- <1>     db  066h, 066h, 000h, 000h, 000h, 000h, 058h, 000h, 000h, 0c3h, 0c3h, 066h, 03ch, 018h, 018h, 03ch
759 00014E12 C3C3663C18183C      <1>
760 00014E19 66C3C3000000005900- <1>     db  066h, 0c3h, 0c3h, 000h, 000h, 000h, 000h, 059h, 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h
760 00014E22 00C3C3C3663C18      <1>
761 00014E29 1818183C000000005A- <1>     db  018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 05ah, 000h, 000h, 0ffh, 0c3h, 086h, 00ch, 018h
761 00014E32 0000FFC3860C18      <1>
762 00014E39 3060C1C3FF00000000- <1>     db  030h, 060h, 0c1h, 0c3h, 0ffh, 000h, 000h, 000h, 000h, 06dh, 000h, 000h, 000h, 000h, 000h, 0e6h
762 00014E42 6D0000000000E6      <1>
763 00014E49 FFDBDBDBDBDB000000- <1>     db  0ffh, 0dbh, 0dbh, 0dbh, 0dbh, 0dbh, 000h, 000h, 000h, 000h, 076h, 000h, 000h, 000h, 000h, 000h
763 00014E52 00760000000000      <1>
764 00014E59 C3C3C3C3663C180000- <1>     db  0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 000h, 077h, 000h, 000h, 000h, 000h
764 00014E62 00007700000000      <1>
```

```
 765 00014E69 00C3C3C3DBDBFF6600- <1>     db  000h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h, 000h, 000h, 000h, 078h, 000h, 000h, 000h
 765 00014E72 00000078000000      <1>
 766 00014E79 0000C3663C183C66C3- <1>     db  000h, 000h, 0c3h, 066h, 03ch, 018h, 03ch, 066h, 0c3h, 000h, 000h, 000h, 000h, 091h, 000h, 000h
 766 00014E82 000000000910000     <1>
 767 00014E89 0000006E3B1B7ED8DC- <1>     db  000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h, 000h, 000h, 09bh, 000h
 767 00014E92 77000000009B00      <1>
 768 00014E99 18187EC3C0C0C0C37E- <1>     db  018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h, 09dh
 768 00014EA2 1818000000009D      <1>
 769 00014EA9 0000C3663C18FF18FF- <1>     db  000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 000h, 000h, 000h, 000h
 769 00014EB2 18181800000000      <1>
 770 00014EB9 9E00FC66667C62666F- <1>     db  09eh, 000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 066h, 0f3h, 000h, 000h, 000h
 770 00014EC2 666666F3000000      <1>
 771 00014EC9 00AB00C0C0C2C6CC18- <1>     db  000h, 0abh, 000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 060h, 0ceh, 09bh, 006h, 00ch, 01fh
 771 00014ED2 3060CE9B060C1F      <1>
 772 00014ED9 0000AC00C0C0C2C6CC- <1>     db  000h, 000h, 0ach, 000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 066h, 0ceh, 096h, 03eh, 006h
 772 00014EE2 183066CE963E06      <1>
 773 00014EE9 06000000            <1>     db  006h, 000h, 000h, 000h
2645
2646 00014EED 90                          align 2
2647
2648                                      ; EPOCH Variables
2649                                      ; 13/04/2015 - Retro UNIX 386 v1 Beginning
2650                                      ; 09/04/2013 epoch variables
2651                                      ; Retro UNIX 8086 v1 Prototype: UNIXCOPY.ASM, 10/03/2013
2652                                      ;
2653 00014EEE B207                       year:       dw 1970
2654 00014EF0 0100                       month:      dw 1
2655 00014EF2 0100                       day:  dw 1
2656 00014EF4 0000                       hour:       dw 0
2657 00014EF6 0000                       minute: dw 0
2658 00014EF8 0000                       second: dw 0
2659
2660                                      DMonth:
2661 00014EFA 0000                               dw 0
2662 00014EFC 1F00                               dw 31
2663 00014EFE 3B00                               dw 59
2664 00014F00 5A00                               dw 90
2665 00014F02 7800                               dw 120
2666 00014F04 9700                               dw 151
2667 00014F06 B500                               dw 181
2668 00014F08 D400                               dw 212
2669 00014F0A F300                               dw 243
2670 00014F0C 1101                               dw 273
2671 00014F0E 3001                               dw 304
2672 00014F10 4E01                               dw 334
2673
2674                                       ; 20/02/2017
2675                                       KERNELFSIZE  equ $  ; 04/07/2016
2676
2677                                     bss_start:
2678
2679                                     ABSOLUTE bss_start
2680
2681 00014F12 <res 00000006>             alignb 8 ; 25/12/2016
2682
2683                                       ; 15/04/2016
2684                                       ; TRDOS 386 (TRDOS v2.0)
2685                                       ;      80 interrupts
2686                                       ; 11/03/2015
2687                                       ; Interrupt Descriptor Table (20/08/2014)
2688                                     idt:
2689                                       ;resb  64*8 ; INT 0 to INT 3Fh
2690                                       ; 15/04/2016
2691 00014F18 <res 00000280>              resb   80*8 ; INT 0 to INT 4Fh
2692
2693                                     idt_end:
2694
2695                                     ;alignb 4
2696
2697                                     task_state_segment:
2698                                       ; 24/03/2015
2699 00015198 <res 00000002>             tss.link:   resw 1
2700 0001519A <res 00000002>                  resw 1
2701                                     ; tss offset 4
2702 0001519C <res 00000004>             tss.esp0:   resd 1
2703 000151A0 <res 00000002>             tss.ss0:    resw 1
2704 000151A2 <res 00000002>                  resw 1
2705 000151A4 <res 00000004>             tss.esp1:   resd 1
2706 000151A8 <res 00000002>             tss.ss1:    resw 1
2707 000151AA <res 00000002>                  resw 1
2708 000151AC <res 00000004>             tss.esp2:   resd 1
2709 000151B0 <res 00000002>             tss.ss2:    resw 1
2710 000151B2 <res 00000002>                  resw 1
2711                                     ; tss offset 28
2712 000151B4 <res 00000004>             tss.CR3:    resd 1
2713 000151B8 <res 00000004>             tss.eip:    resd 1
2714 000151BC <res 00000004>             tss.eflags: resd 1
2715                                     ; tss offset 40
2716 000151C0 <res 00000004>             tss.eax:    resd 1
2717 000151C4 <res 00000004>             tss.ecx:    resd 1
2718 000151C8 <res 00000004>             tss.edx:    resd 1
2719 000151CC <res 00000004>             tss.ebx:    resd 1
2720 000151D0 <res 00000004>             tss.esp:    resd 1
2721 000151D4 <res 00000004>             tss.ebp:    resd 1
2722 000151D8 <res 00000004>             tss.esi:    resd 1
2723 000151DC <res 00000004>             tss.edi:    resd 1
2724                                     ; tss offset 72
2725 000151E0 <res 00000002>             tss.ES:     resw 1
2726 000151E2 <res 00000002>                  resw 1
2727 000151E4 <res 00000002>             tss.CS:      resw 1
2728 000151E6 <res 00000002>                  resw 1
2729 000151E8 <res 00000002>             tss.SS:      resw 1
2730 000151EA <res 00000002>                  resw 1
2731 000151EC <res 00000002>             tss.DS:      resw 1
2732 000151EE <res 00000002>                  resw 1
2733 000151F0 <res 00000002>             tss.FS:      resw 1
2734 000151F2 <res 00000002>                  resw 1
```

```
2735 000151F4 <res 00000002>          tss.GS:      resw 1
2736 000151F6 <res 00000002>               resw 1
2737 000151F8 <res 00000002>          tss.LDTR:   resw 1
2738 000151FA <res 00000002>               resw 1

2739                                  ; tss offset 100
2740 000151FC <res 00000002>               resw 1
2741 000151FE <res 00000002>          tss.IOPB:   resw 1
2742                                  ; tss offset 104
2743                                  tss_end:
2744
2745 00015200 <res 00000004>          k_page_dir:  resd 1 ; Kernel's (System) Page Directory address
2746                                               ; (Physical address = Virtual address)
2747 00015204 <res 00000004>          memory_size: resd 1 ; memory size in pages
2748 00015208 <res 00000004>          free_pages:  resd 1 ; number of free pages
2749 0001520C <res 00000004>          next_page:   resd 1 ; offset value in M.A.T. for
2750                                               ; first free page search
2751 00015210 <res 00000004>          last_page:   resd 1 ; offset value in M.A.T. which
2752                                               ; next free page search will be
2753                                               ; stopped after it. (end of M.A.T.)
2754 00015214 <res 00000004>          first_page:  resd 1 ; offset value in M.A.T. which
2755                                               ; first free page search
2756                                               ; will be started on it. (for user)
2757 00015218 <res 00000004>          mat_size:    resd 1 ; Memory Allocation Table size in pages
2758
2759                                  ; 02/09/2014 (Retro UNIX 386 v1)
2760                                  ; 04/12/2013 (Retro UNIX 8086 v1)
2761 0001521C <res 00000002>          CRT_START:   resw 1     ; starting address in regen buffer
2762                                                   ; NOTE: active page only
2763 0001521E <res 00000010>          CURSOR_POSN: resw 8 ; cursor positions for video pages
2764                                  ACTIVE_PAGE:
2765 0001522E <res 00000001>          ptty:         resb 1 ; current tty
2766                                  ; 01/07/2015 - 29/01/2016
2767 0001522F <res 00000001>          ccolor:       resb 1 ; current color attribute
2768                                  ; 26/10/2015
2769                                  ; 07/09/2014
2770 00015230 <res 00000014>          ttychr:      resw ntty+2 ; Character buffer (multiscreen)
2771
2772                                  ; 18/05/2015 (03/06/2013 - Retro UNIX 8086 v1 feature only!)
2773 00015244 <res 00000004>          p_time:      resd 1     ; present time (for systime & sysmdate)
2774
2775                                  ; 18/05/2015 (16/08/2013 - Retro UNIX 8086 v1 feature only !)
2776                                  ; (open mode locks for pseudo TTYs)
2777                                  ; [ major tty locks (return error in any conflicts) ]
2778 00015248 <res 00000014>          ttyl:        resw ntty+2 ; opening locks for TTYs.
2779
2780                                  ; 15/04/2015 (Retro UNIX 386 v1)
2781                                  ; 22/09/2013 (Retro UNIX 8086 v1)
2782 0001525C <res 0000000A>          wlist:       resb ntty+2 ; wait channel list (0 to 9 for TTYs)
2783                                  ; 15/04/2015 (Retro UNIX 386 v1)
2784                                  ;; 12/07/2014 -> sp_init set comm. parameters as 0E3h
2785                                  ;; 0 means serial port is not available
2786                                  ;;comprm: ; 25/06/2014
2787 00015266 <res 00000001>          com1p:       resb 1  ;;0E3h
2788 00015267 <res 00000001>          com2p:       resb 1  ;;0E3h
2789
2790                                  ; 17/11/2015
2791                                  ; request for response (from the terminal)
2792 00015268 <res 00000002>          req_resp:    resw 1
2793                                  ; 07/11/2015
2794 0001526A <res 00000001>          ccomport:    resb 1 ; current COM (serial) port
2795                                               ; (0= COM1, 1= COM2)
2796                                  ; 09/11/2015
2797 0001526B <res 00000001>          comqr:        resb 1 ; 'query or response' sign (u9.s, 'sndc')
2798                                  ; 07/11/2015
2799 0001526C <res 00000002>          rchar:        resw 1 ; last received char for COM 1 and COM 2
2800 0001526E <res 00000002>          schar:        resw 1 ; last sent char for COM 1 and COM 2
2801
2802                                  ; 22/08/2014 (RTC)
2803                                  ; (Packed BCD)
2804 00015270 <res 00000001>          time_seconds: resb 1
2805 00015271 <res 00000001>          time_minutes: resb 1
2806 00015272 <res 00000001>          time_hours:   resb 1
2807 00015273 <res 00000001>          date_wday:    resb 1
2808 00015274 <res 00000001>          date_day:     resb 1
2809 00015275 <res 00000001>          date_month:   resb 1
2810 00015276 <res 00000001>          date_year:    resb 1
2811 00015277 <res 00000001>          date_century: resb 1
2812
2813                                  ; 24/01/2016
2814 00015278 <res 00000004>          RTC_LH:        resd 1
2815 0001527C <res 00000001>          RTC_WAIT_FLAG: resb 1
2816 0001527D <res 00000001>          USER_FLAG:     resb 1
2817                                  ; 19/05/2016
2818                                  ;RTC_second:
2819 0001527E <res 00000001>          RTC_2Hz:      resb 1 ;  from 2Hz interrupt to 1Hz timer event function
2820
2821                                  %include 'diskbss.s'   ; UNINITIALIZED DISK (BIOS) DATA
   1                                 <1> ; ****************************************************************************
   2                                 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - diskbss.s
   3                                 <1> ; ----------------------------------------------------------------------------
   4                                 <1> ; Last Update: 24/01/2016
   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> ; diskbss.inc (10/07/2015)
  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 - DISKBSS.INC
  20                                 <1> ; Last Modification: 10/07/2015
```

```
                            <1> ; (Unnitialized Disk Parameters Data section for 'DISKIO.INC')
                            <1>
0001527F <res 00000001>     <1> alignb 2
                            <1>
                            <1> ;----------------------------------------
                            <1> ; TIMER DATA AREA            :
                            <1> ;----------------------------------------
                            <1>
                            <1> TIMER_LH:       ; 16/02/205
00015280 <res 00000002>     <1> TIMER_LOW:      resw   1              ; LOW WORD OF TIMER COUNT
00015282 <res 00000002>     <1> TIMER_HIGH:     resw   1              ; HIGH WORD OF TIMER COUNT
00015284 <res 00000001>     <1> TIMER_OFL:      resb   1              ; TIMER HAS ROLLED OVER SINCE LAST READ
                            <1>
                            <1> ;----------------------------------------
                            <1> ; DISKETTE DATA AREAS        :
                            <1> ;----------------------------------------
                            <1>
00015285 <res 00000001>     <1> SEEK_STATUS:    resb   1
00015286 <res 00000001>     <1> MOTOR_STATUS:   resb   1
00015287 <res 00000001>     <1> MOTOR_COUNT:    resb   1
00015288 <res 00000001>     <1> DSKETTE_STATUS: resb   1
00015289 <res 00000007>     <1> NEC_STATUS:     resb   7
                            <1>
                            <1> ;----------------------------------------
                            <1> ; ADDITIONAL MEDIA DATA             :
                            <1> ;----------------------------------------
                            <1>
00015290 <res 00000001>     <1> LASTRATE:       resb   1
00015291 <res 00000001>     <1> HF_STATUS:      resb   1
00015292 <res 00000001>     <1> HF_ERROR:       resb   1
00015293 <res 00000001>     <1> HF_INT_FLAG:    resb   1
00015294 <res 00000001>     <1> HF_CNTRL:       resb   1
00015295 <res 00000004>     <1> DSK_STATE:      resb   4
00015299 <res 00000002>     <1> DSK_TRK: resb   2
                            <1>
0001529B <res 00000001>     <1> DISK_STATUS1:   resb   1              ; FIXED DISK STATUS
0001529C <res 00000001>     <1> HF_NUM:         resb   1              ; COUNT OF FIXED DISK DRIVES
0001529D <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>
0001529E <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
000152A0 <res 00000004>     <1> HDPM_TBL_VEC:   resd   1              ; Primary master disk param. tbl. pointer
000152A4 <res 00000004>     <1> HDPS_TBL_VEC:   resd   1              ; Primary slave disk param. tbl. pointer
000152A8 <res 00000004>     <1> HDSM_TBL_VEC:   resd   1              ; Secondary master disk param. tbl. pointer
000152AC <res 00000004>     <1> HDSS_TBL_VEC:   resd   1              ; Secondary slave disk param. tbl. pointer
                            <1>
                            <1> ; 03/01/2015
000152B0 <res 00000001>     <1> LBAMode:        resb   1
                            <1>
                            <1> ; ****************************************************************************

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

                                ;alignb 2

                                ; 10/01/2016
                                %include 'trdoskx.s'   ; UNINITIALIZED KERNEL (Logical Drive & FS) DATA
                            <1> ; ****************************************************************************
                            <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - UNINITIALIZED DATA : trdoskx.s
                            <1> ; ----------------------------------------------------------------------------
                            <1> ; Last Update: 28/08/2017
                            <1> ; ----------------------------------------------------------------------------
                            <1> ; Beginning: 04/01/2016
                            <1> ; ----------------------------------------------------------------------------
                            <1> ; Assembler: NASM version 2.11 (trdos386.s)
                            <1> ; ----------------------------------------------------------------------------
                            <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
                            <1> ; TRDOS2.ASM (09/11/2011)
                            <1> ; ****************************************************************************
                            <1> ; DRV_INIT.ASM [26/09/2009] Last Update: 07/08/2011
                            <1> ; MAINPROG.ASM [17/01/2004] Last Update: 09/11/2011
                            <1> ; DIR.ASM      [17/01/2004] Last Update: 09/10/2011
                            <1> ; CMD_INTR.ASM [29/01/2005] Last update: 09/11/2011
                            <1> ; DRV_FAT.ASM  [07/07/2009] Last update: 21/08/2011
                            <1>
000152B1 <res 00000003>     <1> alignb 4
                            <1>
                            <1> ; MAINPROG.ASM
000152B4 <res 00000004>     <1> MainProgCfg_FileSize:   resd 1 ; 14/04/2016
000152B8 <res 00000004>     <1> MainProgCfg_LineOffset: resd 1 ; 14/04/2016
                            <1>
000152BC <res 00000004>     <1> Current_VolSerial: resd 1
                            <1>
000152C0 <res 00000004>     <1> Current_Dir_FCluster: resd 1
                            <1>
000152C4 <res 00000001>     <1> Current_Dir_Level: resb 1
000152C5 <res 00000001>     <1> Current_FATType: resb 1
000152C6 <res 00000001>     <1> Current_Drv: resb 1
000152C7 <res 00000001>     <1> Current_Dir_Drv:   resb 1 ; '?'
000152C8 <res 00000001>     <1>                    resb 1 ; ':'
000152C9 <res 00000001>     <1> Current_Dir_Root:  resb 1 ; '/'
000152CA <res 0000005A>     <1> Current_Directory: resb 90
00015324 <res 00000001>     <1> End_Of_Current_Dir_Str: resb 1
00015325 <res 00000001>     <1> Current_Dir_StrLen: resb 1
                            <1>
00015326 <res 00000001>     <1> CursorColumn:   resb 1
00015327 <res 00000001>     <1> CmdArgStart:    resb 1
                            <1>
```

```
                            <1> ; 03/02/2016
00015328 <res 0000004E>     <1> Remark:         resb 78
                            <1>
00015376 <res 00000050>     <1> CommandBuffer:  resb 80
                            <1>
000153C6 <res 00000100>     <1> TextBuffer:     resb 256
                            <1>
                            <1> MasterBootBuff:
000154C6 <res 000001BE>     <1> MasterBootCode: resb 1BEh
00015684 <res 00000040>     <1> PartitionTable: resb 64
000156C4 <res 00000002>     <1> MBIDCode: resw 1
                            <1>
                            <1> PTable_Buffer:
000156C6 <res 00000040>     <1> PTable_hd0: resb 64
00015706 <res 00000040>     <1> PTable_hd1: resb 64
00015746 <res 00000040>     <1> PTable_hd2: resb 64
00015786 <res 00000040>     <1> PTable_hd3: resb 64
000157C6 <res 00000040>     <1> PTable_ep0: resb 64
00015806 <res 00000040>     <1> PTable_ep1: resb 64
00015846 <res 00000040>     <1> PTable_ep2: resb 64
00015886 <res 00000040>     <1> PTable_ep3: resb 64
                            <1>
000158C6 <res 00000001>     <1> scount:  resb 1 ; 16/05/2016 (diskio.s, 'int33h:')
000158C7 <res 00000001>     <1> HD_LBA_yes: resb 1
000158C8 <res 00000001>     <1> PP_Counter: resb 1
000158C9 <res 00000001>     <1> EP_Counter: resb 1
                            <1>
000158CA <res 00000004>     <1> EP_StartSector: resd 1
000158CE <res 00000004>     <1>                 resd 1
000158D2 <res 00000004>     <1>                 resd 1
000158D6 <res 00000004>     <1>                 resd 1
                            <1>
000158DA <res 00000200>     <1> DOSBootSectorBuff: resb 512
                            <1>
                            <1> FAT_BuffDescriptor:
00015ADA <res 00000004>     <1> FAT_CurrentCluster: resd 1
00015ADE <res 00000001>     <1> FAT_BuffValidData: resb 1
00015ADF <res 00000001>     <1> FAT_BuffDrvName: resb 1
00015AE0 <res 00000002>     <1> FAT_BuffOffset: resw 1
00015AE2 <res 00000004>     <1> FAT_BuffSector: resd 1
                            <1>
00015AE6 <res 00000004>     <1> FAT_ClusterCounter: resd 1
00015AEA <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:
00015AEE <res 00000001>     <1> DirBuff_DRV: resb 1
00015AEF <res 00000001>     <1> DirBuff_FATType: resb 1
00015AF0 <res 00000001>     <1> DirBuff_ValidData: resb 1
00015AF1 <res 00000002>     <1> DirBuff_CurrentEntry: resw 1
00015AF3 <res 00000002>     <1> DirBuff_LastEntry: resw 1
00015AF5 <res 00000004>     <1> DirBuff_Cluster: resd 1
00015AF9 <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>
00015AFB <res 00000004>     <1> FreeClusterCount: resd 1
                            <1>
00015AFF <res 00000004>     <1> VolSize_Unit1:   resd 1
00015B03 <res 00000004>     <1> VolSize_Unit2:   resd 1
                            <1>
00015B07 <res 00000004>     <1> Vol_Tot_Sec_Str_Start:    resd 1
00015B0B <res 0000000A>     <1> Vol_Tot_Sec_Str:          resb 10
00015B15 <res 00000001>     <1> Vol_Tot_Sec_Str_End:      resb 1
00015B16 <res 00000001>     <1> resb 1
00015B17 <res 00000004>     <1> Vol_Free_Sectors_Str_Start: resd 1
00015B1B <res 0000000A>     <1> Vol_Free_Sectors_Str:     resb 10
00015B25 <res 00000001>     <1> Vol_Free_Sectors_Str_End:   resb 1
                            <1>
                            <1> ; 10/02/2016
00015B26 <res 00000001>     <1> RUN_CDRV: resb 1 ; CMD_INTR.ASM  ; 09/11/2011
                            <1>
                            <1> ; 24/01/2016
00015B27 <res 00000080>     <1> PATH_Array:     resb 128 ; DIR.ASM ; 09/10/2011
                            <1> ; 06/02/2016
00015BA7 <res 00000004>     <1> CCD_DriveDT:    resd 1 ; DIR.ASM ; (word)
00015BAB <res 00000001>     <1> CCD_Level:      resb 1 ; DIR.ASM
00015BAC <res 00000001>     <1> Last_Dir_Level: resb 1 ; DIR.ASM
                            <1> ;
00015BAD <res 00000002>     <1> CDLF_AttributesMask: resw 1 ; DIR.ASM
00015BAF <res 00000004>     <1> CDLF_FNAddress: resd 1 ; DIR.ASM (word)
00015BB3 <res 00000002>     <1> CDLF_DEType:    resw 1 ; DIR.ASM
                            <1> ;
00015BB5 <res 00000001>     <1> CD_COMMAND:     resb 1 ; DIR.ASM
                            <1>
00015BB6 <res 00000002>     <1> alignb 4
                            <1>
                            <1> ; 29/01/2016
00015BB8 <res 00000001>     <1> Program_Exit:   resb 1 ; CMD_INTR.ASM  ; 09/11/2011
                            <1>
                            <1> ;alignb 4
                            <1> ; 23/02/2016
00015BB9 <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
00015BBA <res 00000001>     <1> retry_count:    resb 1 ; DISK_IO.ASM ; 20/07/2011 (CHS_RetryCount)
00015BBB <res 00000001>     <1> disk_rw_err:    resb 1 ; DISK_IO.ASM ; (Disk_IO_err_code)
00015BBC <res 00000004>     <1> sector_count:   resd 1 ; DISK_IO.ASM ; (Disk_RW_SectorCount)
```

```
                            <1>
                            <1> ; 06/02/2016 (long name)
00015BC0 <res 00000002>     <1> FDE_AttrMask:     resw 1 ; DIR.ASM
00015BC2 <res 00000002>     <1> AmbiguousFileName: resw 1 ; DIR.ASM
00015BC4 <res 00000001>     <1> PreviousAttr:     resb 1 ; DIR.ASM
                            <1> ;
00015BC5 <res 00000001>     <1> LongNameFound:   resb 1        ; DIR.ASM
00015BC6 <res 00000001>     <1> LFN_EntryLength: resb 1   ; DIR.ASM
00015BC7 <res 00000001>     <1> LFN_CheckSum:    resb 1   ; DIR.ASM
00015BC8 <res 00000084>     <1> LongFileName:    resb 132 ; DIR.ASM
                            <1>
                            <1> ;PATH_Array_Ptr: resw 1 ; DIR.ASM
00015C4C <res 00000001>     <1> PATH_CDLevel:    resb 1 ; DIR.ASM
00015C4D <res 00000001>     <1> PATH_Level:      resb 1 ; DIR.ASM
                            <1>
                            <1> ; 07/02/2016
00015C4E <res 0000000D>     <1> Dir_File_Name:  resb 13 ; DIR.ASM ; 09/10/2011
                            <1>
                            <1> ; 10/02/2016
00015C5B <res 0000000D>     <1> Dir_Entry_Name: resb 13 ; DIR.ASM
                            <1>
                            <1> alignb 2
                            <1>
00015C68 <res 00000002>     <1> AttributesMask: resw 1 ; CMD_INTR.ASM ; 09/11/2011
                            <1>
                            <1> ; 10/02/2016 (128 bytes -> 126 bytes)
                            <1> ; 08/02/2016
                            <1> ;FFF Structure (128 bytes) ; DIR.ASM ; 09/10/2011
00015C6A <res 00000001>     <1> FindFile_Drv:           resb 1
00015C6B <res 00000041>     <1> FindFile_Directory:     resb 65
00015CAC <res 0000000D>     <1> FindFile_Name:          resb 13
                            <1> FindFile_LongNameEntryLength:
00015CB9 <res 00000001>     <1> FindFile_LongNameYes:   resb 1 ; Sign for longname procedures
                            <1> ;Above 80 bytes form
                            <1> ;TR-DOS Source/Destination File FullName Format/Structure
00015CBA <res 00000002>     <1> FindFile_AttributesMask:  resw 1
00015CBC <res 00000020>     <1> FindFile_DirEntry:      resb 32
00015CDC <res 00000004>     <1> FindFile_DirFirstCluster: resd 1
00015CE0 <res 00000004>     <1> FindFile_DirCluster:    resd 1
00015CE4 <res 00000002>     <1> FindFile_DirEntryNumber:  resw 1
00015CE6 <res 00000002>     <1> FindFile_MatchCounter:  resw 1
00015CE8 <res 00000002>     <1> FindFile_Reserved:      resw 1 ; 06/03/2016
                            <1>
00015CEA <res 00000004>     <1> First_Path_Pos: resd 1 ; DIR.ASM ; 09/10/2011
00015CEE <res 00000004>     <1> Last_Slash_Pos: resd 1 ; DIR.ASM
                            <1>
                            <1> ; 10/02/2016
00015CF2 <res 00000002>     <1> File_Count:     resw 1       ; DIR.ASM ; 09/10/2011
00015CF4 <res 00000002>     <1> Dir_Count:      resw 1
00015CF6 <res 00000004>     <1> Total_FSize:    resd 1
00015CFA <res 00000004>     <1> TFS_Dec_Begin:  resd 1
00015CFE <res 0000000A>     <1>                 resb 10
00015D08 <res 00000001>     <1> TFS_Dec_End:    resb 1
                            <1>
00015D09 <res 00000001>     <1> PrintDir_RowCounter: resb 1
                            <1>
00015D0A <res 00000002>     <1> alignb 4
                            <1> ; 15/02/2015 ('show' command variables)
00015D0C <res 00000004>     <1> Show_FDT:       resd 1
00015D10 <res 00000004>     <1> Show_LDDDT:     resd 1
00015D14 <res 00000004>     <1> Show_Cluster:   resd 1
00015D18 <res 00000004>     <1> Show_FileSize:  resd 1
00015D1C <res 00000004>     <1> Show_FilePointer: resd 1
00015D20 <res 00000002>     <1> Show_ClusterPointer: resw 1
00015D22 <res 00000002>     <1> Show_ClusterSize: resw 1
00015D24 <res 00000001>     <1> Show_RowCount:  resb 1
                            <1>
00015D25 <res 00000003>     <1> alignb 4
                            <1> ; 21/02/2016
00015D28 <res 00000004>     <1> DelFile_FNPointer:    resd 1 ; ; CMD_INTR.ASM (word) ; 09/11/2011
                            <1> ; 27/02/2016
                            <1> ; DIR.ASM (09/10/2011)
00015D2C <res 00000004>     <1> DelFile_FCluster:     resd 1
00015D30 <res 00000002>     <1> DelFile_EntryCounter:  resw 1
00015D32 <res 00000001>     <1> DelFile_LNEL:         resb 1
00015D33 <res 00000001>     <1> resb 1
                            <1>
                            <1> ; DIR.ASM
00015D34 <res 00000004>     <1> mkdir_DirName_Offset:  resd 1
00015D38 <res 00000004>     <1> mkdir_FFCluster:       resd 1
00015D3C <res 00000004>     <1> mkdir_LastDirCluster:  resd 1
00015D40 <res 00000004>     <1> mkdir_FreeSectors:     resd 1
00015D44 <res 00000002>     <1> mkdir_attrib:          resw 1
00015D46 <res 00000001>     <1> mkdir_SecPerClust:     resb 1
00015D47 <res 00000001>     <1> mkdir_add_new_cluster: resb 1
00015D48 <res 0000000D>     <1> mkdir_Name:            resb 13
00015D55 <res 00000002>     <1> resw 1 ; 01/03/2016
                            <1> ; 27/02/2016
00015D57 <res 00000001>     <1> RmDir_MultiClusters:   resb 1
00015D58 <res 00000004>     <1> RmDir_DirEntryOffset:  resd 1 ; 01/03/2016 (word -> dword)
00015D5C <res 00000004>     <1> RmDir_ParentDirCluster: resd 1
00015D60 <res 00000004>     <1> RmDir_DirLastCluster:   resd 1
00015D64 <res 00000004>     <1> RmDir_PreviousCluster:  resd 1
                            <1> ; 22/02/2016
00015D68 <res 00000001>     <1> UPDLMDT_CDirLevel:     resb 1
00015D69 <res 00000004>     <1> UPDLMDT_CDirFCluster:  resd 1
                            <1>
00015D6D <res 00000003>     <1> alignb 4
                            <1> ; DRV_FAT.ASM ; 21/08/2011
00015D70 <res 00000004>     <1> gffc_next_free_cluster:  resd 1
00015D74 <res 00000004>     <1> gffc_first_free_cluster: resd 1
00015D78 <res 00000004>     <1> gffc_last_free_cluster:  resd 1
                            <1>
                            <1> ;29/04/2016
                            <1> Cluster_Index: ; resd 1
                            <1> ; 22/02/2016
00015D7C <res 00000004>     <1> ClusterValue:   resd 1
                            <1> ; 04/03/2016
```

```
258 00015D80 <res 00000001>      <1> Attributes:    resb 1
259                              <1> ;;CFS_error:  resb 1 ;; 01/03/2016
260 00015D81 <res 00000001>      <1> resb 1
261 00015D82 <res 00000001>      <1> CFS_OPType: resb 1
262 00015D83 <res 00000001>      <1> CFS_Drv:    resb 1
263 00015D84 <res 00000004>      <1> CFS_CC:     resd 1
264 00015D88 <res 00000004>      <1> CFS_FAT32FSINFOSEC: resd 1
265 00015D8C <res 00000004>      <1> CFS_FAT32FC: resd 1
266                              <1>
267                              <1> ; 27/02/2016
268                              <1> ;alignb 4
269 00015D90 <res 00000004>      <1> glc_prevcluster: resd 1 ; DRV_FAT.ASM (21/08/2011)
270                              <1> ; 22/10/2016
271 00015D94 <res 00000004>      <1> glc_index:      resd 1 ;  Last Cluster Index (22/10/2016)
272                              <1>
273                              <1> ; DIR.ASM
274 00015D98 <res 00000002>      <1> DLN_EntryNumber: resw 1
275 00015D9A <res 00000001>      <1> DLN_40h:  resb 1
276                              <1> ; 28/02/2016
277 00015D9B <res 00000001>      <1> TCC_FATErr:     resb 1 ; DRV_FAT.ASM
278                              <1>
279                              <1> alignb 4
280                              <1> ; DIR.ASM (09/10/2011)
281 00015D9C <res 00000002>      <1> LCDE_EntryIndex: resw 1 ; LCDE_EntryOffset
282 00015D9E <res 00000002>      <1> LCDE_ClusterSN:  resw 1
283 00015DA0 <res 00000004>      <1> LCDE_Cluster:   resd 1
284 00015DA4 <res 00000004>      <1> LCDE_ByteOffset: resd 1
285                              <1>
286                              <1> ;alignb4
287                              <1> ; 06/03/2016 (word -> dword)
288                              <1> ; CMD_INTR.ASM (01/08/2010)
289 00015DA8 <res 00000004>      <1> SourceFilePath:     resd 1
290 00015DAC <res 00000004>      <1> DestinationFilePath: resd 1
291                              <1>
292                              <1> ;alignb 4
293                              <1> ; 06/03/2016
294                              <1> ; FILE.ASM (09/10/2011)
295                              <1> ;Source File Structure (same with 'Find File' Structure)
296 00015DB0 <res 00000001>      <1> SourceFile_Drv:             resb 1
297 00015DB1 <res 00000041>      <1> SourceFile_Directory:       resb 65
298 00015DF2 <res 0000000D>      <1> SourceFile_Name:            resb 13
299                              <1> SourceFile_LongNameEntryLength:
300 00015DFF <res 00000001>      <1> SourceFile_LongNameYes:             resb 1 ; Sign for longname procedures
301                              <1> ;Above 80 bytes
302                              <1> ;is TR-DOS Source File FullName Format/Structure
303 00015E00 <res 00000002>      <1> SourceFile_AttributesMask:   resw 1
304 00015E02 <res 00000020>      <1> SourceFile_DirEntry:         resb 32
305 00015E22 <res 00000004>      <1> SourceFile_DirFirstCluster:  resd 1
306 00015E26 <res 00000004>      <1> SourceFile_DirCluster:       resd 1
307 00015E2A <res 00000002>      <1> SourceFile_DirEntryNumber:   resw 1
308 00015E2C <res 00000002>      <1> SourceFile_MatchCounter:     resw 1
309                              <1> ; 16/03/2016
310 00015E2E <res 00000001>      <1> SourceFile_SecPerClust:             resb 1
311 00015E2F <res 00000001>      <1> SourceFile_Reserved:         resb 1
312                              <1> ; Above is 128 bytes
313                              <1>
314                              <1> ;Destination File Structure (same with 'Find File' Structure)
315 00015E30 <res 00000001>      <1> DestinationFile_Drv:         resb 1
316 00015E31 <res 00000041>      <1> DestinationFile_Directory:   resb 65
317 00015E72 <res 0000000D>      <1> DestinationFile_Name:        resb 13
318                              <1> DestinationFile_LongNameEntryLength:
319 00015E7F <res 00000001>      <1> DestinationFile_LongNameYes: resb 1 ; Sign for longname procedures
320                              <1> ;Above 80 bytes
321                              <1> ;is TR-DOS Destination File FullName Format/Structure
322 00015E80 <res 00000002>      <1> DestinationFile_AttributesMask: resw 1
323 00015E82 <res 00000020>      <1> DestinationFile_DirEntry:    resb 32
324 00015EA2 <res 00000004>      <1> DestinationFile_DirFirstCluster: resd 1
325 00015EA6 <res 00000004>      <1> DestinationFile_DirCluster:  resd 1
326 00015EAA <res 00000002>      <1> DestinationFile_DirEntryNumber: resw 1
327 00015EAC <res 00000002>      <1> DestinationFile_MatchCounter: resw 1
328                              <1> ; 16/03/2016
329 00015EAE <res 00000001>      <1> DestinationFile_SecPerClust: resb 1
330 00015EAF <res 00000001>      <1> DestinationFile_Reserved:    resb 1
331                              <1> ; Above is 128 bytes
332                              <1>
333                              <1> ; 24/04/2016
334 00015EB0 <res 00000002>      <1> resw 1
335                              <1>
336                              <1> ; 10/03/2016
337                              <1> ; FILE.ASM
338 00015EB2 <res 00000001>      <1> move_cmd_phase:    resb 1
339 00015EB3 <res 00000001>      <1> msftdf_sf_df_drv:  resb 1
340 00015EB4 <res 00000004>      <1> msftdf_drv_offset: resd 1
341                              <1>
342                              <1> ; 11/03/2016
343                              <1> ; DRV_FAT.ASM (21/08/2011)
344 00015EB8 <res 00000004>      <1> FAT_anc_LCluster:  resd 1
345 00015EBC <res 00000004>      <1> FAT_anc_FFCluster: resd 1
346                              <1>
347                              <1> ;alignb 4
348                              <1>
349                              <1> ; 14/03/2016
350                              <1> ; TRDOS 386 = TRDOS v2.0 feature only !
351                              <1> ; 'allocate_memory_block' in 'memory.s'
352 00015EC0 <res 00000004>      <1> mem_ipg_count:  resd 1 ; page count (for contiguous allocation)
353 00015EC4 <res 00000004>      <1> mem_pg_count:   resd 1 ; page count (for count down)
354 00015EC8 <res 00000004>      <1> mem_aperture:   resd 1 ; contiguous free pages (current)
355 00015ECC <res 00000004>      <1> mem_max_aperture: resd 1 ; maximum value of contiguous free pages
356 00015ED0 <res 00000004>      <1> mem_pg_pos:     resd 1 ; mem. position (page #) of current aperture
357 00015ED4 <res 00000004>      <1> mem_max_pg_pos: resd 1 ; mem. position (page #) of max. aperture
358                              <1>
359                              <1> ; 15/03/2016
360                              <1> ; FILE.ASM ('copy_source_file_to_destination_file')
361 00015ED8 <res 00000001>      <1> copy_cmd_phase:       resb 1
362 00015ED9 <res 00000001>      <1> csftdf_rw_err:        resb 1
363 00015EDA <res 00000001>      <1> DestinationFileFound: resb 1
364 00015EDB <res 00000001>      <1> csftdf_cdrv:          resb 1
365 00015EDC <res 00000004>      <1> csftdf_filesize:      resd 1
```

```
                            <1> ; TRDOS386 (TRDOS v2.0)
00015EE0 <res 00000004>     <1> csftdf_sf_mem_addr:   resd 1
00015EE4 <res 00000004>     <1> csftdf_sf_mem_bsize:  resd 1
                            <1> ;
                            <1>
00015EE8 <res 00000004>     <1> csftdf_sf_cluster:    resd 1 ; 16/03/2016
00015EEC <res 00000004>     <1> csftdf_df_cluster:    resd 1
                            <1> ; 16/03/2016
00015EF0 <res 00000004>     <1> csftdf_r_size:        resd 1
00015EF4 <res 00000004>     <1> csftdf_w_size:        resd 1
00015EF8 <res 00000004>     <1> csftdf_sf_rbytes:     resd 1
00015EFC <res 00000004>     <1> csftdf_df_wbytes:     resd 1
00015F00 <res 00000001>     <1> csftdf_percentage:    resb 1
                            <1> ; 17/03/2016
00015F01 <res 00000001>     <1> csftdf_videopage:     resb 1
00015F02 <res 00000002>     <1> csftdf_cursorpos:     resw 1
00015F04 <res 00000004>     <1> csftdf_sf_drv_dt:     resd 1
00015F08 <res 00000004>     <1> csftdf_df_drv_dt:     resd 1
                            <1>
                            <1> ; 21/03/2016
                            <1> ; 20/03/2016
                            <1> ; FILE.ASM
00015F0C <res 00000004>     <1> createfile_Name_Offset:  resd 1
00015F10 <res 00000004>     <1> createfile_FreeSectors:  resd 1
00015F14 <res 00000004>     <1> createfile_size:         resd 1
00015F18 <res 00000004>     <1> createfile_FFCluster:    resd 1 ; 11/03/2016
00015F1C <res 00000004>     <1> createfile_LastDirCluster: resd 1
00015F20 <res 00000004>     <1> createfile_Cluster:      resd 1
00015F24 <res 00000004>     <1> createfile_PCluster:     resd 1
00015F28 <res 00000001>     <1> createfile_attrib:     resb 1
00015F29 <res 00000001>     <1> createfile_SecPerClust:  resb 1
00015F2A <res 00000002>     <1> createfile_DirIndex:     resw 1
00015F2C <res 00000004>     <1> createfile_CCount:     resd 1
00015F30 <res 00000002>     <1> createfile_BytesPerSec:       resw 1 ; 23/03/2016
00015F32 <res 00000001>     <1> createfile_wfc:         resb 1
00015F33 <res 00000001>     <1> createfile_UpdatePDir:  resb 1 ; 31/03/2016
                            <1>
                            <1> ;alignb 4
                            <1>
                            <1> ; 11/04/2016
00015F34 <res 00000002>     <1> env_var_length:   resw 1
                            <1>
00015F36 <res 00000002>     <1> alignb 4
                            <1>
                            <1> ; 25/04/2016
00015F38 <res 00000001>     <1> readi.valid:    resb 1 ; valid data (>0 = valid for readi)
00015F39 <res 00000001>     <1> readi.drv:      resb 1 ; drive number (0, 1,2,3,4..)
00015F3A <res 00000001>     <1> readi.spc:      resb 1 ; sectors per cluster for 'readi' drive
00015F3B <res 00000001>     <1> readi.s_index:  resb 1 ; sector index in current cluster (buffer)
00015F3C <res 00000004>     <1> readi.sector:   resd 1 ; current disk sector
00015F40 <res 00000002>     <1> readi.bpc:      resw 1 ; bytes per cluster - 1
00015F42 <res 00000002>     <1> readi.offset:   resw 1 ; byte offset in cluster buffer
00015F44 <res 00000004>     <1> readi.cluster:  resd 1 ; current cluster number
00015F48 <res 00000004>     <1> readi.c_index:  resd 1 ; cluster index of the current cluster (0,1,2,3..)
00015F4C <res 00000004>     <1> readi.fclust:   resd 1 ; first cluster of the current cluster
00015F50 <res 00000004>     <1> readi.fs_index: resd 1 ; sector index in disk/file section (for Singlix FS)
                            <1> ;readi.buffer:  resd 1 ; readi sector buffer address
                            <1>
                            <1> ;alignb 4
                            <1>
00015F54 <res 00000001>     <1> writei.valid:   resb 1 ; valid data (>0 = valid for writei)
00015F55 <res 00000001>     <1> writei.drv:     resb 1 ; drive number (0, 1,2,3,4..)
00015F56 <res 00000001>     <1> writei.spc:     resb 1 ; sectors per cluster for 'writei' drive
00015F57 <res 00000001>     <1> writei.s_index: resb 1 ; sector index in current cluster (buffer)
00015F58 <res 00000004>     <1> writei.sector:  resd 1 ; current disk sector
00015F5C <res 00000002>     <1> writei.bpc:     resw 1 ; bytes per cluster - 1
00015F5E <res 00000002>     <1> writei.offset:  resw 1 ; byte offset in cluster buffer
00015F60 <res 00000004>     <1> writei.cluster: resd 1 ; current cluster number
00015F64 <res 00000004>     <1> writei.c_index: resd 1 ; cluster index of the current cluster (0,1,2,3..)
00015F68 <res 00000004>     <1> writei.fclust:  resd 1 ; first cluster of the current cluster
00015F6C <res 00000004>     <1> writei.fs_index: resd 1 ; sector index in disk/file section (for Singlix FS)
                            <1> ;writei.buffer: resd 1 ; writei sector buffer address
00015F70 <res 00000004>     <1> writei.lclust:  resd 1 ; writei last cluster (mget_w) ; 23/10/2016
00015F74 <res 00000004>     <1> writei.l_index: resd 1 ; writei last cluster index (mget_w) ; 23/10/2016
00015F78 <res 00000001>     <1> writei.ofn:     resb 1 ; open file number (to be written) ; 23/10/2016
                            <1>
00015F79 <res 00000003>     <1> alignb 4
                            <1>
                            <1> ; 29/04/2016
00015F7C <res 00000004>     <1> Run_CDirFC:     resd 1
00015F80 <res 00000001>     <1> Run_Auto_Path:  resb 1
00015F81 <res 00000001>     <1> Run_Manual_Path: resb 1 ; 0 -> auto path sequence needed
00015F82 <res 00000001>     <1> EXE_ID:         resb 1
00015F83 <res 00000001>     <1> EXE_dot: resb 1
                            <1>
                            <1> ; 06/05/2016
00015F84 <res 00000004>     <1> mainprog_return_addr: resd 1
00015F88 <res 00000004>     <1> last_error:     resd 1  ; this will be used to return error code to MainProg
                            <1>                 ; 'lasterror' keyword will be used later to get the
                            <1>                 ; last error code/number/status.
                            <1> ; 12/05/2016
00015F8C <res 00000004>     <1> video_eax:      resd 1  ; eax return value of video function
                            <1>
                            <1> ; 01/06/2016
00015F90 <res 00000004>     <1> user_buffer:    resd 1  ; 'diskio.s' (INT 33h, Function 08h, floppy disk type)
                            <1>
                            <1> ; 21/05/2016 - TRDOS 386 ('swap/switch', 'rswap', [u.pri])
00015F94 <res 00000001>     <1> priority:       resb 1  ; running priority level of process (0,1,2)
                            <1>                 ; (run queue which is process comes from)
                            <1> ; 22/05/2016 - TRDOS 386 ('set_run_sequence', 'rtc_int', 'u_timer')
00015F95 <res 00000001>     <1> p_change:       resb 1  ; process change status (for timer events)
                            <1> ; 23/05/2016 - TRDOS 386 ('clock')
00015F96 <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
00015F97 <res 00000001>     <1> timer_events:   resb 1  ; number of (active) timer events, <= 16
                            <1>
                            <1> ; 24/06/2016
```

```
00015F98 <res 00000001>      <1> w_str_cmd:     resb 1 ; WRITE_STRING command (0,1,2,3) ; video.s
00015F99 <res 00000001>      <1> p_crt_mode:    resb 1  ; previous video mode (=3 or 0), backup mark/sign
                             <1> ; 26/06/2016
00015F9A <res 00000001>      <1> p_crt_page:    resb 1  ; previous active page (for 'set_mode')
                             <1> ; 04/07/2016
00015F9B <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
00015F9C <res 00000002>      <1> CRT_LEN: resw 1  ; length of regen buffer in bytes
00015F9E <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)
00015FAE <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
00015FB2 <res 00000001>      <1> VGA_MTYPE:     resb 1  ; 0=CTEXT,1=MTEXT,2=CGA,3=PLANAR1,4=PLANAR4,5=LINEAR
                             <1>
                             <1> ; 23/10/2016
00015FB3 <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
00015FB4 <res 00000004>      <1> FFF_UBuffer:   resd 1  ; User's buffer address for FFF & FNF system calls
                             <1> ; 15/10/2016
00015FB8 <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)
00015FB9 <res 00000001>      <1> FFF_Attrib:    resb 1 ; Find First File attributes for Find Next File (LB)
00015FBA <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)
00015FBB <res 00000001>      <1> SWP_inv_fname:  resb 1 ; Set Working Path - Invalid File Name
00015FBC <res 00000002>      <1> SWP_Mode:      resw 1 ; Set Working Path - Mode
00015FBE <res 00000001>      <1> SWP_DRV: resb 1 ; Set Working Path - Drive
00015FBF <res 00000001>      <1> SWP_DRV_chg:   resb 1 ; Set Working Path - Drive Change
                             <1>
                             <1> ; 27/02/2017
00015FC0 <res 00000001>      <1> fpready: resb 1 ; '80387 fpu is ready' flag
                             <1>
                             <1> ; 08/10/2016
00015FC1 <res 00000009>      <1> device_name:    resb 9  ; capitalized (and zero padded) device canem
                             <1>                ; (example: "TTY0",0,0,0,0,0")
                             <1>
00015FCA <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)
00015FCC <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>
00015FEC <res 00000040>      <1> IDEV_NAME:  resb 8*NUMIDEV
                             <1>                ; 8 byte names of installable device drivers
                             <1>
0001602C <res 00000008>      <1> IDEV_TYPE:  resb NUMIDEV ; Driver type of installable device drivers
```

```
582 00016034 <res 00000008>      <1> IDEV_FLAGS: resb NUMIDEV ; Device access parameters for installable
583                              <1>                          ; device drivers (These values are set while
584                              <1>                        ; the device driver is being loaded.)
585 0001603C <res 00000020>      <1> IDEV_OADDR: resd NUMIDEV ; open function addr for installable dev driver
586 0001605C <res 00000020>      <1> IDEV_CADDR: resd NUMIDEV ; close function addr for installable dev driver
587 0001607C <res 00000020>      <1> IDEV_RADDR: resd NUMIDEV ; read function addr for installable dev driver
588 0001609C <res 00000020>      <1> IDEV_WADDR: resd NUMIDEV ; write function addr for installable dev driver
589                              <1>
590                              <1> ; 08/10/2016
591                              <1> ; 07/10/2016
592                              <1> ; Device Open and Access parameters
593 000160BC <res 0000001E>      <1> DEV_ACCESS:     resb NUMOFDEVICES   ; bit 0 = accessable by normal users
594                              <1>                                     ; bit 1 = read access permission
595                              <1>                                     ; bit 2 = write access permission
596                              <1>                                     ; bit 3 = IOCTL permission to users
597                              <1>                                     ; bit 4 = block device if it is set
598                              <1>                                     ; bit 5 = 16 bit or 1024 byte data
599                              <1>                                     ; bit 6 = 32 bit or 2048 byte data
600                              <1>                                     ; bit 7 = installable device driver
601 000160DA <res 0000001E>      <1> DEV_R_OWNER:    resb NUMOFDEVICES   ; Reading owner no (u.uid) of devices
602 000160F8 <res 0000001E>      <1> DEV_R_OPENCOUNT: resb NUMOFDEVICES   ; Reading open count
603 00016116 <res 0000001E>      <1> DEV_W_OWNER:    resb NUMOFDEVICES   ; Writing owner no (u.uid) of devices
604 00016134 <res 0000001E>      <1> DEV_W_OPENCOUNT: resb NUMOFDEVICES   ; Writing open count
605 00016152 <res 0000001E>      <1> DEV_DRIVER:     resb NUMOFDEVICES   ; device driver number (1 to 7Fh)
606                              <1>                                     ; *if bit 7 is set (80 to FFh)
607                              <1>                                     ; *if it is installable device driver
608                              <1>                                     ; *index (0 to 7Fh)
609                              <1>                                     ; otherwise it is kernel device index
610 00016170 <res 0000001E>      <1> DEV_OPENMODE:   resb NUMOFDEVICES   ; 1 = read mode
611                              <1>                                     ; 2 = write mode
612                              <1>                                     ; 3 = read & write
613                              <1>                                     ; 0 = not open (free)
614 0001618E <res 00000078>      <1> DEV_NAME_PTR:    resd NUMOFDEVICES   ; pointers to name addresses of drivers
615                              <1>                                     ; Address base: KDEV_NAME+
616                              <1>                                     ; or IDEV_NAME+
617 00016206 <res 00000078>      <1> DEV_R_POINTER:   resd NUMOFDEVICES   ; reading pointer, writing pointer
618 0001627E <res 00000078>      <1> DEV_W_POINTER:   resd NUMOFDEVICES   ; sector number if block device
619                              <1>                                     ; character offset if char device
620 000162F6 <res 00000002>      <1> alignb 4
621                              <1>
622                              <1> ; 06/10/2016
623                              <1> ; Open File Parameters
624 000162F8 <res 00000028>      <1> OF_FCLUSTER:    resd OPENFILES  ; First clusters of open files
625 00016320 <res 0000000A>      <1> OF_DRIVE:       resb OPENFILES  ; Logical DOS drive numbers of open files
626 0001632A <res 0000000A>      <1> OF_MODE: resb OPENFILES  ; Open mode (1 = read, 2 = write, 3 = r&w)
627 00016334 <res 0000000A>      <1> OF_STATUS:      resb OPENFILES  ; (bit 0 = read, bit 1 = write)
628 0001633E <res 0000000A>      <1> OF_OPENCOUNT:   resb OPENFILES  ; Open counts of open files
629 00016348 <res 00000028>      <1> OF_POINTER:     resd OPENFILES      ; File seek/read/write pointer
630 00016370 <res 00000028>      <1> OF_SIZE: resd OPENFILES       ; File sizes of open files (in bytes)
631 00016398 <res 00000028>      <1> OF_DIRFCLUSTER: resd OPENFILES  ; Directory First Clusters of open files
632 000163C0 <res 00000028>      <1> OF_DIRCLUSTER:  resd OPENFILES  ; Directory (Entry) Clusters of open files
633 000163E8 <res 00000028>      <1> OF_VOLUMEID:    resd OPENFILES  ; Vol ID for removable drives of open files
634 00016410 <res 00000028>      <1> OF_CCLUSTER:    resd OPENFILES  ; Current clusters of open files
635 00016438 <res 00000028>      <1> OF_CCINDEX:     resd OPENFILES  ; Cluster index numbers of current clusters
636                              <1> ; 24/10/2016
637 00016460 <res 00000014>      <1> OF_DIRENTRY:    resw OPENFILES  ; Directory entry index no. in dir cluster
638                              <1>                         ; Sector index = entry index / 16
639                              <1> ;alignb 2
640                              <1>
641 00016474 <res 00000060>      <1> DTA:            resd 24             ; Find First File data transfer area
642                              <1>
643                              <1> ; 19/12/2016
644 000164D4 <res 00000001>      <1> tcallback:      resb 1        ; Timer callback method flag for 'systimer'
645 000164D5 <res 00000001>      <1> trtc:           resb 1        ; Timer interrupt type flag for 'systimer'
646                              <1> ; 20/02/2017
647 000164D6 <res 00000001>      <1> no_page_swap:   resb 1        ; Swap lock for Signal Response Byte pages
648                              <1> ;;15/01/2017
649                              <1> ; 02/01/2017
650                              <1> ;;intflg:       resb 1        ; software interrupt in progress signal
651                              <1>                         ; (for timer interrupt)
652                              <1>
653 000164D7 <res 00000001>      <1> alignb 4
654                              <1> ; 13/04/2017
655 000164D8 <res 0000001E>      <1> DEV_INTR:       resb NUMOFDEVICES ; Device Interrupt (IRQ) number + 1
656                              <1>                         ; (0= not available, 1= IRQ 0, 16= IRQ 15)
657 000164F6 <res 00000040>      <1> DEV_INT_HNDLR:  resd 16             ; Device Interrupt Handler addr, if > 0
658                              <1>
659                              <1>
660                              <1> ;alignb 4
661                              <1>
662                              <1> ; 26/02/2017 ; IRQ Callback parameters ('syscalbac')
663                              <1> ;Index: ; 0 to 8
664                              <1> ; 0 = IRQ3, 1 = IRQ4, 2 = IRQ5, 3 = IRQ7
665                              <1> ; 4 = IRQ9, 5 = IRQ10, 6 = IRQ11, 7 = IRQ12, 8 = IRQ13
666 00016536 <res 00000009>      <1> IRQ.owner:      resb 9        ; owner, 0 = free, >0 = [u.uno]
667 0001653F <res 00000009>      <1> IRQ.dev: resb 9        ; 0 = default/kernel, >0 = device number
668 00016548 <res 00000009>      <1> IRQ.method:     resb 9              ; 0 = Signal Response Byte, 1 = Callback
669 00016551 <res 00000009>      <1> IRQ.srb: resb 9               ; Signal Response/Return Byte value
670 0001655A <res 00000024>      <1> IRQ.addr:       resd 9        ; Rignal Response Byte address (physical)
671                              <1>                         ; or Callback service address (virtual)
672                              <1> ; 28/02/2017
673 0001657E <res 00000004>      <1> IRQ_cr3: resd 1        ; for saving cr3 register in IRQ handler
674 00016582 <res 00000001>      <1> IRQnum:         resb 1        ; IRQ number for IRQ handler (trdosk8.s)
675                              <1>
676                              <1> ; 10/04/2017
677                              <1> ; 03/04/2017
678                              <1> ; UNINITIALIZED AUDIO DATA
679 00016583 <res 00000001>      <1> alignb 4
680 00016584 <res 00000001>      <1> audio_pci:      resb 1
681 00016585 <res 00000001>      <1> audio_device:   resb 1
682 00016586 <res 00000001>      <1> audio_mode:     resb 1
683 00016587 <res 00000001>      <1> audio_intr:     resb 1
684 00016588 <res 00000001>      <1> audio_busy:     resb 1  ; Busy flag for audio irq ; 21/04/2017
685 00016589 <res 00000001>      <1> audio_reserved: resb 1
686 0001658A <res 00000002>      <1> audio_io_base:  resw 1        ; Base I/O address of audio device
687 0001658C <res 00000004>      <1> audio_dev_id:   resd 1 ; BUS/DEV/FN ; 00000000BBBBBBBBDDDDDFFF00000000
688 00016590 <res 00000004>      <1> audio_vendor:   resd 1
689 00016594 <res 00000004>      <1> audio_stats_cmd: resd 1
```

```
690                                     <1> ;
691 00016598 <res 00000004>             <1> audio_buffer:   resd 1 ; virtual address of user's audio buffer
692 0001659C <res 00000004>             <1> audio_p_buffer: resd 1 ; Physical address of user's audio buffer
693 000165A0 <res 00000004>             <1> audio_buff_size: resd 1 ; user's audio buffer size (half buffer size)
694 000165A4 <res 00000004>             <1> audio_dma_buff: resd 1  ; dma buffer address
695 000165A8 <res 00000004>             <1> audio_dmabuff_size: resd 1 ; dma buffer size (2 * half buffer size)
696 000165AC <res 00000001>             <1> audio_flag:     resb 1  ; dma buffer flag (1st half = 0, 2nd half = 1)
697 000165AD <res 00000001>             <1> audio_user:     resb 1 ; user number of the owner
698 000165AE <res 00000001>             <1> audio_cb_mode:  resb 1 ; 0 = signal response byte method
699                                     <1>                 ; 1 = callback method
700                                     <1>                 ; 2 = s.r.b. method with auto increment
701 000165AF <res 00000001>             <1> audio_srb:      resb 1 ; signal response byte value
702 000165B0 <res 00000004>             <1> audio_cb_addr:  resd 1  ; callback service address or s.r.b. address
703                                     <1>                 ; (s.r.b. addr is physical, cbs addr is virtual)
704                                     <1>
705 000165B4 <res 00000001>             <1> audio_bps:      resb 1  ; selected mode: 8 bit, 16 bit
706 000165B5 <res 00000001>             <1> audio_stmo:     resb 1 ; selected mode: mono /stereo
707 000165B6 <res 00000002>             <1> audio_freq:     resw 1 ; sampling rate
708                                     <1>
709                                     <1> ; 21/04/2017
710 000165B8 <res 00000001>             <1> audio_play_cmd: resb 1  ; Play/Stop command (1 = play, 0 = stop)
711                                     <1> audio_civ: ; 28/05/2017 ; Current Buffer Index (AC'97)
712 000165B9 <res 00000001>             <1> audio_flag_eol: resb 1  ; End of Link status (vt8233, EOL/FLAG)
713                                     <1>
714                                     <1> audio_master_volume:
715 000165BA <res 00000001>             <1> audio_master_volume_l: resb 1 ; sound volume (lineout) left channel
716 000165BB <res 00000001>             <1> audio_master_volume_r: resb 1 ; sound volume (lineout) right channel
717                                     <1>
718                                     <1> alignb 4
719                                     <1> ; 28/05/2017
720                                     <1> ; AC'97 Audio Controller Base Adress Registers
721 000165BC <res 00000002>             <1> NAMBAR:         resw 1 ; Native Audio Mixer Base Address
722 000165BE <res 00000002>             <1> NABMBAR: resw 1 ; Native Audio Bus Mastering Base Address
723                                     <1>
724                                     <1> ;alignb 4
725                                     <1> ; 21/04/2017
726 000165C0 <res 00000400>             <1> audio_bdl_buff: resd 32*8  ; VT8233 (AC97) BDL Buffer Size
727                                     <1> ; 12/05/2017
728 000169C0 <res 00000004>             <1> base_addr:      resd 1 ; 'direct_memory_access' (memory.s)
729                                     <1>
730                                     <1> ; 28/08/2017
731                                     <1> ; 20/08/2017
732 000169C4 <res 00000001>             <1>         resb 1  ;
733 000169C5 <res 00000001>             <1> dma_user:       resb 1 ; user number for sysdma
734 000169C6 <res 00000001>             <1> dma_channel:    resb 1 ; dma channel for sysdma
735 000169C7 <res 00000001>             <1> dma_mode:       resb 1  ; dma mode for sysdma
736 000169C8 <res 00000004>             <1> dma_addr:       resd 1 ; dma buffer physical addr for sysdma
737 000169CC <res 00000004>             <1> dma_size:       resd 1  ; dma buffer size (in bytes) for sysdma
738 000169D0 <res 00000004>             <1> dma_start:      resd 1  ; dma start address for sysdma
739 000169D4 <res 00000004>             <1> dma_count:      resd 1  ; dma count (in bytes) for sysdma
740                                     <1>
741 000169D8 <res 00009628>             <1> alignb 65536
742                                     <1> ; 09/08/2017
743                                     <1> ; 12/05/2017
744 00020000 <res 00010000>             <1> sb16_dma_buffer: resb 65536 ; DMA buffer for sb16 audio playing.
2829                                        ; 24/01/2016
2830                                        %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.
  35                                    <1>
  36 00030000 <res 00000002>            <1>   i.flgs:        resw 1
  37 00030002 <res 00000001>            <1>   i.nlks:        resb 1
  38 00030003 <res 00000001>            <1>   i.uid:  resb 1
  39                                    <1>         ;i.size:  resw 1 ; size
  40 00030004 <res 00000002>            <1>   resw 1 ; 29/04/2016
  41 00030006 <res 00000010>            <1>   i.dskp:        resw 8 ; 16 bytes
  42 00030016 <res 00000004>            <1>   i.ctim:        resd 1
  43 0003001A <res 00000004>            <1>   i.mtim:        resd 1
  44 0003001E <res 00000002>            <1>   i.rsvd:  resw 1 ; Reserved (ZERO/Undefined word for UNIX v1.)
  45                                    <1>
  46                                    <1> I_SIZE   equ $ - inode
  47                                    <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 start address of user's memory space (for sys exec)
00030444 <res 00000004>         <1> nbase:   resd 1 ; physical base address for 'namei' & 'sysexec'
00030448 <res 00000002>         <1> ncount: resw 1  ; remain byte count in page for 'namei' & 'sysexec'
0003044A <res 00000002>         <1> argc:    resw 1 ; argument count for 'sysexec'
0003044C <res 00000004>         <1> argv:    resd 1 ; argument list (recent) address for 'sysexec'
                                <1>
                                <1> ; 03/06/2015 - Retro UNIX 386 v1 Beginning
                                <1> ; 07/04/2013 - 31/07/2013 - Retro UNIX 8086 v1
00030450 <res 00000001>         <1> rw:      resb 1 ;; Read/Write sign (iget)
                                <1>
```

```
 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
2831
2832 00030459 <res 00000003>                alignb 4
2833
2834                                        ; 23/05/2016 (TRDOS 386)
2835                                        ; 14/10/2015 (Retro UNIX 386 v1, 'unix386.s')
2836 0003045C <res 00000004>                cr3reg:   resd 1  ; cr3 register content at the beginning of the timer
2837                                                  ; (or RTC) interrupt handler.
2838
2839                                        ; 10/12/2016 (callback)
2840                                        ; 10/06/2016
2841                                        ; 19/05/2016
2842                                        ; 18/05/2016 - TRDOS 386 feature only !
2843 00030460 <res 00000100>                timer_set: resd 16*4   ; 256 bytes memory space for 16 timer events
2844                                          ; Timer Event Structure: (max. 16 timer events, 16*16 bytes)
2845                                          ;       Owner:              resb 1 ; 0 = free
2846                                          ;                                  ;>0 = process number (u.uno)
2847                                          ;      Callback:     resb 1 ; 0 = response byte address (phy)
2848                                          ;                                   1 = callback address (virtual)
2849                                          ;      Interrupt:      resb 1 ; 0 = Timer interrupt (or none)
2850                                          ;                                  ; 1 = Real Time Clock interrupt
2851                                          ;      Response:       resb 1 ; 0 to 255, signal return value
2852                                          ;      Count Limit:  resd 1 ; count of ticks (total/set)
2853                                          ;      Current Count:       resd 1 ; count of ticks (current)
2854                                          ;      Response Addr:  resd 1 ; response byte (pointer) address
2855                                          ;                                  ; (or callback -user service- address)
2856
2857                                        ;; Memory (swap) Data (11/03/2015)
2858                                        ; 09/03/2015
2859 00030560 <res 00000002>                swpq_count: resw 1 ; count of pages on the swap queue
2860 00030562 <res 00000004>                swp_drv:    resd 1 ; logical drive description table address of the swap drive/disk
2861 00030566 <res 00000004>                swpd_size:  resd 1 ; size of swap drive/disk (volume) in sectors (512 bytes).

2862 0003056A <res 00000004>                swpd_free:  resd 1 ; free page blocks (4096 bytes) on swap disk/drive (logical)
2863 0003056E <res 00000004>                swpd_next:  resd 1 ; next free page block
2864 00030572 <res 00000004>                swpd_last:  resd 1 ; last swap page block
2865
2866 00030576 <res 00000002>                alignb 4
2867
2868                                        ; 10/07/2015
2869                                        ; 28/08/2014
2870 00030578 <res 00000004>                error_code:    resd 1
2871                                        ; 29/08/2014
2872 0003057C <res 00000004>                FaultOffset:   resd 1
2873                                        ; 21/09/2015
2874 00030580 <res 00000004>                PF_Count:      resd 1 ; total page fault count
2875                                                               ; (for debugging - page fault analyze)
2876                                                       ; 'page_fault_handler' (memory.inc)
2877                                                       ; 'sysgeterr' (u9.s)
2878
2879                                        ; 29/04/2016 (TRDOS 386 = TRDOS v2.0)
2880                                        ; 22/08/2015 (Retro UNIX 386 v1)
2881                                        buffer:
2882 00030584 <res 00000008>                  resb   8
2883                                        readi_buffer:
2884 0003058C <res 00000200>                  resb   512
2885 0003078C <res 00000008>                  resb   8
2886                                        writei_buffer:
2887 00030794 <res 00000200>                  resb   512
2888                                        ; 24/10/2016
2889 00030994 <res 00000008>                  resb   8
2890                                        rw_buffer:
2891 0003099C <res 00000800>                  resb   2048  ; general purposed, r/w sector buffer
2892
2893                                        bss_end:
2894
2895                                        ; 27/12/2013
2896                                        _end:  ; end of kernel code
```